Machine learning-based anti-cancer drug treatment optimization

Dissertation
zur Erlangung des Grades
der Doktorin der Naturwissenschaften (Dr. rer. nat)
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes
von

Kerstin Lenhof

Saarbrücken
– 2024 –

Tag des Kolloquiums: 22.7.2024
Dekan der Fakultät: Univ.-Prof. Dr. Roland Speicher

Prüfungsausschuss

Vorsitzende des Prüfungsausschusses: Prof. Dr. Olga Kalinina
Berichterstatter: Prof. Dr. Hans-Peter Lenhof, Prof. Dr. Sven Rahmann, Prof. Dr. Gunnar W. Klau

Wissenschaftlicher Mitarbeiter: Dr. Michael Backenköhler

Abstract

Elves seldom give unguarded advice, for advice is a dangerous gift, even from the wise to the wise.

(Gildor, LotR)

Machine learning (ML) systems are about to expand into every area of our lives, including human healthcare. Thus, ensuring their trustworthiness represents one of today's most pressuring scientific and societal issues.
In this thesis, we present novel ML-based decision support tools for one of the most complex, prevalent, and mortal diseases of our time: cancer. In particular, we focus on developing trustworthy ML methods for predicting anti-cancer drug responses from personalized multi-omics data. Our methods encompass strategies to minimize the effect of data-related issues such as class or regression imbalance, to achieve the interpretability of the models, and to increase the reliability of the models.
Our first approach, *MERIDA*, is dedicated to interpretability: it delivers Boolean rules as output and considers a priori pharmacogenomic knowledge to a previously unconsidered extent. With *SAURON-RF*, we devised a simultaneous classification and regression method that improved the statistical performance for the under-represented yet essential group of drug-sensitive samples, whose performance has mainly been neglected in the scientific literature. Its successor, *reliable SAURON-RF*, provides a conformal prediction framework, which, for the first time, ensures the reliability of classification and regression with certainty guarantees. Moreover, we propose a novel drug sensitivity measure that addresses the shortcomings of the commonly used measures.

Kurzfassung

Elves seldom give unguarded advice, for advice is a dangerous gift, even from the wise to the wise.

(Gildor, LotR)

Systeme des machinellen Lernens (ML) sind im Begriff, in jeden Bereich unseres Lebens vorzudringen, inklusive der Gesundheitsversorgung. Dementsprechend ist die Sicherstellung der Vertrauenswürdigkeit dieser Systeme eine der größten gesellschaftlichen und wissenschaftlichen Herausforderungen unserer Zeit.
Diese Dissertation stellt ML-basierte Entscheidungshilfeverfahren für eine der komplexesten und am weitesten verbreiteten Krankheiten der Welt vor: Krebs. Der Fokus dieser Arbeit liegt hierbei auf der Entwicklung vertrauenswürdiger Vorhersagemodelle für die Wirksamkeit von Krebsmedikamenten basierend auf personalisierten *omics* Daten. Unsere Methoden umfassen dabei Strategien zur Minimierung der Auswirkungen datenbezogener Probleme wie Klassen- oder Regressionsungleichgewicht, zur Verbesserung der Interpretierbarkeit der Modelle und zur Erhöhung der Zuverlässigkeit der Modelle.
Unser erster Ansatz, *MERIDA*, ist der Interpretierbarkeit gewidmet: *MERIDA* liefert logische Regeln als Ausgabe und berücksichtigt dabei pharmakogenomisches a priori Wissen in bisher nicht betrachtetem Umfang. Mit unserem zweiten Ansatz, *SAURON-RF*, haben wir eine Methode zur gleichzeitigen Regression und Klassifikation entwickelt. *SAURON-RF* verbessert die Vorhersagekraft für die unterrepräsentierten Gruppe der arzneimittelempfindlichen Proben, deren Wichtigkeit bisher vernachlässigt wurde. Der Nachfolger, *reliable SAURON-RF*, nutzt konforme Vorhersage (conformal prediction), um die Verlässlichkeit der Klassifikation und Regression zu gewährleisten. Darüber hinaus schlagen wir ein neuartiges Maß für die Medikamentenwirksamkeit vor, welches Unzulänglichkeiten üblicher Maße behebt.

Acknowledgement

I don't know half of you half as well as I should like; and I like less than half of you half as well as you deserve.

(Bilbo Baggins, LotR)

In the past, I was in the fortunate position of making the acquaintance with a variety of people that, with their thoughts and deeds, managed to pave their way into my mind and stay. I dare say that my life would look distinctly different if I had not met them. Some of them supported me, some of them challenged me, and some - if not even all - of them did both. No matter how they influenced me, they definitely hold their stake in the genesis of this thesis! Based on my deep conviction that you truly made a difference in my life as a whole, I thank you (ordered chronologically by first significant appearance): Kurt and Regina Lenhof, Chiara Sommer, Nico Gerstner, Hans-Peter Lenhof, Nadja Grammes, Lukas Wallacher, Pia Scherer-Geiß, Jérémy Amand, Daniel Stöckel, Lara Schneider, Tim Kehl, Elena Arensкötter, Nadine Wilhelm, Robinson Ferrara, Cordula Alznauer, Lea Eckhart, Afnan Sultan, Niklas König, and last but not least Sebastian Roth.

Of course, this thesis is not only the result of several years of research by me but also of fruitful discussions inside and outside of lectures and collaborations with (former) colleagues and (fellow) students from Saarland University, whom I had the pleasure of working with. I definitely owe my gratitude to the Graduate School of Computer Science Saarbrücken and the following professors and their research groups (ordered alphabetically): Prof. Dr. Volkhard Helms, Prof. Dr. Andreas Keller, Prof. Dr. Hans-Peter Lenhof, Prof. Dr. Sven Rahmann, and Prof. Dr. Andrea Volkamer. Many thanks to all of my bachelor's and master's students as well (ordered alphabetically): Berit Andres, Georg von Arnim, Alizée Borsche, Lea Eckhart, Nina Engelkamp, Jakob Gemmel, Lutz Herrmann, Emma Hoffmann, Anke King, Carolin Mayer, Lisa-Marie Rolli, Francesco Schmitt, Johanna Schmitz, Georges Schmartz, Louisa Schwed, Pascal Stein, and Afnan Sultan.

Publications

Large parts of this dissertation are based on the following collection of articles I was involved in as a main contributor. The complete list of my publications can be found in Appendix A.

- ✭ **Lenhof, K.**, Gerstner, N., Kehl, T., Eckhart, L., Schneider, L., & Lenhof, H. P. (2021). MERIDA: a novel Boolean logic-based integer linear program for personalized cancer therapy. Bioinformatics, 37(21), 3881-3888. `https://doi.org/10.1093/bioinformatics/btab546`
- ✭ **Lenhof, K.**, Eckhart, L., Gerstner, N., Kehl, T., & Lenhof, H. P. (2022). Simultaneous regression and classification for drug sensitivity prediction using an advanced random forest method. Scientific Reports, 12(1), 13458. `https://doi.org/10.1038/s41598-022-17609-x`
- ✭ **Lenhof, K.**[†], Eckhart, L.[†], Rolli, L.M., Volkamer, A., & Lenhof, H. P. (2023). Reliable anti-cancer drug sensitivity prediction and prioritization. (Research-Square Preprint, currently under review in Scientific Reports) `https://doi.org/10.21203/rs.3.rs-3542373/v2`
- ✭ Eckhart, L., **Lenhof, K.**, Rolli, L.M., & Lenhof, H.P. (2023). A comprehensive benchmarking of machine learning algorithms and dimensionality reduction methods for drug sensitivity prediction. (under submission)
- ✭ **Lenhof, K.**, Eckhart, L.[†], Rolli, L. M.[†], & Lenhof, H. P. (2024). Trust me if you can: a survey on reliability and interpretability of machine learning approaches for drug sensitivity prediction in cancer. (under submission)

[†]These authors contributed equally to the respective work.

Contents

List of Figures

List of Tables

1 Introduction

Speak, friend, and enter.

(Inscription at the door of Moria, LotR)

We live in an era where data is interminably collected everywhere and subsequently digitally stored to become analyzed and leveraged. To name but a few examples: companies gather information on the purchase behavior of their customers to tailor the supply to the demand, social media platforms collect personal information to increase marketing revenues or even influence public opinion, and medical research centers and institutes monitor their patients to help improve prevention, diagnosis, prognosis, and treatment of diseases. The ever-increasing amount and complexity of the data that almost entirely prohibit a manual extraction of relevant patterns gave rise to automated and intelligent computerized data processing and helped artificial intelligence (AI) and especially the machine learning (ML) realm of AI to prosper vividly. Designing ML algorithms that fit the data but also satisfy the constraints imposed by (end) users such as usability, security, privacy, reliability, and interpretability can be extremely challenging. In the healthcare domain, where sensible personal data is used to guide decisions, and the data itself is already prone to technical and biological variance, none of the mentioned requirements should be neglected - posing an even greater challenge to developing ML-based systems.

In medicine, the ultimate goal has always been to deliver tailored decisions for all disease-related circumstances an individual may face [1]. Since the advent of technically advanced molecular measuring techniques, terms such as *personalized medicine*, *precision medicine*, and *individualized medicine* have been coined to describe this customization of healthcare based on molecular measurements of patients [1, 2] (cf. Figure 1.1). The hope is that a computer-aided integrative analysis of these fine-grained measurements - with AI - can shed light on the cellular machinery at a previously unprecedented scale. Subsequently, the gained insights and developed models should then help to guide therapy of complex diseases such as cancer [3, 4]. Cancer is a very heterogeneous class of diseases generally associated with abnormal cell growth and the ability to eventually spread to distant sites from the original tissue [5]. All across the world, it has a high incidence and mortality [6]. According to the German federal statistical office, it was the second most common cause of

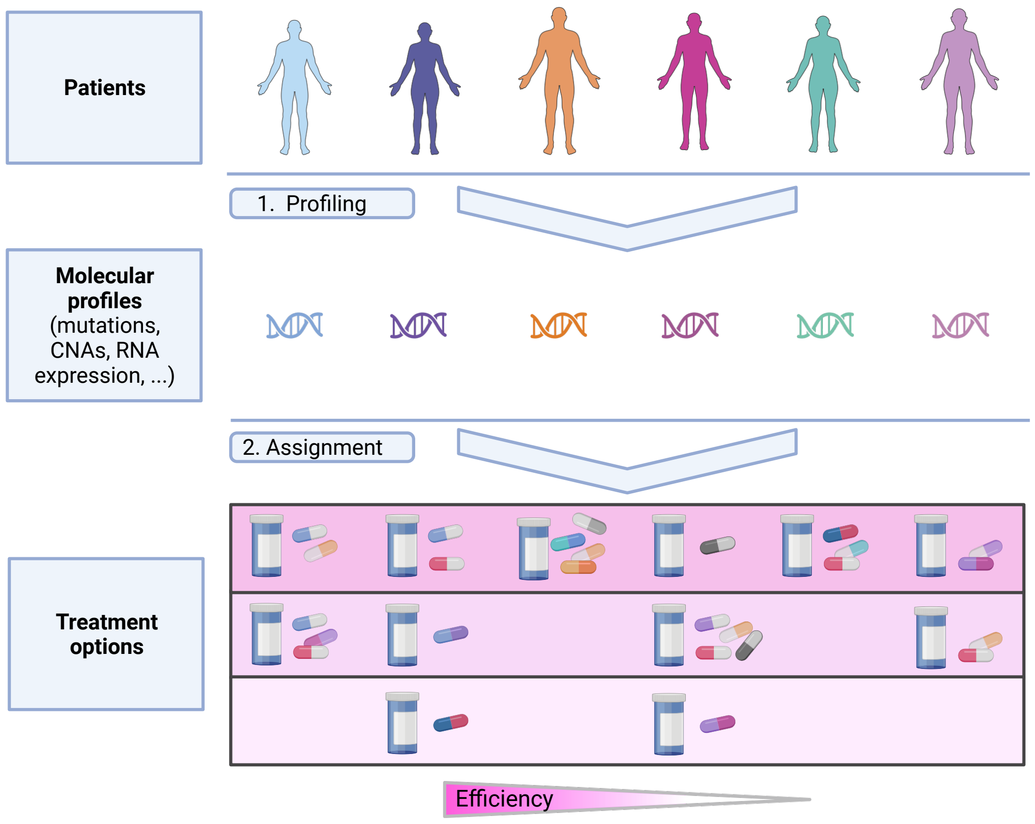


Figure 1.1: Personalized medicine. This figure exemplifies the concept of personalized medicine for our application case. Based on the interpretation of patient-specific molecular data, the optimal drug treatment can be determined. Created with BioRender.com.

death in Germany in 2022, accounting for 21.7% of deaths [7]. One reason for its high incidence and mortality is that it is a cellular disease (almost) everyone can develop without the need for exposure to specific triggers. Rather, it is the result of a multi-step evolutionary process from the complex interplay between the molecular condition of a person and the environmental exposure and lifestyle of that person. In particular, there is a risk accumulation with increasing age which is thought to occur because of an accumulation of damage on the hereditary material [8]. Due to the complexity of the highly individualized formation process, there exists a heterogeneity not only between patients and tumours but also within the same tumour. This heterogeneity, in turn, represents a major obstacle to treatment success: anti-cancer drug response is strongly affected by tumour characteristics, and the emergence of drug resistance is promoted [9, 10]. Another impediment to successful anti-cancer drug treatment is the limited possibility of testing the plethora of available drugs in clinical settings. Currently, it is neither ethically justifiable nor technically feasible to explore the space of all possible treatments exhaustively in real clinical applications.

As a remedy, biomedical research employs model systems such as patient-derived xenografts or cancer cell line panels to study cancer biology and cancer treatment since these can be more easily exposed to the existing multitude of (anti-cancer) drugs [11, 12] (cf. Figure 1.2). Over the last decades, especially cancer cell line panels have been proven to be a useful tool in that respect and therefore enjoy immense popularity for drug-screening community efforts [13, 14, 15, 16, 17]. The Genomics of Drug Sensitivity in Cancer (GDSC) is one of the largest publicly available cancer cell line panels. It contains drug screening data of hundreds of compounds applied to approximately 1000 cancer cell lines. In addition to this pharmacogenomic information, the GDSC database comprises molecular measurements on a diverse set of omics data for these cell lines, e.g., in terms of (epi)genomics and transcriptomics [14, 18, 19, 20, 15].

A diverse set of ML methods has already been applied to such panels to help elucidate the relationship between molecular cancer biology and anti-cancer treatment efficacy [11, 12]. The ultimate goal of these tools is to assist medical decision-making by matching patients to optimal drug treatments (cf. Figure 1.1). Here, the primary task is performing drug prioritization, i.e., identifying effective drugs and drug combinations and subsequently ranking these by their efficiency. In Figure 1.2, we schematically depict how drug prioritization can be performed using the current body of knowledge. The design of appropriate ML approaches has proven to be a great challenge, being affected by (the quality of) the data itself, the choice of methodology, and the requirements imposed by the exceptional diligence this task requests.

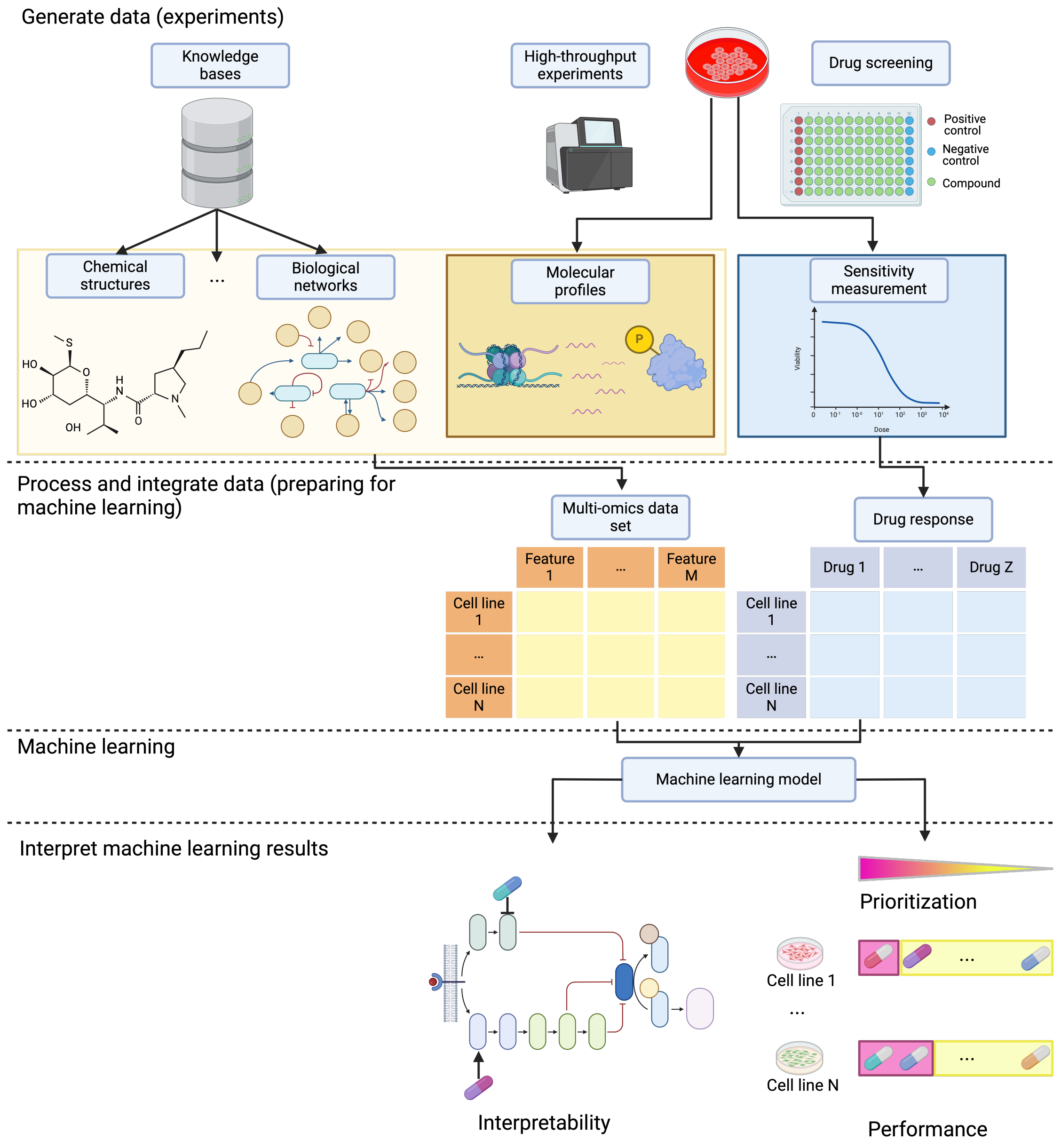


Figure 1.2: An overview of machine learning with anti-cancer drug response cell line data. In this figure, we depict a typical workflow for machine learning-based anti-cancer drug response prediction. Cancer cell line data from extensive high-throughput experiments typically represents the main data source. In addition, there exists a plethora of curated databases encompassing information of prior pharmaceutical, biological, and medical knowledge, including for example chemical structures of molecules, biological pathways, and known oncogenicity of genes. These heterogenous data sources need to be unified to become eligible to ML. After training, the resulting ML model has to be evaluated, e.g., in terms of interpretability and performance. Created with BioRender.com.

1.1 Scope of the thesis

This thesis addresses numerous unresolved issues for trustworthy anti-cancer drug response prediction using ML. We formulate anti-cancer drug response prediction as a supervised ML problem and develop methods to answer the following questions: (1) how to classify samples as sensitive or resistant (classification task), (2) how to predict continuous sensitivity values for samples (regression task), and finally (3) how to prioritize drugs for a given sample (prioritization task). While the former two tasks are frequently addressed in the corresponding literature, the latter has rarely been investigated despite being the ultimate goal. Our novel methods encompass strategies to minimize the effect of data-related issues such as high dimensionality and class or regression imbalance, to achieve the interpretability of the models, and to increase the reliability of the models.

Our *first contribution* is a *review of the ML landscape for drug sensitivity prediction in cancer*. Our categorization of the existing approaches into the four major ML branches, i.e., supervised, unsupervised, semi-supervised, and reinforcement learning, reveals that almost all approaches fall within the supervised ML realm and either focus on classification or regression but not both simultaneously. Moreover, we describe *properties that render ML methods trustworthy, particularly reliability and interpretability*. We find that reliability has hardly been considered for model development. On the other hand, the concept of interpretability is relatively often implemented. Yet, the concept is rather used intuitively, employing different connotations without a clear definition. Therefore, we unify the prevalent connotations and propose a *taxonomy of interpretability* that may serve as a sound and easily extendable basis for future research.

Our *second contribution* is the *MEthod for Rule Identification in multi-omics DAta (MERIDA)* [21]. It is a classification approach for distinguishing between sensitivity and resistance to a particular drug based on an *integer linear programming (ILP) formulation for synthesizing well-interpretable Boolean rules*. With MERIDA, we developed a method focusing on interpretability, which is already reflected in Boolean rules as model output choice. To increase the interpretability of our rules, we integrate *a priori pharmacogenomic knowledge in the form of drug response biomarkers into our models*, which has not been described in the literature before. For selecting and creating features of known relevance to cancer, we leverage information from multiple cancer-related databases: IntOGen [22], COSMIC [23], CIViC [24], On-

coKB [25], and CGI [26]. By using this feature selection and annotation strategy, we could incorporate pharmacogenomic a priori knowledge in our models while simultaneously reducing the dimensionality of the problem. A relatively well-known problem with drug response data is that the high specificity of targeted anti-cancer agents induces a skewed distribution of drug response values in favour of the resistant cell lines. This leads to a class imbalance when the inherently continuous drug response values are binarized. However, since the sensitive cell lines represent cases of treatment success, their correct identification is crucial for personalized oncology. MERIDA counteracts this class imbalance by employing sample-specific weights similar to those introduced by Knijnenburg et al. [27].
While class imbalance is usually addressed in the (drug sensitivity prediction) literature, the related problem for regression, i.e., regression imbalance, has rarely been investigated [28]. Here, ML methods reasonably estimate the mean of the distribution. In contrast, the upper or lower end of the distribution is systematically over- or underestimated while yet being of utmost importance. In our *third contribution*, we demonstrate for the first time that *regression imbalance is a central issue for a variety of ML methods, i.e., elastic net, boosting trees, neural networks, and random forests, when predicting anti-cancer drug responses* [29]. These methods systematically underestimate the response of the highly drug-sensitive cell lines. As a remedy for this problem, we then propose *SimultAneoUs Regression and classificatiON Random Forests (SAURON-RF)*. Unlike all competitor methods that either perform regression or classification, we suggest combining both tasks in SAURON-RF. By jointly performing regression and classification in a random forest regressor, we could also employ standard strategies against class imbalance, e.g., sample weights or upsampling, to mitigate regression imbalance. Furthermore, our results clearly indicate that our joint regression and classification outperforms mere regression, mere classification, and sequential execution of classification followed by regression. Contrary to MERIDA, SAURON-RF does not rely on literature-driven feature selection and creation but on selecting features statistically associated with the response. More specifically, we used a heuristic based on the maximum-relevance-minimum-redundancy principle, which aims to maximize the mutual information between the features and the response (maximum relevance) while minimizing the mutual information between the features (minimum redundancy) [30]. However, we did not exhaustively compare different dimensionality reduction strategies to justify our re-

spective choices. Likewise, most authors from the drug sensitivity prediction domain do not report how they opted for a specific dimensionality reduction technique, i.e., whether they tested other dimensionality reduction techniques and how these performed in comparison. However, given the variety of DR techniques and ML models they could be combined with, the correct choice might strongly influence the ML model performance. In our *fourth contribution*, we dedicated ourselves to gauging the influence of dimensionality reduction techniques on the performance of various ML methods. More specifically, we present a comprehensive *benchmarking study of nine DR techniques combined with four ML methods, ultimately resulting in more than 16,000,000 investigated models.* In total, our analyses reveal that relatively simple methods outperform more complex ones given the current data situation for drug sensitivity prediction: the elastic net (combined with PCA) outperformed all other methods, closely followed by random forests. Neural networks were not competitive, neither as an ML method nor as a DR technique.

While performance measures such as mean-squared error or accuracy are commonly evaluated to assess the quality of a model for drug response prediction, other criteria such as reliability have mainly been neglected. More specifically, only one regression approach (Fang et al.[31]) and no classification approach took reliability into account. In our *fifth and last contribution*, we designed and implemented a *framework for reliable drug response classification and regression, guaranteeing user-specified certainty levels through conformal prediction (CP).* To render SAURON-RF amenable to this approach, we developed a novel quantile regression algorithm adapted from Meinshausen et al. [32]. We could demonstrate that CP not only delivers the desired certainty guarantees but also successfully diminishes false predictions while retaining correct ones - ultimately improving model performance. Up to this point, we used the established IC50 value as a drug response measure. It is comparable across cell lines but not across drugs, preventing a straightforward drug prioritization for one cell line. Since this is a common issue among such measures, we propose a *novel measure with across-drug comparability, the CMax viability.* By combining this measure with our joint classification and regression method SAURON-RF and CP, we can ultimately perform *reliable drug prioritization*, which has not been described in the literature before.

1.2 Structure of the thesis

This thesis consists of 9 chapters. The remaining 8 chapters are structured as described in the following: In Chapter 2, we take a detailed look at cancer biology and treatment. We use a rough overview of the molecular biology of cells as a basis to subsequently illuminate alterations implicated in cancer development and progression. Afterwards, we discuss the effect of such alterations on treatment options, mainly targeted anti-cancer drugs. Chapter 3 contains information on the model systems typically used to quantify anti-cancer drug responses in biomedical research. We particularly focus on cell line-based systems since cancer cell line panels enjoyed the greatest popularity for large-scale drug screening. These data sets represent the basis for ML model training, as we conducted throughout this thesis. In Chapter 4, we describe the fundamental concepts of machine learning. We included basic definitions and descriptions of the four main machine learning branches, i.e. supervised, unsupervised, semi-supervised, and reinforcement learning, since all of them have already been applied to drug response prediction. However, we particularly thoroughly discuss supervised learning because all of our proposed anti-cancer drug sensitivity prediction approaches belong to this branch. In addition, we repeatedly contextualized the existing body of drug sensitivity methods in this chapter, which also constitutes our first contribution to this research area (*Trust me if you can: a survey on reliability and interpretability of machine learning approaches for drug sensitivity prediction in cancer*, Lenhof et al. 2024, under submission). Chapters 5-8 are dedicated to our four major contributions to anti-cancer drug sensitivity prediction:

- Chapter 5 - MERIDA (*MERIDA: a novel Boolean logic-based integer linear program for personalized cancer therapy*, Lenhof et al. 2021 [21]): a classifier with focus on interpretability, including a feature selection and annotation strategy using a priori drug response biomarkers

- Chapter 6 - SAURON-RF (*Simultaneous regression and classification for drug sensitivity prediction using an advanced random forest method*, Lenhof et al. 2022 [29]): a simultaneous classification and regression method that tackles class and regression imbalance

- Chapter 7 - Benchmarking (*A comprehensive benchmarking of machine learning algorithms and dimensionality reduction methods for drug sensitivity prediction*, Eckhart et al. , under submission): Benchmarking of 9 DR techniques combined with 4 ML techniques
- Chapter 8 - *Reliable* SAURON-RF (*Reliable anti-cancer drug sensitivity prediction and prioritization*, Lenhof and Eckhart et al. 2023 [33], under submission): extension of SAURON-RF with CP, introduction of the CMax viability, implementation of a reliable drug prioritization pipeline

In each chapter, we present the related literature and pinpoint how our approach contributes to solving particular problems: In Chapter 9, we confront the weaknesses of our approaches and discuss which research directions seem to be most promising to finally implement the dream of personalized anti-cancer drug treatment.

Authors' contributions
Many parts of this thesis represent the result of joint research efforts that have already been published in peer-reviewed journals. In the following chapters, these boxes are used to indicate my contributions to the respective works.

2 The biology of cancer and cancer treatment

During the 19th and 20th centuries scientists began to elucidate that what we perceive of an organism, i.e., its phenotype, is a direct result of the heritable entity, i.e., the genome, of that organism. Yet, since this discovery, the study of the relationship between phenotype and genotype has been the main focus of research in biology, partly because this research is expected to explain and subsequently cure many if not all diseases [34]. Indeed, if we, for example, consider the human organism alone, there are already more than 7,000 phenotypes (traits and diseases) that can be (partially) explained by genotypic variation and more than 4,000 genes with a variation directly linked to a phenotype [35]. The spectrum of disease complexity ranges from monogenic diseases, i.e., diseases caused by mutations in a single gene [36], to extremely complex diseases involving aberrations in multiple genomic regions as well as environmental and lifestyle factors. The clear causal relationship between genotype and phenotype in monogenic diseases enormously facilitates research in that area. However, monogenic diseases are rare [37], and common diseases such as asthma, diabetes, or cancer are often the result of a complex interplay of less penetrant, more common sequence variants with the environment and lifestyle of a person. Typically, these more common variants also exhibit their effects later in life (late onset) than rare variants (early onset), which further complicates research [34]. Cancer is the prototypical example of a common disease: it is a leading cause of death all over the world [38, 6] with a median age of 66 years at diagnosis according to the NCI [39]. In Germany in 2022, cancer accounted for 21.7% of deaths, making it the second most common cause of death (cf. Figure 2.1). While there exist genetic variants that clearly predispose to cancer development [41], the onset of tumorigenesis of most cancer types involves multiple genetic factors as well as lifestyle choices and environmental conditions. To date, more than 500 genes are known to be implicated in its development [42]. Given its high prevalence, it is not surprising that there is a high public interest in driving cancer research forward. However, its complexity still poses a great challenge to comprehensive treatment success.
In this chapter, we illuminate the biology of cancer and cancer treatment. To this

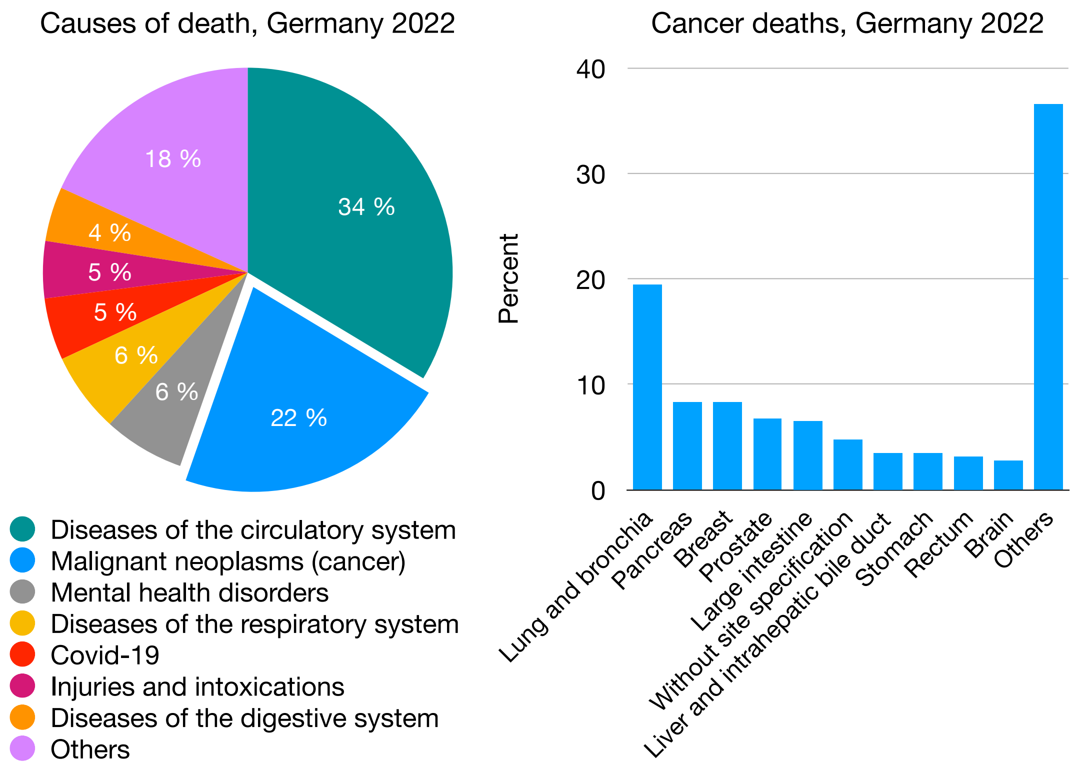


Figure 2.1: Causes of death, Germany 2022. On the left, the pie chart depicts the causes of death in Germany in 2022 according to the German Federal Statistical Office (destatis) [7]. On the right, the bar chart shows the breakdown of cancer deaths by cancer type as reported by the German Federal Statistical office [40].

end, we start with a brief description of the molecular biology of cells, followed by (epi)genetic aberrations that can deregulate cellular processes. Then, we discuss cancer, including its development, characteristics, and common treatment options. Note that the information presented in the following sections largely stems from the book *The biology of cancer* by Robert Weinberg [43], two landmark papers by Douglas Hanahan and Robert Weinberg on the biology of cancer [44, 45], and a novel paper of this series by Douglas Hanahan alone [46].

Authors' contributions
The structure and content of my bachelor's and master's thesis are partly used as basis of this chapter, while the text has been completely rewritten.

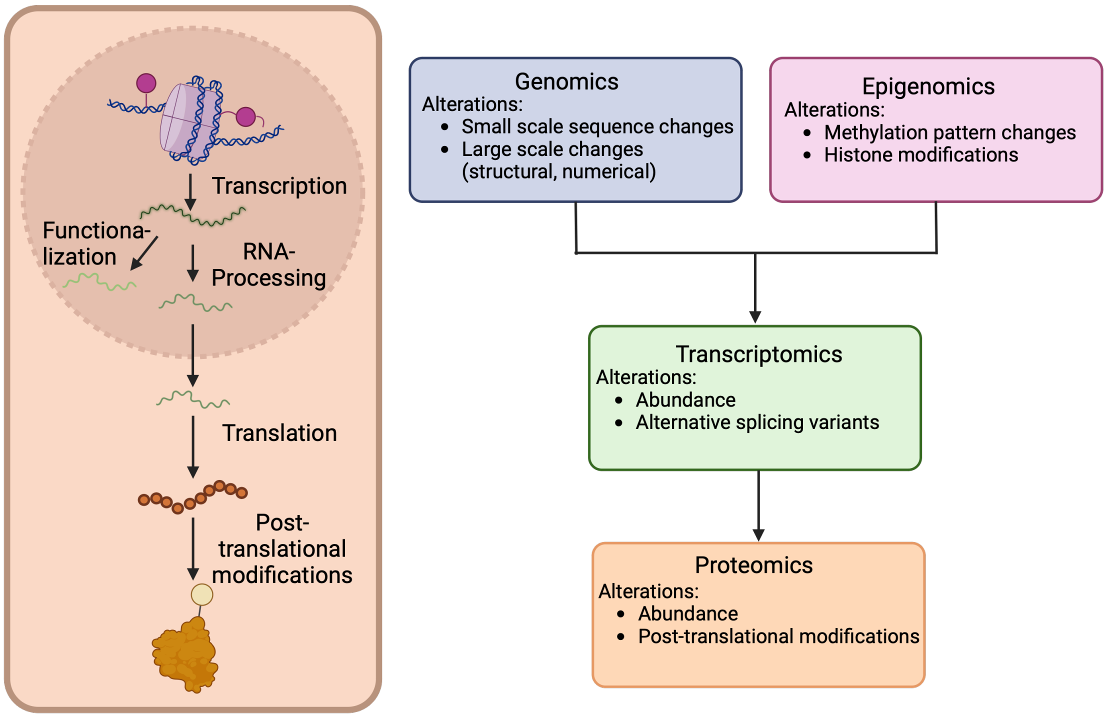

Transcription
Functiona-lization
RNA-Processing
Translation
Post-translational modifications
Genomics
Alterations:
• Small scale sequence changes
• Large scale changes (structural, numerical)
Epigenomics
Alterations:
• Methylation pattern changes
• Histone modifications
Transcriptomics
Alterations:
• Abundance
• Alternative splicing variants
Proteomics
Alterations:
• Abundance
• Post-translational modifications

Figure 2.2: Flow of genetic information. In this figure, we visualize the flow of genetic information, a simplified scheme typically used to describe the expression of phenotypes from the genetic material of a cell. In the boxes on the right, we give examples of measurable alterations that serve as indicators of disturbed information signaling in a cell. Created with BioRender.com.

2.1 The flow of genetic information in human cells

It is estimated that the human body consists of approximately 3×10^{13} cells [47], distributed across more than 400 documented cell types [48] that fulfil distinct functions. All of these cells are descendants of the same fertilized egg, the zygote, which has the ability to differentiate into any cell type of the human body [49]. The development of the different cells from the zygote is a highly regulated process whose description is out of the scope of this thesis. However, we will briefly describe the main cellular processes that enable the expression of different phenotypes from the same genetic material.

The genome level: Traditionally, the flow of genetic information (cf. Figure 2.2) is used as a simplified scheme to explain how the hereditary information is translated to molecules that fulfil the variety of functions within a cell. In this scheme, the entirety of desoxyribonucleic acid (DNA) in the nucleus, i.e., the genome, is the information carrier. It is a double-stranded helix with each helix strand consisting of a sequence of nucleotides, each comprising a phosphate group, the 5-carbon sugar desoxyribose and one of the following four nucleobases: guanine, adenine, cytosine, and thymine. Roughly speaking, the order of the nucleobases determines, which functional products can be synthesized [50]. Moreover, some sequences, such as promoters, silencers, or enhancers, are involved in the regulation of the synthesis. Apart from that, sequences with no known or potentially no function exist. Note that mitochondria also contain hereditary information in form of DNA. However, the organization and inheritance mechanism of the mitochondrial genome deviate from that of the nuclear genome.

The epigenome level: The DNA is densely packed in the nucleus in a form known as chromatin. The chromatin is formed by the interaction of the DNA with DNA-binding proteins, mainly histones. The basic unit of packed DNA is the nucleosome, a histone octameter with 146 base pairs of DNA wrapped around it [51]. Linker histones bind to the nucleosomes and allow for the building of the next higher level of ordering, the chromatosomes [52] that then spatially organize further involving other proteins, e.g., Polycomb, and non-coding RNAs [53]. The regulation of this packing, whose study is known as epigenetics [53], governs the (in)activation of genomic regions by changing sequence accessibility. Known regulatory mechanisms include but are not limited to the methylation of the DNA itself, chromatin remodelling, covalent histone modifications, or incorporation of histone variants [51].

The transcriptome level: If the DNA at a specific genomic region encoding for a functional product is accessible, the transcription of the DNA can be initiated. The transcription is catalyzed by an enzyme called RNA polymerase, that usually requires the interaction with regulatory elements, e.g., transcription factors, to start synthesis. Thereby, transcription factors can either act as activators or repressors of transcription. Besides the direct stabilization or blocking of the interaction between the RNA polymerase and the DNA, transcription factors can regulate transcription using a variety of other mechanisms. The product of transcription is some form of ribonucleic acid (RNA) that is complementary to the originating DNA sequence. RNA is a single-stranded sequence of nucleotides, each consisting of a phosphate group, the 5-carbon sugar ribose, and one of the following four bases: guanine, adenine, cytosine, and uracil (as the equivalent to thymine in DNA). A plethora of different RNAs can be distinguished. Yet, the most fundamental distinction is the one between (precursor) messenger RNA ((pre-)mRNA) and non-coding RNA types. The precursor mRNA undergoes post-transcriptional processing before the mature mRNA is transported to the ribosomes in the cytoplasm, where it becomes translated into proteins. Most prominently, in a process termed alternative splicing, specific segments of the sequence can be left or removed from the premature mRNA, enabling the generation of a set of mature mRNAs from the pre-mRNA, ultimately not only increasing mRNA diversity but also protein diversity [54]. Non-coding RNA types may also be processed; however, they are not translated to proteins and exhibit a variety of other functions in the cell.

The proteome level: Once the mRNA reaches the ribosomes in the cytoplasm, the mRNA can serve as the blueprint for synthesising a protein. More specifically, three consecutive nucleobases, called codon in this context, are translated to one amino acid by a ribosome. Each codon specifically encodes one amino acid. However, the code is degenerate: several codons can encode the same amino acid [55]. The translation starts at a specific codon, the so-called start codon. Then, the ribosome traverses the RNA-sequence codon by codon until a codon indicating termination, a so-called stop codon, is encountered. Note that one nucleobase is only part of one codon, i.e., codons are non-overlapping. The resulting amino acid sequence is called the primary structure. Interactions between the amino acids then cause the protein to fold into an energetically favourable 3D conformation, a process that helper proteins, the chaperones, can also assist. Proteins can be subject to post-translational modifications, i.e., they can be covalently modified after translation. These modifications can significantly alter the protein structure, function, or activity [56]. Some proteins need to further assemble into protein complexes to fulfil their function.

2.2 Aberrations

Within the human population, a significant amount of natural variation occurs, leading to the manifestation of unique molecular processes and, subsequently, phenotypic traits per individual [57]. The majority of common genetic variations is supposedly not severely affecting the correct functioning of the human organism. However, many of them can still contribute to disease progression, especially in complex diseases [58]. While common genetic variants already reside in the popu-

lation for some generations, they arise through the same mechanisms that can lead to cancer. In the following, we briefly discuss these mechanisms and differentiate between different classes of alterations. Most information in this section is based on (chapter nine of) a genetics book by Jochen Graw [59].
An alteration of the nucleic acid sequence in the genome of an organism is called mutation [59]. The classification system of mutations is complex. In the following, we discuss classifications based on cause, hereditability, size, and effect.
Cause: In terms of cause, we can distinguish between spontaneous mutations and induced mutations. Spontaneous mutations encompass molecular decay, errors introduced during DNA repair and errors introduced during DNA replication, e.g., error-prone translesion synthesis [60]. Some regions in the genome are particularly susceptible to mutations during DNA repair or replication, e.g., repetitive sequences such as expanding triplets or multiplets [61]. In contrast to these spontaneous mutations, induced mutations have an exogenous source from the environment. Examples of such sources are radiation, chemicals, viruses and bacteria.
Hereditability: The human organism consists of two fundamental cell types: germ cells (ova and spermatozoa) and somatic cells (cells of any other cell type of the human body). Depending on whether a mutation affects a germ or a somatic cell, different consequences regarding hereditability can be observed. Germline mutations can be passed on to the offspring. In that case, all somatic cells of the newly developed organism will carry this alteration. Mutations occurring in the soma can solely be passed on to descendants of the mutated cell of the respective organism.
Size: Yet another means to classify mutations is the size of the affected genomic region. Typically, the scientific literature distinguishes between rather small-scale changes (e.g., substitutions, insertions, or deletions of a few nucleotides) and large-scale changes affecting large parts of the genome. Large-scale alterations are usually further subdivided into structural alterations, i.e., rearrangements (inversions and translocations) and copy number alterations (deletions or duplications) of genomic regions, and numerical alterations, i.e., changes in the number of chromosomes (fusions, fissions, aneuploidies, and polyploidies).
Effect: None of the previous classifications says anything about the effect of a particular mutation on a functional product or the phenotype of an affected cell more generally. In principle, one distinguishes between mutations occurring in coding regions, i.e., mutations potentially capable of directly affecting functional products,

and mutations occurring in non-coding regions, i.e., mutations potentially capable of indirectly affecting functional products. Small-scale mutations in protein-coding regions (substitutions, insertions, and deletions of one or a few nucleotides) are further sub-classified into synonymous and non-synonymous mutations. Synonymous mutations (usually substitutions) are called silent since - due to the degeneracy of the genetic code - the resulting codon encodes for the same amino acid as the original codon. However, such mutations can still influence the phenotype of a cell, e.g., by altering the folding of the mRNA and, hence, the translation efficiency. Non-synonymous mutations can have various effects on the resulting protein. Usually, we can differentiate between mutations resulting in amino acid exchange(s), protein truncation, protein elongation, or no protein product. An additional effect-based classification on the protein level is the partition into gain-of-function and loss-of-function mutations. Gain-of-function mutations improve the activity/function of the protein or introduce a novel function, while loss-of-function mutations decrease the protein activity.

Historically, the term mutation was coined to describe changes in the genome of an organism. Since research has shown that epigenetic modifications can also be passed on from a cell to its descendants (from mother to daughter cells and from generation to generation) [62], the term mutation is occasionally expanded to epigenetic marks as well. Indeed, mutations of the epigenome, e.g., histone modifications or methylation pattern changes, affect cells similarly to genomic mutations and, thus, often contribute to the onset of complex diseases, including cancer [63]. For example, the hypermethylation of cytosines in a CpG context in promoters is linked to gene inactivation [64]. Similarly, alterations in histone modifications can regulate the accessibility of DNA, potentially activating or repressing the transcription of a gene [65].

2.3 Cancer

Cancer is a class of diseases characterized by abnormal and uncontrolled growth of cells. The cancerous cells are descendants of healthy cells that transformed into these malignant ones by a complex multi-step process involving multiple cell generations affected by various (epi)genomic aberrations. In the following, we outline this process and describe the commonalities shared between different cancer types using the conceptual framework by Hanahan and Weinberg, known as *the hallmarks of cancer*, as basis.

2.3.1 Cancer development

The (epi)genome of somatic cells is repeatedly subject to spontaneous or induced alterations during the lifespan of an organism (cf. Section 2.2). To prevent such mutations from manifesting, there exist cellular damage detection and repair systems that maintain the integrity and stability of cells. However, not all alterations can be eliminated successfully and, thus, become part of the (epi)genome. While most alterations will have little effect, and some may even be disadvantageous for cell survival, others may possess favourable traits under specific circumstances. These cells may still be detected and eliminated by the immune system. Nevertheless, the more time passes, the more likely such mutations occur, manifest, and accumulate, ultimately significantly altering cell function. This continuous process of natural selection acting upon mutations in somatic cells can be interpreted as a Darwinian-like evolution within a population. Progressively, cell populations with abnormal growth potential, i.e., cancer, can develop. The process of cancer formation is called carcinogenesis and depends not only on the genetic predisposition but also on environmental exposure and lifestyle choices of a person. Carcinogenesis usually requires time, and thus, it is not surprising that cancer risk increases with age. The exact period until a specific individual develops a particular type of cancer is barely foreseeable. However, some germline mutations may expedite this process since they clearly predispose to cancer development [66].
As already indicated in the previous section, not all mutations actively contribute to carcinogenesis. Therefore, one distinguishes between driver mutations, which promote(d) tumour growth and have been positively selected during the tumour evolution, and passenger mutations, which are considered happenstances [67, 68]. Genes that can be affected by driver mutations are known as cancer (or driver) genes [67, 68]. Typically, we can classify them as belonging to one of the following two non-exclusive categories: (proto-)oncogenes and tumour suppressor genes [69]. Proto-oncogenes are genes positively involved in cell growth. If their activity or expression is increased through some mutational event (usually a gain-of-function mutation), they become oncogenes that help cells to grow and divide disproportion-

ally [69]. On the contrary, tumour suppressor genes are mostly negative regulators of cell growth. Upon deactivation (e.g., via loss-of-function mutations), they lose the capability to serve as gatekeepers of cell growth. While mutations in proto-oncogenes are usually dominantly acting, i.e., only one mutated allele is sufficient to exhibit a malfunctioning phenotype, mutations in tumour suppressor genes are typically recessively acting, i.e., both alleles of the corresponding gene have to be deactivated [67].

2.3.2 Cancer classification and common characteristics

Clearly, the cell type of origin and its associated cell type-dependent gene expression play a role during carcinogenesis and also in the final appearance of the tumour. This observation gave rise to a cancer classification system based on cell type: on the highest level, carcinomas (epithelial origin), sarcomas (mesenchymal origin), leukaemia and lymphomas (haematopoietic and lymphoid origin), and neuroectodermal tumours (originating from central and peripheral nervous system cells) can be distinguished. Other classification systems based on the primary site of origin or stage of the tumour also exist. However, samples from different classes are often jointly analyzed in a pan-cancer manner because of molecular and phenotypical similarities across different cancer types. Indeed, the same molecular mechanisms are implicated in carcinogenesis and therapy responsiveness of tumours from different classes.

In *the hallmarks of cancer*, Hanahan and Weinberg unify these cellular commonalities between different cancer types [44, 45, 46]. Currently, their framework consists of the ten most protruding and common characteristics of cancer cells, which they refer to as *hallmarks*, as well as four so-called enabling mechanisms that allow for the acquisition of the hallmark capabilities. In Table 2.1, we summarize the *hallmarks of cancer* as envisioned by Hanahan in the latest version [46]. Briefly, incipient cancer cells manage to escape the state of tissue homeostasis, i.e., the persistently maintained balance between cell regeneration and death in healthy tissue. To this end, they need to evade inter- and intracellular growth-suppressing and death-inducing signals, establish elevated levels of proliferative signalling, and ensure a stable nutrient supply. Moreover, they need to obtain the capability to cross tissue boundaries, ultimately allowing the cancer to spread to distant sites of the human body. The

Table 2.1: The latest version of the *hallmarks of cancer*. In this table, we provide an overview of the *hallmarks of cancer* as envisioned by Hanahan in the latest conceptual update. We provide the name of the property (column 1), information on whether it is a hallmark or enabling mechanism (column 2), information on whether it is an established or emerging property (column 3), and an explanation or example of the property (column 4).

Property name	Type	Established	Explanation/example
Resisting cell death	hallmark	✓	avoidance or inactivation of apoptotic programs, e.g., by inactivation of DNA damage sensors that trigger apoptosis
Deregulating cellular metabolism	hallmark	✓	adapting the energy metabolism to the needs of the cancerous tissue
Sustaining proliferative signaling	hallmark	✓	disturbance of homeostasis, e.g., by increasing production of growth factors or growth factor receptors
Evading growth suppressors	hallmark	✓	deactivation of tumour suppressors, e.g., RB or TP53
Avoiding immune destruction	hallmark	✓	circumventing detection and subsequent destruction by immune cells
Enabling replicative immortality	hallmark	✓	counteracting telomere erosion, e.g., by reactivation of telomerases adding suitable segments to the end of chromosomes
Activating invasion and metastasis	hallmark	✓	obtaining the ability to invade other tissues and distant sites, e.g., by disturbing the attachment to other cells and the extracellular matrix
Inducing or accessing vasculature	hallmark	✓	ensuring nutrient supply and possibilities for waste disposal by the sprouting of new blood vessels, e.g. by deregulation of vascular endothelial growth factor (VEGF) signaling
Senescent cells	hallmark	✗	the role of senescent cells is only partly understood, senescence-associated secretory phenotype (SASP) is, however, pro-inflammatory
Unlocking phenotypic plasticity	hallmark	✗	evading or reverting terminal differentiation by dedifferentiation, blocked differentiation, or transdifferentiation
Genome instability and mutation	enabling mechanism	✓	mutations that confer growth advantages can manifest in the genome and subsequently be propagated to daughter cells
Tumour-promoting inflammation	enabling mechanism	✓	infiltration of cancerous tissue with immune cells that foster tumour growth, e.g., by supply of growth-supporting molecules
Nonmutational epigenetic reprogramming	enabling mechanism	✗	alterations of epigenetic marks confer growth advantages
Polymorphic microbiomes	enabling mechanism	✗	microbial species within the tumour microenvironment but also important host microbiomes more generally (e.g., the gut microbiome) influence carcinogenesis

main driving forces of this malignant transformation were already discussed in the previous section on cancer development, i.e., (epi)genomic aberrations that confer growth advantages to cells. In the model of Hanahan and Weinberg, this corresponds to the two enabling mechanisms called *genome instability and mutation* and *nonmutational epigenetic reprogramming*. In addition to these two mechanisms, Hanahan and Weinberg consider inflammation an important enabling factor: immune cells can serve as suppliers of bioactive molecules, amongst others, directly fostering cell growth, limiting cell death, or supporting angiogenesis. Moreover, the tumour microbiome and the microbiotic state of the organism as a whole can positively or negatively influence inflammation, genome stability, and response to therapeutic interventions, rendering microbiota another enabling mechanism of carcinogenesis.

2.3.3 Drug-based anti-cancer therapy

To date, there are still three main treatment options for cancer: surgery, radiotherapy, and systemic therapy. Which therapy or combination of therapies to choose depends on various factors, including the localisation and stage of the cancerous disease, as well as the patient's general health status. While early-stage tumours can typically be treated effectively with surgery alone, therapeutic modalities must be administered concurrently or sequentially in many other cases [70]. In particular, if the cancer has already metastasized, delivery through the bloodstream, i.e., systemic therapy, is indispensable. Systemic therapy can currently be divided into classical chemotherapy, targeted therapy and immunotherapy [10]. Each of these three approaches follows a specific principle to eradicate cancer cells. However, the boundaries between the different approaches are often blurred.
Chemotherapy: In the previous sections, we thoroughly discussed that cancer cells are characterized by their abnormal ability to grow and proliferate. Classical chemotherapeutic agents make use of this fact and disrupt the cell cycle by different mechanisms of action [70]. Consequently, they attack all rapidly dividing cells in the human body [70], which is often associated with relatively high toxicity.
Targeted therapy: With a growing understanding of the genetic causes of cancer, the development of targeted agents started: these substances were designed to specifically target disrupted signalling cascades with pivotal roles during tumour progression and maintenance, e.g., molecular pathways helping to establish the hallmark capabilities. Consequently, these drugs should be less damaging to healthy tissue than classical chemotherapy. Typically, the direct targets of these

drugs are the protein products translated from oncogenes, the so-called oncoproteins, or signal-transducing proteins downstream of cancer driver genes [71]. While tumour suppressor genes are not directly targetable, proteins from genes in a synthetic lethality relationship with a tumour suppressor gene can also be drugged [71]. The PI3K-AKT-mTOR pathway is a prototypical example of an attractive target pathway: it contains a variety of (proto-)oncogenes and tumour suppressor genes mutated across a variety of cancer types [72, 73, 74]. Here, the mammalian target of Rapamycin (mTOR), a member of the PI3K-AKT-mTOR pathway, has been a main focus of anti-cancer drug development [75].

Immunotherapy: Immunotherapies are designed to modulate the immune response of the patient. They include checkpoint blockade [10], adoptive T-cell therapy using either tumour-infiltrating lymphocytes (TIL) or modified T-cells expressing T-cell receptors (TCR) or chimeric antigen receptors (CAR) [76, 77], and cancer vaccines [78]. While the review of all of these immunological therapeutic strategies is out of the scope of this thesis, we would like to mention that monoclonal antibodies used as checkpoint blockade inhibitors (directed against CTLA-4 , PD-1, or PD-L1) can be regarded as targeted agents as well.

Independent of the chosen systemic treatment strategy, the administration of a single drug (monotherapy) is often not sufficient to successfully treat cancer: even if the tumour cells responded sensitively to initial treatment, intra-tumour heterogeneity could allow for rapid adoption to selective pressure induced by the drug, finally leading to drug resistance. Consequently, drug combination strategies are developed as countermeasures [10]. Here, the rationale is to exploit several non-overlapping, possibly complementary mechanisms of action to avoid escape from therapeutic success. Still, the same mechanisms that lead to the failure of single agents also occur for combination strategies. Briefly summarized, the complexity of the disease caused by the extremely individual interplay of the (epi)genomic condition of the affected human, as well as lifestyle choices and environmental influences, leads to a gap in knowledge about the disease. Cancer biology and the biology of anti-cancer treatment efficacy are far from being fully elucidated. To complete our knowledge, model systems, e.g., cancer cell lines, may be employed since they can more easily be exposed to the multitude of already existing anti-cancer drugs as well as drug candidates.

3 Quantifying anti-cancer drug sensitivity

Often, it is nearly impossible to study human diseases in human beings, e.g., because of ethical concerns regarding unknown effects of new treatments or because of extremely long putative observation periods. Therefore, biomedical research typically relies on model systems such as Caenorhabditis elegans, Drosophila melanogaster, Mus musculus, or cell cultures instead [79]. While each model system has a specific set of (dis)advantages - rendering it particularly suitable for particular use cases - all share several favourable properties compared to humans. These systems are relatively easy to establish and maintain in the laboratory, they replicate fast while exhibiting a high pheno- and genotypic stability, their cultivation cost is low, and significantly fewer ethical concerns arise during their use.

For cancer research and especially anti-cancer drug screening, 2D cell cultures derived from tumours and, more recently, patient-derived xenografts (PDX), i.e., immunodeficient mice engrafted with cancer cells from tumours, were intensively investigated [80, 79, 81]. However, neither 2D cell cultures nor PDX are able to represent the physiopathology of human tissue and the tumour microenvironment. Consequently, they at least partially fail to mimic in vivo anti-cancer drug responses [82, 80]. Therefore, 3D cell cultures, e.g., in the form of patient-derived organoids (PDO), were developed over the past few years [80, 83].

However, most publicly available anti-cancer drug screening databases are still based on 2D cell cultures (cf. Table 3.1 for an overview of datasets in the public domain). We are unaware of a single database with large-scale organoid data and only one with PDX data, i.e., NIBR-PDXE [84]. Yet, NIBR-PDXE is still comparatively small compared to 2D cell line panels such as the Genomics of Drug Sensitivity in Cancer (GDSC) database. Since machine learning algorithms benefit from a large number of samples, we decided to analyze the GDSC database, one of the largest databases in terms of available samples and screened drugs.

In the following, we will first present the typical workflow of a drug screen on a 2D cell culture. Then, we describe the GDSC database in more detail.

Table 3.1: Publicly available drug screening data sets. In this table, we provide an overview of the existing large-scale drug screening databases of the public domain. See Section 3.1.1 for an explanation of viability assays.

Database	Version	Type	#Samples	#Compounds	Assay
NCI60 [13]	– , developed 1980 - today	Monotherapy for cancer cell lines	60	>100,000 [85]	Sulforhodamine B [86]
GDSC1 [18, 19]	Release 8.4 (July 2022), developed 2010-2015 [87]	Monotherapy for cancer cell lines	970	403	Resazurin or Syto60
GDSC2 [87]	Release 8.5 (October 2023), developed 2015 - today [87]	Monotherapy for cancer cell lines	969	297	CellTiter-Glo
CTRPv1 [88]	– (corresponding paper published 2013 [88])	Monotherapy for cancer cell lines	242	185	CellTiter-Glo
CTRPv2 [16, 89]	– (corresponding papers published in 2015 and 2016 [16, 89])	Monotherapy for cancer cell lines	860	481	CellTiter-Glo
Primary PRISM repurposing data set [90]	– (corresponding paper published 2020)	Monotherapy for cancer cell lines	578	4,518	PRISM assay
Secondary PRISM repurposing data set [91]	– (no dedicated publication yet)	Monotherapy for cancer cell lines	499	1,448	PRISM assay
NIBR-PDXE [84]	– (corresponding paper published in 2015 [84])	Monotherapy and combination therapy for PDX	approx. 270	38	PDX clinical trial
NCI-ALMANAC [92]	– (corresponding paper published 2017 [92])	Combination screen for cancer cell lines	60	pairwise combinations of 104	Sulforhodamine B, CellTiter-Glo
O'Neil [93]	– (corresponding paper published 2016 [93])	Combination screen for cancer cell lines	39	pairwise combinations of 38	CellTiter-Glo
AstraZeneca DREAM challenge [94]	– (corresponding paper published 2019 [94])	Combination screen for cancer cell lines	85	pairwise combinations of 118	Sytox Green

3.1 Cell viability assays applied to 2D cell cultures

When treating cancer patients with anti-cancer drugs, one is interested in efficiently eliminating tumorous tissue while minimizing side effects. Thus, approved drugs have undergone a complex development pipeline encompassing numerous stringent tests to assure their usefulness and safety. Viability screening on model systems such as 2D cell cultures is one key technique in the early stages of the drug development pipeline, helping to identify candidate drugs [95]. Roughly speaking, a cancer cell line is exposed to different concentrations of a compound, and for each concentration, the number of living cells, i.e., the cell viability, is determined. By investigating the change in viability, the usefulness of the drug (e.g., efficacy and efficiency) can be estimated. Especially when combined with multi-omics measurements on the corresponding cell lines, these screens can help gain insights into the molecular biology of treatment response and design predictive machine learning models that can then be part of the drug development pipeline or medical decision support systems.

3.1.1 In vitro workflow

There exist various experimental methods to measure cell viability [96, 97] (cf. Table 3.1). These methods leverage a differentiating characteristic of either live or dead cells to generate a detectable signal. For example, the CellTiter-Glo assay uses ATP as a marker of metabolically active, i.e., viable, cells [97]: under the consumption of ATP from viable cells, the luciferase enzyme converts luciferin to oxyluciferin, a reaction in which light proportional to the amount of available ATP is emitted. In the following, we describe the general components and proceedings of drug screening based on viability assays. Whenever specific assay or workflow details are required, we refer to the information from the GDSC database.
After the selection of cell cultures and compounds of interest, a viability assay starts with the preparation of a microtiter plate, a plate with a grid of small test tubes (wells), in which the assay will be performed (cf. Figure 3.1). A careful plate design is required to avoid technical biases because of the arrangement of experiments on the plate. Note that such a viability assay is typically performed simultaneously for several cell lines and compounds. Given a specific cancer cell line and one particular

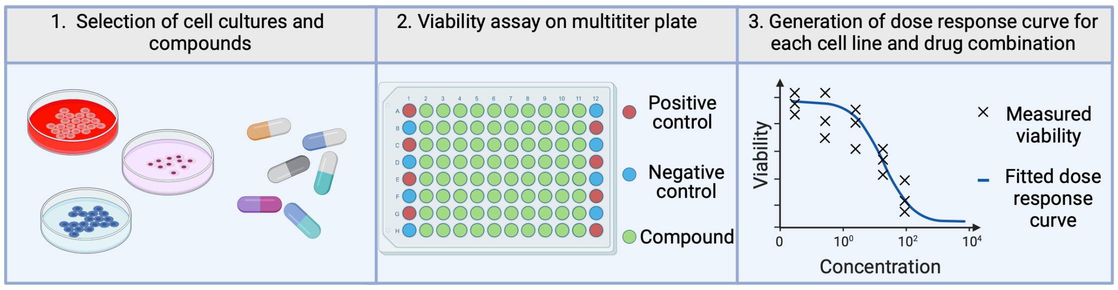

1. Selection of cell cultures and compounds
2. Viability assay on multititer plate
Positive control
Negative control
Compound
3. Generation of dose response curve for each cell line and drug combination
Viability
Concentration
0
10⁰
10²
10⁴
Measured viability
Fitted dose response curve

Figure 3.1: Cell viability measurement. This figure visualizes the workflow of in vitro cell viability measurement for 2D cell cultures. Created with BioRender.com.

drug, we can distinguish between two general types of well content:

- Wells with a treated sample: Initially, these wells contain medium and cells. Later, they will be incubated with a specific concentration of the investigated compound.
- Wells serving as control: These wells will not contain the investigated compound and serve as control instances that measure the overall validity of the experiments. Depending on the used assay, there are different sub-types of control wells with various compositions. In the GDSC database, we can distinguish between [98]:
  - Wells serving as negative control: These wells contain medium and cells. They serve as representatives for 100% viability.
  - Wells serving as positive control: These wells contain medium only and are also denoted as blanks. They represent 0% viability.

To ensure comparability between the wells with a particular cell culture, each such well should contain a nearly identical number of cells.
After the preparation of the microtiter plate(s), the compounds are added to the respective wells and the wells are incubated with them for a fixed time span, e.g., 72 h [98]. Typically, not only one concentration of a compound is tested, but several ones. These different concentrations can be obtained by a dilution series [98]. Moreover, the same experimental conditions are investigated several times to ensure reproducibility and robustness of the experiment series.
After the incubation period, the viability assay components are added to the wells,

and a detectable signal, e.g., luminescence for CellTiter-Glo, is emitted. The signal can be measured and is then processed as described in the ensuing section.

3.1.2 Processing of intensity values

The performed viability assay results in one intensity value for each well, and each intensity value corresponds to the amount of viable cells within a well. A mandatory first processing step after conducting a biological experiment is to validate its success by performing background correction and quality control. The viability assay already contained positive and negative control wells for quality control. The raw intensity values of these wells were assumed to represent 0% and 100% viability, respectively. Thus, the average intensity value of the positive control should be low, most likely lower than all other values, since no living cell was seeded in those wells. The average intensity of the negative control wells should be comparatively high. A commonly applied background correction for viability assays also includes the use of the control wells. It is defined by subtracting the average intensity value of all positive control wells from other wells [99]. More specifically, we can accomplish this as described in the following. For the simplicity of this description, we assume that each experiment has only been conducted once, i.e., no replicates have been generated. Let N be the number of investigated cell lines, K be the number of tested compounds, and $C_1, \ldots, C_K$ be the respective sets of tested concentrations for each compound. Furthermore, let $\bar{n}_i$ be the average of the negative control wells belonging to the i-th cell line and let $\bar{p}$ be the average of the positive control wells. Note that depending on the plate design, $\bar{p}$ could be calculated based on all positive control wells or a selection of positive control wells. Furthermore, let v_{ikx} be the intensity of the i-th cell line treated with compound k at concentration x. We then calculate the corrected intensities $\tilde{n}_i$ and $\tilde{v}_{ikx}$ as follows

$$\tilde{n}_i = \bar{n}_i - \bar{p}, \quad \forall i \in \{1, \ldots, N\} \tag{3.1}$$

and

$$\tilde{v}_{ikx} = v_{ikx} - \bar{p}, \quad \forall i \in \{1, \ldots, N\}, \forall k \in \{1, \ldots, K\}, \forall x \in C_k. \tag{3.2}$$

After background correction, we can assess the effect of a treatment by a comparison between corrected intensity of the treated well and the corresponding corrected negative control, i.e., we can calculate the so-called relative viability

$$r_{ikx} = \frac{\tilde{v}_{ikx}}{\tilde{n}_i}, \quad \forall i \in \{1, \ldots, N\}, \forall k \in \{1, \ldots, K\}, \forall x \in C_k. \tag{3.3}$$

The relative viability is 1 if the treatment did not affect cell growth, and ≤ 1 if an inhibition occurs. Also, values ≥ 1 can be observed because of experimental fluctuations or if the treatment fosters cell growth. Negative values should usually not occur. However, if a compound eradicates all living cells, it may occasionally happen that the numerator is negative, i.e., if the average positive control value $\bar{p}$ is larger than the observed viability v_{ikx}, resulting in a negative r_{ikx}. However, the average of the positive control wells should never be higher than the average of negative control wells, precluding a negative denominator. Vis et al. clip values outside the $[0, 1]$ interval [99]. Note that we describe the relative viability formula given in the code of the gdscIC50 R package, which is, however, not identical to the formula presented in the corresponding paper by Vis et al.(cf. [99]), which seems incorrect.

Curve fitting

After the generation of the relative viabilities, we can plot the relative viabilities against the tested concentrations for each combination of cell line and drug, which results in one dose-response plot per combination (cf. Figure 3.1). Using this plot, we can visually inspect how the drug response varies with increasing concentration and evaluate whether replicate measurements lie close by. Typically, a dose-response curve is fitted to this point cloud to reduce noise (e.g., from replicate measurements) and create inter- and extrapolations for non-investigated concentrations. Moreover, the curve can carry additional information that may help to compare different cell lines or drugs, which we will also describe later in this chapter when discussing dose-response metrics.
Often, it is assumed that the dose-response curve takes a sigmoidal form [99, 100, 101]. In particular, Vis et al. assume they can model the dose-response relationship with a two-parameter sigmoidal function [99]. In their model, the first parameter

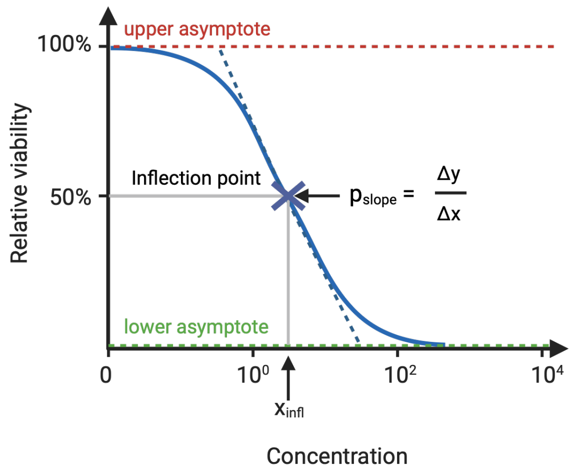


Figure 3.2: Dose response curve. This figure visualizes a sigmoidal dose response curve as employed by Vis et al. [99]. Created with BioRender.com.

x_{infl} denotes the position (concentration) of the inflection point and the second parameter p_{slope} corresponds to the slope of the curve at the inflection point (see Figure 3.2). The upper and lower asymptotes are fixed to 1 and 0, representing 100% and 0% viability, respectively. In total, they use the following two-parameter logistic function that depends on the concentration x

$$f(x) = \frac{1}{1 + e^{-\frac{x - x_{\text{infl}}}{p_{\text{slope}}}}}. \tag{3.4}$$

While they allow x_{infl} to vary for each combination of cell line and drug, they restrict p_{slope} to vary across cell lines only. This means that they expect the drug to affect the position but not the slope of the inflection point. Estimating the parameters is accomplished by reducing the distance of the observed points to the fitted curve by employing the non-linear mixed effects model by Lindstrom and Bates [102]. This model has been developed to fit the needs of repeated measures data (e.g., time series data), where a set of samples is measured at differing conditions, and it is, furthermore, tenable to assume that the responses of the samples follow a similar non-linear functional relationship.
After model fitting, the model fitness should be assessed as a further quality control step. To this end, Vis et al. employ the root-mean-squared error (RMSE)

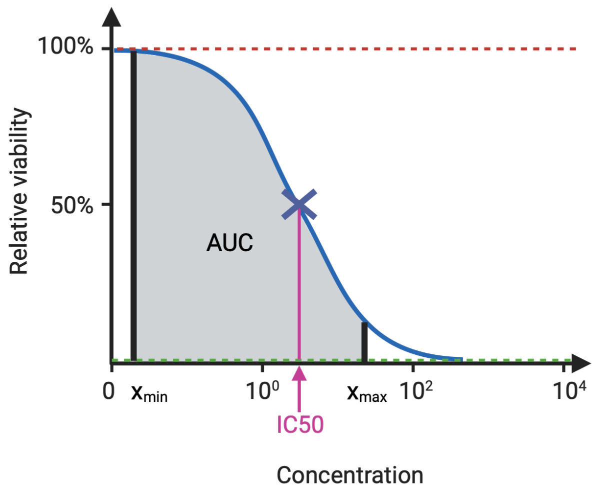


Figure 3.3: Dose response metrics This figure visualizes two popular dose response metrics, i.e., the IC50 value (half maximal inhibitory concentration) and the AUC (area under the viability curve for the tested concentration range). Created with BioRender.com.

between the area under the dose-response measurements and the area under the fitted curve.

Commonly used metrics

In order to compare the effect of different compounds, characteristic values of the curve are employed. These values are calculated using metrics that condense the information of one dose-response curve in one continuous value. In the following, we briefly present the most commonly applied metrics.

IC50. The half maximal inhibitory concentration (IC50) denotes the concentration of a compound needed to achieve 50% relative viability [103]. If we fit the dose-response curve with the upper asymptote fixed to 1 and the lower asymptote fixed to 0, it corresponds to the concentration of the inflection point of the sigmoidal curve (see Figure 3.3). The IC50 value is comparable between cell lines treated with the same compound: a lower IC50 indicates an earlier success of the compound. However, different drugs usually possess different feasible concentration ranges. Thus, IC50 values are generally not comparable between drugs.

AUC. The area under the viability curve (AUC) is typically defined as the area under the viability curve for the tested concentration range [103] (cf. Figure 3.3). As long as the tested concentration range is equal across experiments, it is comparable across cell lines and drugs. In this case, a small AUC corresponds to a strong response and a high AUC to a weak response. While employing the same concentration range across different cell lines for the same compound is relatively common, it is uncommon or even infeasible to test the same concentration range for different compounds since their feasible concentrations may differ by magnitudes. As a partial remedy for compounds with overlapping concentration ranges, Pozdeyev et al. suggested the adjusted AUC, which is calculated solely using the shared concentration range [100]. Vis et al. [99] suggest normalizing the AUC by dividing the original value by the area of the box defined by the four vertices $(x_{min}, 0), (x_{min}, 100), (x_{max}, 0), (x_{max}, 100)$, where x_{min} and x_{max} are the minimal and maximal tested concentrations, respectively. While this enforces an AUC range of $[0, 1]$, these values can still be artificially inflated or deflated if the chosen concentration range covers merely a part of the feasible concentration range.

GR metrics. In 2016, Hafner et al. showed that metrics such as IC50 and AUC, which are defined using relative viability, are affected by one fundamental flaw [101]: they depend on the proliferation rate of the cell lines. When performing an assay with a fixed duration, a cell line with a lower proliferation rate will be reported as having a lower IC50 or AUC than a biologically very similar cell line with a higher proliferation rate. Thus, Hafner et al. propose computing a normalized growth rate inhibition (GR value) instead of the relative viability and defining various metrics using the GR value. To calculate GR values, they either need the initial cell count measured directly prior to drug exposure or the division time of the cell line [101]. Also, note that only if we know the initial cell count can we faithfully distinguish cytostatic (slowing down or stopping tumour growth) from cytotoxic (killing tumour cells) compounds [103]. However, in many studies, neither the initial cell count nor the division time of cell lines is supplied [103]. To the best of our knowledge, the GDSC database does neither provide initial cell counts nor cell division times, so we cannot determine GR values.

3.2 Genomics of Drug Sensitivity in Cancer (GDSC) database

The Genomics of Drug Sensitivity in Cancer (GDSC) database, a joint endeavour by the Wellcome Trust Sanger Institute and the Center for Molecular Therapeutics Massachusetts General Hospital Center, is one of the largest publicly available cancer cell line drug screening panels in terms of screened drugs and cell lines (cf. Table 3.1). Since its inception in 2012 [14], the data set has been continuously maintained and developed until today. Generally, the GDSC database encompasses a plethora of chemotherapeutic and targeted drugs, ranging from approved anti-cancer compounds to putative drugs still in early development. The drug screening data is accompanied by molecular profiles of the cell lines, rendering the GDSC a valuable resource for drug response biomarker discovery and anti-cancer drug sensitivity prediction. The molecular profiling data encompasses mutations, copy number alterations, microsatellite instabilities, gene fusions, methylation information, and gene expression data.
In the following, we limit ourselves to describing the data types we used throughout this thesis, i.e., the drug response, the mutation, the copy number, and the gene expression data. Descriptions for the generation of the other data types can be found in the Supplement ([104]) of a GDSC publication by Iorio et al. [19, 104]. Note that different releases of the GDSC data set exist. These mainly differ in the number of screened cell lines and drugs. Here, we describe the drug response data of the latest release (Release 8.5, October 2023), as well as the mutation, copy number, and gene expression data of the release they were introduced.

3.2.1 Drug response

The GDSC drug response data can be divided into two subsets, i.e., GDSC1 and GDSC2. They differ in the experimental setup that was used to measure cell viability. For GDSC1, which was in development from 2010 to 2015, adherent cell lines (cell lines growing attached to the surface of their vessel) were screened with the Syto60 assay, while a Resazurin assay was employed for suspension cell lines [104] (cell lines growing free-floating). From 2015 onwards, cell viability was measured with the CellTiter-Glo assay instead, resulting in the still actively maintained

GDSC2 data set [98]. After generation of the raw viability values with either technique, the relative viabilities were calculated (cf. Section 3.1.2) and dose-response curves were fitted with the described model by Vis et al. [99]. For each subset (GDSC1/GDSC2), all dose-response curves were fitted simultaneously [98]. The GDSC website provides downloads of the raw and fitted viability data, including logarithmized IC50 and AUC values.

3.2.2 Mutation calling

The cell lines were genetically characterized using the Agilent SureSelectXT Human All Exon 50Mb bait set [105], which is a procedure that can generate information on mutations in the human exome. The procedure captures the exome of a sample by hybridization to a DNA library containing the desired exon sequences. Then, the obtained fragments are sequenced. To identify sequencing variants, i.e., substitutions and small insertions or deletions, CaVEMan [106] and Pindel [107] were applied [104]. Lastly, putative sequencing artefacts and germ line variants were removed as described in the Supplement of Iorio et al. [104]. The resulting data has been deposited on the COSMIC website [108, 104], the version that we used (Release v71) can be downloaded from the GDSC website [109] and contains mutations in 19100 genes.

3.2.3 Copy number estimation

In addition to relatively small-scale mutations in the genome, losses and gains of whole genomic regions were also quantified. To this end, the Affymetrix SNP6.0 Array was used [105]. Generally, this microarray encompasses oligonucleotides that interrogate interesting positions in the genome, e.g., SNP or CNV-associated positions [110]. The genomic sample of interest can be characterized by hybridization of the fragmented and labelled sample to the complementary sequences of the microarray [110]. In the GDSC database, the raw data from the microarray experiment was processed with the PICNIC algorithm [111], a specialized tool for absolute copy number estimation from microarray data in cancer cells [112]. These absolute copy numbers were then employed to calculate losses and gains of genomic regions [111].

Given the estimated average genome ploidy of a cell line, loss and gain were defined as [111]

$$\text{gain} = \begin{cases} \text{avg. genome ploidy} \leq 2.7 \wedge \text{total copy number} \geq 5 \\ \text{avg. genome ploidy} > 2.7 \wedge \text{total copy number} \geq 9 \end{cases}$$

and

$$\text{loss} = \begin{cases} \text{avg. genome ploidy} \leq 2.7 \wedge \text{total copy number} = 0 \\ \text{avg. genome ploidy} > 2.7 \wedge \text{total copy number} < (\text{avg. genome ploidy} - 2.7) \end{cases}$$

The corresponding file can be downloaded from COSMIC and, amongst others, contains information on the start and end position of the copy number variant as well as the gene name (17618 genes). Note that we could not find further information on the presented definition of copy number loss and copy number gain. However, it seems that this definition was motivated by a finding of van Loo et al. [113] that the average genome ploidy of breast carcinoma is larger than 2.7.

3.2.4 Gene expression

The cancer cell lines were also profiled regarding their gene expression. The data was obtained with the Affymetrix Human Genome U219 array [105]. The generated raw values were normalized with the robust multi-array analysis (RMA) algorithm by Irizarry et al. [114], resulting in gene expression values for 18562 loci as defined in the chip definition file (BrainArray v.10) by Dai et al. [115].

4 Machine learning

Just like the conception of personalized medicine can be traced back to the ancient Greek philosopher Hippocrates [1], the prospect of machines with human-like *intelligent* behaviour has already fascinated and inspired thinkers and writers many centuries ago - also giving rise to the science fiction literature genre, in which such futuristic machines commonly occur [116].
However, the birth date of the academic discipline surrounding the quest of designing intelligent machines, i.e., *artificial intelligence* (AI), is located somewhere in the 1950s when Alan Turing dedicated himself to the topic as well [117]. At the same time, *machine learning* (ML) evolved as a subdomain of artificial intelligence concerned with developing algorithms and software for learning from data. While it is widely accepted that the term machine learning had been coined or at least been popularized by the IBM employee Arthur Samuel in the 1950s [118], early works on simple artificial neural networks - which should later become a key technique of ML - had already been published in 1943 by McCulloch and Pitts [119, 116, 120]. The so-called McCulloch-Pitts neuron was a mathematical abstraction of the basic functionality of a biological neuron in the brain and could represent Boolean functions, i.e., could accept Boolean inputs and produce a Boolean output. In 1957, Rosenblatt introduced the perceptron model, a more generalized version of the McCulloch-Pitts neuron, often regarded as the first artificial neural network [121, 122]. Over the last few decades, the public interest in AI and ML fluctuated while a variety of novel ML algorithms were devised and refined to fit the needs of the available data and the users. However, over the course of the previous years, a growing interest in AI and ML developed. After decades of technological advances in computer science and engineering, it seems the time has finally come for their widespread use. When OpenAI made their chatbot ChatGPT - a large language model based on artificial neural networks - publicly available in November 2022, the public interest in AI and ML reached its peak (cf. Figure 4.1). Indeed, it seems that almost every academic discipline and business alike currently integrates AI and ML algorithms into their workflows.
Historically speaking, ML and neighbouring scientific fields, e.g., optimization, statistics, data mining and pattern recognition, have been integral to bioinformatics

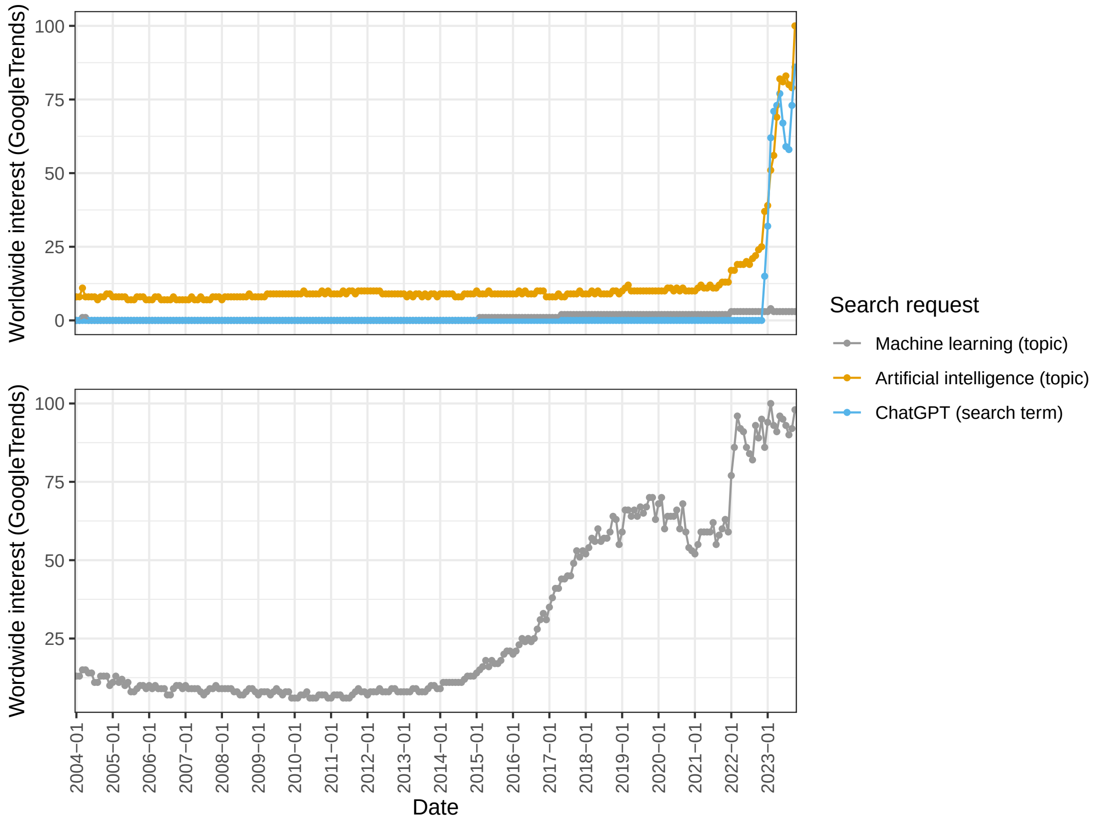


Figure 4.1: GoogleTrends for AI and ML. This figure depicts the relative worldwide interest in artificial intelligence (AI), machine learning (ML), and the chatbot ChatGPT over almost the complete last two decades as reported and measured by GoogleTrends. The data was retrieved from the GoogleTrends website on 10th of October, 2023. The upper plot shows the comparison between the three terms, where it can be clearly observed that ML draws the least attention from the public. Since we could not visibly inspect the development of the interest in ML in this plot, the lower plot only depicts the relative interest in ML alone.

since its birth. Especially research on large cancer cell line panels has literally been shaped by the use of ML techniques.

In the following, we first provide a brief yet comprehensive overview of the ML landscape of today. Then, we will delve into the aspects most relevant to this work. To this end, we present the key algorithms on which our work builds.

Authors' contributions
While this chapter is not strictly based on the publications presented in this thesis, it contains descriptions of algorithms that were also part of them. In particular, the random forest description and the minimum-redundancy maximum-relevance algorithm description in this chapter have been adopted from the SAURON-RF publication [29], and the text on reliability estimation rests on the information presented in the follow-up work *reliable SAURON-RF* [33]. I drafted both manuscripts. Moreover, I drafted the review paper *Trust me if you can: a survey on reliability and interpretability of machine learning approaches for drug sensitivity prediction in cancer* during the genesis of this chapter. Thus, the text of this chapter overlaps with information in the corresponding manuscript.

4.1 The different realms of machine learning

Clearly, machine learning is a multidisciplinary research area, sharing its methodology with various other disciplines such as optimization, statistics, and information theory. Yet, what all ML methods have in common is that they are employed for learning from data. Tom Mitchell has given a central definition of what constitutes learning in the ML context:

> *'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.'* (Tom Mitchell, 1997 [123])

Arguably, the currently existing body of ML methods can be divided into four major realms, each tailored to fulfilling specific tasks: supervised, unsupervised, semi-supervised, and reinforcement learning [124]. In the following, we will briefly contrast the ideas and concepts underlying these four realms and highlight their application cases within anti-cancer drug sensitivity prediction.

4.1.1 Supervised learning

In supervised learning, we are given a set of observed pairs $\{(x_1, y_1), (x_2, y_2), ...(x_N, y_N)\}$, where $x_i \in \mathscr{X}$ is called the feature vector and $y_i \in \mathscr{Y}$ the response, and we are interested in finding a mapping $\hat{f} : \mathscr{X} \to \mathscr{Y}$ that best approximates the assumed, yet unknown functional relationship f between the features and the response [125, 126]. A standard assumption of many supervised learning algorithms is that these pairs are drawn iid from a distribution with range $\mathscr{X} \times \mathscr{Y}$ [125]. The term iid refers to the set of random variables of a probability distribution and means that these random variables are (1) sharing the same probability distribution (identical) and (2) not influencing each other in any way (independent). However, especially the latter assumption is frequently violated by real-world (biological) data.
Usually, we employ a matrix-vector notation to represent the observed pairs. Here, $\mathbf{X}$ is an $N \times P$-matrix ($N \in \mathbb{N}, P \in \mathbb{N}$) with each row corresponding to one x_i, where x_i is a row vector containing values of P features. The vector $\mathbf{y}$ is the accordingly ordered N-dimensional response vector with y_i as entries. Depending on whether the output variable is quantitative (continuous) or qualitative (discrete), the supervised learning task is called regression or classification, respectively [126]. Note that for convenience, we often denote a discrete response vector with $\mathbf{d}$ instead of $\mathbf{y}$ throughout this work. Furthermore, note that for simplicity, we follow the notation in the book *Semi-supervised learning* [125] and assume that all distributions have densities for the following description. Instead of directly learning a function, many supervised ML algorithms are probabilistic models in the sense that they estimate the conditional density $p(y|x)$. A so-called discriminative ML model directly models this conditional density $p(y|x)$, whereas a generative one models the joint density $p(x, y) = p(x|y) \cdot p(y)$, which can be plugged into Bayes' theorem to determine the desired conditional density: $p(y|x) = \frac{p(x|y) \cdot p(y)}{\int_z p(x|z) \cdot p(z) dz}$ [125].
Clearly, the straightforward ML modeling of drug response prediction is via supervised learning. Indeed, most publications fall within this realm (cf. Table 4.1, p. 50). Here, the features are results from multi-omics measurements, e.g., gene expression values, and the response is the observed drug sensitivity from a drug screening assay reported in the form of some summary metric such as IC50 (cf. Chapter 3). Consequently, each row of the model matrix $\mathbf{X}$ then contains the molecular characterization of one cancer cell line, and each entry of the response vector $\mathbf{y}$ is the measured sensitivity value. While drug response prediction is inherently a regres-

sion task, it can be formulated as a classification task by discretising the continuous sensitivity value.

4.1.2 Unsupervised learning

Unsupervised learning can be interpreted as the task of finding interesting structures in data without a specific variable that guides or supervises the model [125]. Mathematically, this can be described as a situation where we have the model matrix $\mathbf{X}$, generated by drawing samples iid from a distribution with range $\mathscr{X}$, but no designated response vector $\mathbf{y}$. From a probabilistic view, we might thus aim to estimate the density function $p(x)$. In low-dimensional spaces, this can already be achieved relatively well, while high-dimensionality seems to necessitate the usage of simpler approaches more loosely learning the structure of the data [127].
The drug response prediction task, as such, is usually not modeled as an unsupervised learning task. Yet, there still exists a plethora of functions within this task that can be fulfilled by unsupervised learning methods: Since data obtained from high-throughput multi-omics measurements suffers from the curse of dimensionality ($P \gg N$), unsupervised learning algorithms such as principal component analysis are frequently employed for reducing the dimensionality of the model matrix. Clustering algorithms, e.g., k-medoid clustering, can help divide the samples into groups when a group association is unknown beforehand. Furthermore, some techniques can be repurposed for supervised learning, e.g., association rule mining can be reformulated as a supervised learning task by enforcing the rules to adopt a specific form.

4.1.3 Semi-supervised learning

Classical semi-supervised learning lies conceptually between supervised and unsupervised learning: the model matrix $\mathbf{X}$ can be divided into two parts, one sub-matrix $\mathbf{X}^r$ for which an associated response vector $\mathbf{y}^r$ is available, and a second sub-matrix $\mathbf{X}^w$ without an associated response. While we can interpret this setting as a supervised learning application and, thus, simply train a supervised model on the former data set and then apply the resulting predictor to the latter one, we can also argue that the unlabeled data might provide additional information for improving our

model. This means we require $p(x)$ to improve the estimation of $p(y|x)$. Different ideas and associated assumptions exist on how $p(x)$ interacts with $p(y|x)$, which inspired the development of different algorithms. Three very popular assumptions, i.e., the smoothness assumption, the low density assumption, and the cluster assumption, have a common principle [125], and may even be interpreted as different phrasing for the same principle that can roughly be summarized as follows: if two points x_i and x_j from a high-density region lie close by, they should not be separated by a decision boundary, i.e., their class labels y_i and y_j should be equal. If such an assumption holds, then unlabeled points can clearly help in locating decision boundaries by providing information on the density of a region. Typically, semi-supervised methods can be divided into two categories, inductive and transductive methods [125, 128]. Inductive semi-supervised methods can be considered a direct extension of supervised learning and yield a predictive function $\hat{f} : \mathscr{X} \rightarrow \mathscr{Y}$ as output, whereas transductive semi-supervised methods only generate predictions for the samples without response [128].
Currently, we are only aware of one method that employs semi-supervised learning for drug response prediction: a deep generative neural network approach (variational autoencoder) by Rampášek et al. [129].

4.1.4 Reinforcement learning

In contrast to all previously presented ML realms, reinforcement learning is concerned with dynamically learning from interaction instead of learning from a given set of static data points [123]. This concept can be described as follows. Let there be a so-called agent that can perform all possible actions from a set of actions A. Moreover, let there be an environment that can be in a set of states S. Each time an agent performs an action, the environment returns a reward or penalty and changes its state [130]. Clearly, the agent should pursue the goal of maximizing its reward. In that sense, reinforcement learning is most similar to supervised learning since the environment provides some form of supervision. However, there are differences between supervised learning and reinforcement learning and specific challenges rather unique to reinforcement learning. For example, since there is no immediate access to some static mapping of features to responses, the agent must explore its environment by taking actions (trial and error search). The action an agent takes at a specific

point in time may not only influence the direct reward but also all future rewards (delayed reward principle). Mathematically speaking, reinforcement learning can be formulated as an incompletely known Markov decision process [130], the details of which are out of the scope of this thesis.
In drug response prediction, this form of modeling has only rarely been investigated at the time this thesis was written (cf. Table 4.1, p. 50). However, we could interpret the task of drug prioritization in terms of reinforcement learning. The model by Liu et al. [131], for example, assumes that an agent must construct the correct drug ranking. At each time step, the model opts for one specific drug and obtains a reward from the environment based on the accuracy of the choice. We think reinforcement learning can be a rewarding avenue for drug response prediction and discuss its potential in the last chapter of this thesis (cf. Chapter 9).

4.2 Requirements for machine learning systems in practice

In the previous sections, we gave a brief overview of the machine learning landscape of today and highlighted how the diverse set of available techniques may be applied to anti-cancer drug response prediction. The current state of the art and this thesis mainly focus on supervised learning combined with some unsupervised techniques to achieve progress in drug response prediction. Therefore, the remainder of this chapter mainly revolves around considerations that play a role in these techniques. However, many of the principles can be straightforwardly translated into semi-supervised and reinforcement learning. In the ensuing section, we mainly deal with requirements that ML systems should comply with to become trustworthy.

4.2.1 Trustworthiness

The term trustworthiness does not refer to a single mathematical property but rather a continuously extensible collection of such that together should guarantee lawful, ethical, and robust use of developed machine learning systems [132]. The European Commission recently released guidelines on how to implement trustworthy AI systems in practice [132]. Compliance with these guidelines should ensure that humans interacting with the system receive maximal benefit while any harm should be prevented. For example, the data should be securely stored and protected, the decisions of the system should be reliable, reproducible and traceable, and the system should not be intentionally discriminatory against specific societal groups. Moreover, the user should be made aware of the fact that an interaction with an AI-based system takes place and learn to understand the risks associated with this circumstance. Especially for a high-stake application case such as medical decision support systems, we need to demand for complying with the highest possible standards in all mentioned aspects.

4.2.2 Statistical performance

Clearly, an ML system is only helpful if its output is correct, i.e., if it learned to perform a specific task. In supervised learning, we explicitly optimize for correctness by finding a function $\hat{f} : \mathscr{X} \to \mathscr{Y}$ that best approximates the assumed relationship between the $(N \times P)$-dimensional sample matrix $\mathbf{X}$ and the N-dimensional response vector $\mathbf{y}$. Indeed, we already employ performance measures in the model training phase to minimize false predictions. In supervised learning, performance can be measured by comparing the predictions $\hat{\mathbf{y}}$ from the trained model to the known response $\mathbf{y}$. If $\mathbf{y}$ is continuous, measures such as the mean-squared error (MSE) or mean absolute error (MAE) are routinely employed. The co-domain of both measures is $[0, \infty)$. Both evaluate to their minimal value 0 iff the model is perfectly representing $\mathbf{y}$. The larger the value becomes, the worse the model approximates the response. The MSE is defined as

$$\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y_i})^2 \tag{4.1}$$

and the MAE can be determined analogously by replacing the square deviation with the absolute deviation. If one is more generally interested in finding out whether the trained model correctly captured a specific trend, correlation measures such as Pearson correlation [133, 134] and Spearman correlation [135] can be employed instead. Compared to MSE and MAE, these correlation coefficients have the advantage that they evaluate to the closed co-domain $[-1, 1]$ rendering their outputs more comparable per se. Pearson correlation measures the linear dependency between two variables. It is 0 iff no linear dependency between two variables can be detected. If $\mathbf{y}$ and $\hat{\mathbf{y}}$ tend to linearly increase simultaneously, their Pearson correlation will be positive. If either increases while the other decreases, their Pearson correlation coefficient will be negative. The magnitude of the value indicates the strength of the correlation. Let $\bar{y}$ and $\bar{\hat{y}}$ be the empirically determined mean of $\mathbf{y}$ and $\hat{\mathbf{y}}$, respectively. Then, the Pearson correlation coefficient (PCC) can be calculated as

$$\frac{\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2 \sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}}. \tag{4.2}$$

The PCC is only able to measure linear dependency between two variables. However, variables might be non-linearly dependent. The Spearman correlation coefficient (SCC) can capture a monotonic relationship between two variables. The SCC is defined as the PCC between the ranks instead of the values of the two variables. Analogous to the PCC, the SCC is 0 iff there is no monotonic relationship between the two variables. If they tend to increase concurrently, the sign of the SCC is positive. If either of them increases while the other decreases, it is negative. Once again, the magnitude of the value indicates the strength of the correlation.
If we deal with a discrete response variable $\mathbf{d}$ instead, one typically generates a confusion matrix to assess the correctness of a classifier [136]. A confusion matrix can be defined as follows: Let k be the number of different classes the discrete response can potentially assume. The confusion matrix is a $k \times k$-matrix whose rows correspond to the predicted classes and whose columns correspond to the observed classes. Each entry of this confusion matrix contains the co-occurrence frequency of one predicted class and one observed class such that the sum in each row is equal to the number of instances predicted to be of one particular class, and the sum of each column is equal to the number of instances actually within a class. In Figure 4.2, we depict a confusion matrix for a binary classifier distinguishing between a positive and negative class, which may, for example, correspond to sensitive and resistant cell lines in drug sensitivity prediction, respectively. Here, sensitive cell lines predicted to be sensitive are called true positives (TP), sensitive cell lines predicted to be

resistant are called false negatives (FN), resistant cell lines predicted to be resistant are called true negatives (TN), and resistant cell lines predicted to be sensitive are called false positives (FP). The sum of TP and FN is equal to all positives (P), the sum of TN and FP is equal to all negatives (N), the sum of TP and FP is equal to all predicted positives (PP), and the sum of TN and FN is equal to all predicted negatives (PN).

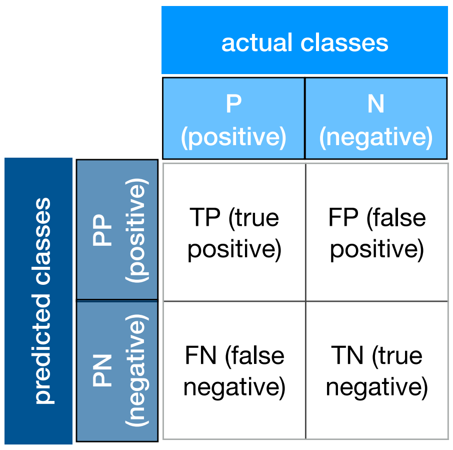


Figure 4.2: Binary confusion matrix. In this figure, we exemplary show a confusion matrix for a binary classification task. Each row corresponds to a predicted class, and each column to an actual class. The investigated classes are called positive and negative, and may, for example, correspond to drug-sensitive cell lines (positive) and drug-resistant cell lines (negative) for drug sensitivity prediction.

Such a confusion matrix serves as the basis for defining a plethora of metrics for evaluating classifier performance. In the following, we restrict ourselves to defining metrics for binary classification since this is the main focus of this work. Arguably, the most simple metric we could define is the accuracy given as the proportion of correct predictions

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}. \tag{4.3}$$

It suffers from one major drawback: if the two classes are highly imbalanced, the result of an accuracy computation can be misleading. Suppose a classifier simply returns the value of the majority class for each possible input sample. In that case, the accuracy can still be very high despite the classifier having no discriminatory ability at all. As a remedy, we can consider the sensitivity and specificity of a classifier. The sensitivity of a classifier is the ability of the classifier to correctly identify positive cases out of all positive cases

$$\text{sensitivity} = \frac{\text{TP}}{\text{P}}. \tag{4.4}$$

Analogously, the specificity is the ability of the classifier to correctly identify negative cases out of all negative cases

$$\text{specificity} = \frac{\text{TN}}{\text{N}}. \tag{4.5}$$

If a classifier simply returns the value of the majority class, either sensitivity or specificity evaluates to 0. If one aims to evaluate a single, most informative metric, Chicco and Jurman [137] argue that Matthew's correlation coefficient (MCC) [138] should be employed since it considers all four entries of the binary confusion matrix and does mathematically not distinguish between what is defined as positive or negative class. The MCC can be calculated as

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP}+\text{FP}) \cdot (\text{TP}+\text{FN}) \cdot (\text{TN}+\text{FP}) \cdot (\text{TN}+\text{FN})}} \tag{4.6}$$

4.2.3 Reliability and (un)certainty

When we address the question of whether we have trust in a prediction, performance evaluation can point towards a certain direction since we expect to observe a similar performance on new samples. However, performance estimates typically tell us nothing about the degree of trust that we have in a prediction for a single, previously unseen instance during model deployment, which is what is often referred to as reliability in the ML literature [139, 140, 141]. One possibility to achieve reliability is via uncertainty quantification [142, 143]. Here, one is ultimately interested in estimating predictive uncertainty, i.e., the uncertainty about the prediction for a specific instance [144, 142]. Thereby, an ML model should be enabled to abstain from casting a prediction for a new instance if it is uncertain, rendering the corresponding model reliable. Note that this proceeding also mimics how medical doctors act when deciding to seek a second opinion before giving medical advice.

A prediction for a new instance constitutes the end of an ML pipeline. Thus, the predictive uncertainty consists of all uncertainties from data generation to the deployment of the trained ML model. Currently, the machine learning literature distinguishes between at least two different forms of uncertainty: aleatoric uncertainty and epistemic uncertainty [144, 143]. Aleatoric uncertainty is the data-inherent uncertainty that stems from randomness in the experiment generating the measurement, e.g., noise [144]. Indeed, we must acknowledge that all biological experiments harbour this source of uncertainty. For drug sensitivity prediction, e.g., we may observe aleatoric uncertainty in the response: the drug response distributions of the sensitive and resistant cell lines may overlap, making any prediction of a classifier in the overlapping region aleatorically uncertain. Aleatoric uncertainty is called irreducible, i.e., it cannot be removed by gathering more data, i.e., more samples. On the contrary, epistemic uncertainty is reducible and refers to the uncertainty arising from a lack of knowledge [144]. Thus, it can potentially be resolved by gathering more samples. Hülllermeier and Waegeman [144] partition epistemic uncertainty into two further types of uncertainty, namely model uncertainty and approximation uncertainty. The former refers to the uncertainty about which model to choose, while the latter denotes the uncertainty in the estimation of the model parameters. Thus, it is also referred to as estimation or parametric uncertainty in the literature instead [143]. There is a direct link between this uncertainty definition and the classical bias-variance tradeoff from statistics that we will briefly revisit in Section 4.3.1. While it seems straightforward to define aleatoric and epistemic uncertainty as presented above, in fact, these definitions are to some extent ambiguous (cf. Hüllermeier and Waegeman [144] and Gruber et al. [143] for thorough discussions) and Gruber et al. argue that they are too simple to cover the full range of uncertainties arising in real-world applications [143], e.g., uncertainties during data collection or model deployment cannot be captured.

We evaluated whether the current landscape of drug sensitivity approaches considers reliability for model development. The results of this assessment are presented in Table 4.1, p. 50. In Chapter 8, we will look at reliability estimation in more detail.

4.2.4 Interpretability and explainability

Up to this point, we have only discussed trust in ML from the perspective of model correctness. Arguably, trust can, however, also be generated by rendering models understandable to humans interacting with the system, which is what is commonly referred to as interpretability [145, 146, 147]. This definition of interpretability may seem trivial at first sight since we all have some intuition and preconceptions about what should constitute interpretability. However, from an ML perspective, there exists a plethora of possibilities to apply this definition and yet no universally agreed approach to achieve or merely assess it [148, 149]. In this thesis, we provide a taxonomy of interpretability in ML (cf. Figure 4.3), which we believe to represent the vast majority of methods within the field of (supervised) ML more generally and drug sensitivity prediction in particular. Note that a specific ML system can possess characteristics from all categories in the last layer of the taxonomy. We derived this taxonomy mainly based on the works by Lipton [150], Biran and Cotton [147], and Imrie, Davis, and van der Schaar [146].

Generally speaking, interpretability can be model-inherent or generated by post hoc explanations. The former concept is often referred to as (model) transparency, while the latter is known as explainability. The source of the transparency can vary. A model can be transparent because of

- its simplicity either in terms of size or computation time needed for inference. Lipton calls this form of model transparency simulatability [150]. Simulatability refers to the simplicity of the entire model.
- the comprehensibility of its components (inputs, parameters, calculation rules, outputs), called intelligibility by Lou et al. [151] and decomposability by Lipton [150]. Decomposability refers to the simplicity of the individual model components.
- the transparency of the learning algorithm defined by the understandability of the solution space in terms of convergence criteria or uniqueness of the

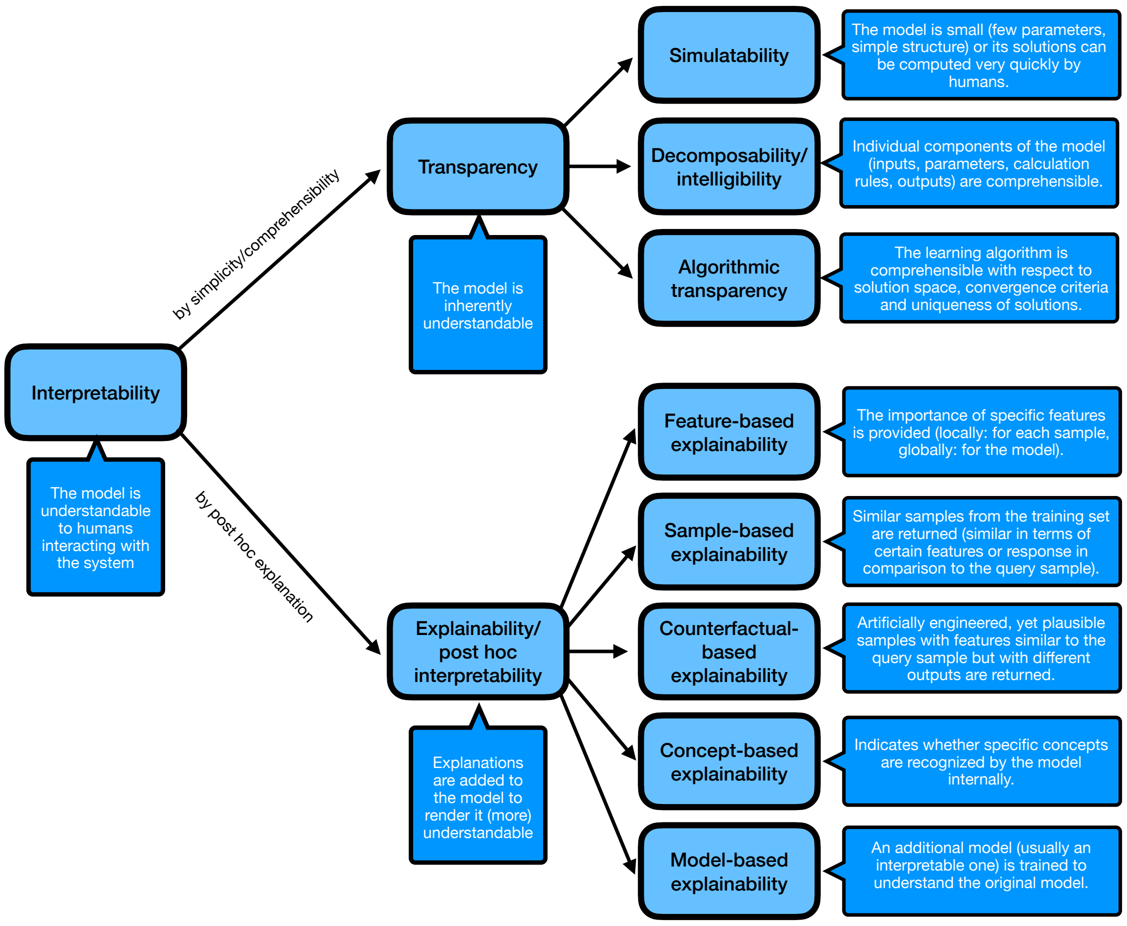

Interpretability
The model is understandable to humans interacting with the system
by simplicity/comprehensibility
by post hoc explanation
Transparency
The model is inherently understandable
Simulatability
The model is small (few parameters, simple structure) or its solutions can be computed very quickly by humans.
Decomposability/ intelligibility
Individual components of the model (inputs, parameters, calculation rules, outputs) are comprehensible.
Algorithmic transparency
The learning algorithm is comprehensible with respect to solution space, convergence criteria and uniqueness of solutions.
Explainability/ post hoc interpretability
Explanations are added to the model to render it (more) understandable
Feature-based explainability
The importance of specific features is provided (locally: for each sample, globally: for the model).
Sample-based explainability
Similar samples from the training set are returned (similar in terms of certain features or response in comparison to the query sample).
Counterfactual-based explainability
Artificially engineered, yet plausible samples with features similar to the query sample but with different outputs are returned.
Concept-based explainability
Indicates whether specific concepts are recognized by the model internally.
Model-based explainability
An additional model (usually an interpretable one) is trained to understand the original model.

Figure 4.3: Taxonomy of interpretability in ML. This figure depicts a taxonomy of interpretability in ML that we derived from works by Lipton [150], Biran and Cotton [147], and Imrie, Davis, and van der Schaar [146]. The bright blue speech bubbles provide brief, intuitive descriptions of the technical terms. Note that a specific ML model may possess properties from all categories of the last layer.

solution. Lipton calls this form of transparency algorithmic transparency, and it refers to the comprehensiveness of the training algorithm and its solutions.

To maximize trust in a machine learning model, it might be tempting to demand for interpretability in this strict sense. However, like Lipton [150], we believe that one must then also ask whether a human can actually be trusted. Until today, we cannot understand how the human brain (or the complete human organism) makes decisions. We only know the explanations and justifications that humans provide. Indeed, humans are like black box ML models augmented with post-hoc explanations instead.

In ML, post-hoc explanations are typically generated if model-inherent interpretability is absent. Nevertheless, it might be useful to augment inherently interpretable (transparent) models with post hoc explanations as well. Imrie, Davis, and van der Schaar [146] divide post hoc explanations into five categories

- feature-based explanations that measure the importance of features, either locally for specific samples or globally across the model.
- example-based explanations that return those samples from the training set most similar to the input sample, giving users the opportunity to apply their domain knowledge.
- counterfactual-based explanations that return artificially engineered yet plausible samples resulting in different outputs than originally obtained for a specific sample. In particular, the idea is to (slightly modify) the input values of as few features as possible to create a different model output.
- model-based explanations that return a second, more interpretable (transparent) model trained on the original model.
- concept-based explanations that indicate whether specific concepts (patterns) are processed or utilized in the model. Typically, this is accomplished by examining whether internal representations of samples with and without the concept differ.

In Table 4.1, we compare various drug sensitivity prediction approaches concerning their interpretability using the suggested taxonomy.

4.2.5 Other requirements

In this thesis, we focus on the above-mentioned desiderata of ML methods in the context of drug sensitivity prediction and prioritization in cancer. However, for a successful deployment of a complete ML system in practice, we have to consider a variety of other equally important factors as well. These include but are not limited to

- privacy mechanisms that protect the utilized data,

Table 4.1: Placement of drug sensitivity prediction approaches in the ML universe. This table visualizes ML approaches pursued for drug sensitivity prediction over the last decade. Supervised learning (white: regression, light blue: simultaneous regression and classification, steel blue: classification) was the preferred research direction, and only few approaches were dedicated to semi-supervised learning (light yellow) or reinforcement learning (light orange). We also assessed the trustworthiness of the approaches in terms of reliability and interpretability.

		Interpretability								Reliability
		Transparency			Explainability					
		Simulatability	Decomposability	Algorithmic transparency	Feature	Sample	Counterfactual	Concept	Model	
Supervised	Menden et al. (2013) [152]	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Zhang et al. (2015) [153]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	SRMF (2017) [154]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	HARF (2017) [155]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	Matlock (2018) [156]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	KRL (2018) [157]	✗	✗	✗	✗	✗	✗	✗	✗	✗
	RWEN (2018) [158]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	CDRscan (2018) [159]	✗	✗	✗	✗	✗	✗	✗	✗	✗
	QRF (2018) [31]	✗	✓	✗	✓	✗	✗	✗	✗	✓
	NCFGER (2018) [160]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	DeepDR (2019) [161]	✗	✗	✗	✗	✗	✗	✗	✗	✗
	netBITE (2019) [162]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	Deng et al. (2020) [163]	✗	✓	✗	✗	✗	✓	✗	✗	✗
	PathDSP (2021) [164]	✗	✗	✗	✓	✗	✗	✗	✗	✗
	GraphDRP (2021) [165]	✗	✓	✗	✓	✗	✗	✗	✗	✗
	SAURON-RF (2022) [29]	✗	✓	✗	✓	✗	✗	✗	✗	✗
	reliable SAURON-RF (2023) [33]	✗	✓	✗	(✓)	✗	✗	✗	✗	✓
	LOBICO (2016) [27]	✓	✓	✓	✓	✗	✗	✗	✗	✗
	Stanfield (2017) [166]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	HNMDRP (2018) [167]	✗	✓	✗	✗	✗	✗	✗	✗	✗
	Deep-Resp-Forest (2019) [168]	✗	✗	✗	✗	✗	✗	✗	✗	✗
	MERIDA (2021) [21]	✓	✓	✓	✗	✗	✗	✗	✗	✗
Semi	Dr.VAE (2019) [129]	✗	✗	✗	✗	✗	✗	✗	✗	✗
Reinf.	PPORank (2022) [131]	✗	✗	✗	✗	✗	✗	✗	✗	✗

- security mechanisms that protect against external threats and intentional misuse of the system,
- safety mechanisms that protect against accidental misuse of the system
- bias-awareness and fairness of an ML system that protect against harm generated by the use of data considered sensitive, e.g., ethnicity or gender.

We refer to Qayyum et al. [169] for further discussions about these topics.

4.3 Supervised learning techniques

In Section 4.1, we have already briefly introduced the concept of supervised machine learning. In this section, we will look at supervised ML in more depth and provide descriptions of algorithms used or extended as part of this thesis.
Recall that in supervised learning, we are given a set of observed pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ as realizations of random variables $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ drawn iid from a distribution with range $\mathscr{X} \times \mathscr{Y}$, where $x_i \in \mathscr{X}$ is called the feature vector and $y_i \in \mathscr{Y}$ the response. We assume that there is a true functional relationship $f : \mathscr{X} \to \mathscr{Y}$ that we can approximate using ML techniques. In particular, we can express this functional relationship as

$$Y = f(X) + \epsilon. \tag{4.7}$$

Here, ϵ is a random error term that typically represents measurement noise. Let $\hat{f} : \mathscr{X} \to \mathscr{Y}$ denote the trained ML model approximating f, then $\hat{f}$ induces the prediction $\hat{Y}$ that we can compare to Y to measure the goodness of our fit. Indeed, training an ML model involves minimizing the difference between the prediction and the observed values. Mathematically speaking, if we hypothesize that a specific model type (hypothesis space) $\mathscr{H}$ containing mappings $h : \mathscr{X} \to \mathscr{Y}$ might be a good fit for our problem, we obtain $\hat{f}$ by minimizing over some loss function $l : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$ within an empirical risk function $R_{\text{emp}}(h)$ [144], i.e.,

$$R_{\text{emp}}(h) = \frac{1}{N}\sum_{i=1}^{N} l(h(x_i), y_i) \tag{4.8}$$

and

$$\hat{f} = \text{argmin}_{h \in \mathscr{H}} R_{\text{emp}}(h). \tag{4.9}$$

By following this approach, our prediction carries a variety of uncertainties (cf. Section 4.4): the error term ϵ is related to the aleatoric, i.e., stochastic uncertainty, the choice of the hypothesis space to the model uncertainty, and the difference between the best model from the hypothesis space and the model that we obtained through empirical risk minimization to the estimation uncertainty.

4.3.1 Training a machine learning model

Suppose that we trained our ML method on our training observations $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ and obtained the trained model $\hat{f}$. By design, $\hat{f}(x_i)$ is close to y_i for all $i \in \{1, \ldots, N\}$ and a performance measure used to derive the model, e.g. the MSE (cf. Section 4.2.2), should have a small value. However, we are not actually interested in predicting values for our training data set since we already know the responses. Rather, we want to select the model that generalizes well to unseen samples. To put it differently, given unseen test observations, we want to identify the model with the lowest test error as opposed to the lowest training error. To understand which properties of our method play a role in selecting a model that minimizes the test MSE for an unseen observation x, we can exploit the bias-variance decomposition of the expected test MSE [126]

$$E[(y - \hat{f}(x))^2] = \text{Var}(\hat{f}(x)) + [\text{Bias}(\hat{f}(x))]^2 + \text{Var}(\epsilon). \tag{4.10}$$

From Equation 4.10, we learn that we have to select a model with low bias and low variance, and, since both the squared bias and the variance are nonnegative

quantities, we can never obtain a model below the variance of the error ϵ. Indeed, we can now draw analogies to our uncertainty definitions: $\mathrm{Var}(\epsilon)$ corresponds to the aleatoric uncertainty, $\mathrm{Var}(\hat{f}(x))$ is the amount by which $\hat{f}$ would change when trained on a different training data set and, consequently, corresponds to the estimation uncertainty, and $[\mathrm{Bias}(\hat{f}(x))]^2$ corresponds to the amount of error introduced by choosing a hypothesis space $\mathscr{H}$ for modeling, i.e., corresponds to the model uncertainty. Generally speaking, the higher the flexibility of a method, the lower its bias. For example, (deep) neural networks make very few assumptions about the underlying function to be approximated and have a very low bias. Yet, the variance increases simultaneously with the flexibility of a method given a fixed training data set and overtraining can occur. Hence, the estimation uncertainty for neural networks is usually rather high but could potentially be resolved by gathering more data. On the contrary, simpler methods will often - but do not have to - result in biased models (undertraining) of real-world data. Yet, they exhibit less variance.
When training an ML model, we should keep this so-called bias-variance tradeoff in mind. However, as long as we have no access to unseen test data, we cannot observe it in practice. To approximate the generalization error, we typically have to employ resampling strategies such as k-fold cross-validation (CV)[126]. Briefly, to perform k-fold CV, we randomly split the complete training data set into k non-overlapping groups, called folds. Each fold $i \in \{1, \dots, k\}$ is used as a test set once, while the remainder of the data serves as the training set for our method. We can then use the average error of the folds as an approximation of the generalization error:

$$\frac{1}{k}\sum_{i=1}^{k} \mathrm{MSE}_i. \tag{4.11}$$

4.3.2 Considerations for drug response data

In Chapter 3, we introduced the data set we use to train and test our models. In the following, we will briefly explain some specific issues of this data set and ML techniques to resolve them. Specifically, we discuss the high dimensionality of the feature space and the techniques we employed to counteract it. Additionally, we describe a clustering algorithm that we used to discretize the inherently continuous response.

Dimensionality reduction

Our cancer cell line data is extremely high dimensional ($P \gg N$), a circumstance that inevitably leads to the curse of dimensionality, i.e., the space in which our samples live is not sufficiently covered to draw conclusions on its structure [170, 171]. Consequently, reducing the dimensionality before (or during) training an ML model is advisable. To tie up with our previous discussion on the bias-variance tradeoff, by reducing the dimensionality, we also reduce the variance of our putative model and minimize the risk of overfitting while we increase its bias. The specific level of reduction in variance and simultaneous increase in bias depends on the employed dimensionality reduction (DR) technique.

Generally speaking, dimensionality reduction techniques can be divided into two sets of approaches: feature selection (FS) and feature extraction (FE) [172, 173]. While FS identifies a relevant subset of variables, FE generates a (smaller) set of new ones obtained by some transformation of the original ones. Since FS approaches preserve the structure of the feature space, they are usually considered more interpretable than FE methods. However, FE methods often incur less bias than FS approaches. For FS, three main groups can be distinguished: filter, wrapper and embedded methods [174]. These three groups differ in how the FS interacts with the ML model training. Briefly summarized, filter methods are applied to the data before ML model training, wrapper methods generate and evaluate subsets of features based on the performance of the respective trained model, and embedded methods select features during model construction. Indeed, many of the ML models we will discuss in the ensuing sections, e.g.,random forest, boosting trees and the elastic net, perform embedded feature selection.

Throughout this thesis, we investigate the performance of a variety of DR methods, especially in Chapter 7, where we compare a diverse set of established FE and FS approaches. Furthermore, we developed a literature-driven FS strategy for MERIDA, which we will present in Chapter 5. In addition, our method SAURON-RF, which we introduce in Chapter 6 and extend in Chapter 8, employs a supervised FS algorithm by Kwak and Choi [30] that is based on the minimum-redundancy-maximum-relevance (MRMR) principle. In the following, we will briefly describe this filter algorithm.

Approaches based on the minimum-redundancy-maximum-relevance principle aim to identify a set of features with a strong dependency on the response variable, i.e., a large relevancy, while simultaneously exhibiting a weak dependency on each other, i.e., having a low redundancy. We already discussed measures of dependence between variables, such as the Pearson correlation coefficient and the Spearman correlation coefficient, in the context of performance measures for training ML models (cf. Section 4.2.2). Kwak and Choi [30] employ mutual information as a measure of mutual dependence between variables. The mutual information between two discrete random variables $X \in \mathscr{X}$ and $Y \in \mathscr{Y}$ is a symmetric measure that indicates how much we can reduce the entropy of one variable by knowing the other variable. With the entropy as the average level of information of a variable $X \in \mathscr{X}$ given by [175]

$$H(X) = -\sum_{x \in \mathscr{X}} \Pr(x) \cdot \log(\Pr(x)) \tag{4.12}$$

and the conditional entropy defined as

$$H(X|Y) = -\sum_{y \in \mathscr{Y}} \sum_{x \in \mathscr{X}} \Pr(x, y) \cdot \log\left(\frac{\Pr(x, y)}{\Pr(y)}\right) \tag{4.13}$$

the mutual information can be defined as [175]

$$I(X;Y) = H(X) - H(X|Y) = \sum_{y \in \mathscr{Y}} \sum_{x \in \mathscr{X}} \Pr(x, y) \cdot \log\left(\frac{\Pr(x, y)}{\Pr(x) \cdot \Pr(y)}\right) \tag{4.14}$$

It is 0 iff X and Y are independent, and its magnitude corresponds to the strength of dependence.

The approach by Kwak and Choi represents a heuristic supervised MRMR feature selection method typically applied as a filter before training the final ML model. In their approach, features are greedily added to the set of already selected features based on a tradeoff between the mutual information of the potential features with the response and the mutual information of the potential features with the already selected features. To this end, let $F = \{X_1, \dots, X_P\}$ initially denote the set of all available features and Y the response variable. Furthermore, let K be the number of features to be selected and $S = \{\}$ the set of already selected features, which is initially empty. Now, we iteratively add that feature $X_i \in F$ to S that maximizes the following term:

$$\max_{X_i \in F} I(Y; X_i) - \sum_{X_s \in S} \frac{I(Y; X_s)}{H(X_s)} \cdot I(X_i; X_s) \tag{4.15}$$

After a feature X_i is selected, F and S are updated:

$$F \leftarrow F \backslash \{X_i\}, \quad S \leftarrow S \cup \{X_i\} \tag{4.16}$$

This procedure is repeated until the desired number of features is selected, i.e., until $|S| = K$.

Group partitioning

The cancer cell line drug response data is typically represented by a real-valued summary metric of the dose response curve, such as the commonly used IC50. This results in the continuous response vector $\mathbf{y} \in \mathbb{R}^N$ for one particular drug. Yet, one might be interested in predicting only whether a specific cell line will respond to a drug (sensitive cell line) or not (resistant cell line) instead. Indeed, this may also be advantageous from an ML perspective since we may eliminate some aleatoric uncertainty of the continuous values and, apart from that, reduce the risk of overfitting. To this end, the continuous response has to be discretized. For the binarization of IC50 values, Knijnenburg et al. [27] developed a heuristic outlier procedure as part of their LOBICO method. We present this procedure in Chapter 5, where we also discuss the LOBICO classifier. We also developed a novel drug sensitivity measure (cf. Chapter 8), which we discretized using partitioning around medoids (PAM) that we outline in the ensuing paragraph.
Partitioning around medoids, also known as k-medoid clustering, is an unsupervised ML algorithm that assigns each point to one of k groups (called clusters) based on some dissimilarity measure such as Euclidean distance [176]. More specifically, the dissimilarity between points assigned to a cluster and the designated center point of that cluster is minimized. In k-medoid clustering, the center point, the so-called medoid, is an actual datapoint instead of an arithmetic mean, such as in k-means

clustering. Generally, the PAM algorithm consists of two phases: an initialization phase and a swapping phase. During the initialization phase, k cluster medoids are greedily selected such that they minimize some cost function defined by the dissimilarity measure. Then, the remaining points are associated with the closest medoid. In the swapping phase, all potential exchanges (swappings) between a medoid and a non-medoid point are considered, and it is evaluated whether they would improve the cost. The best swapping is executed in case it improves over the current clustering. Otherwise, the algorithm terminates.

4.3.3 Linear methods

Our previous considerations have been focused on a general description of ML model training. In this section and the ensuing ones, we now explicitly address the question of how we can achieve this for particular ML model types. This section is specifically dedicated to linear regression, traditionally regarded as interpretable. Regarding the taxonomy in Figure 4.3, the interpretability of linear models translates to model transparency in all three listed aspects (simulatability, decomposability and algorithmic transparency).
For our following description of linear methods, we use $X_1, \ldots, X_P$ to refer to the predictor variables, and Y denotes the response variable. For linear regression, we assume that the true relationship between the feature variables and the response variable is linear, i.e.,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_P X_P + \epsilon. \tag{4.17}$$

Given a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times P}$ and the corresponding response vector $\mathbf{y} \in \mathbb{R}^N$, we can estimate the coefficients with the ordinary least squares minimizer [177, 178]

$$\hat{\beta} = \text{argmin}_\beta \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{P} x_{ij}\beta_j)^2 \tag{4.18}$$

with $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_P)^T$. In the case of the existence of several highly correlated feature variables, the coefficients of a linear model suffer from high variance. To alleviate this issue, regularization techniques can be employed [127, 178]. To this end, Hoerl and Kennard proposed the L_2-penalized least squares estimate known as ridge regression [177]. Here, the coefficients of the ordinary least squares estimate (cf. Equation 4.18) are shrunken by the following constraint

$$\sum_{j=1}^{P} \beta_j^2 \leq c, \tag{4.19}$$

where c limits the size of the coefficients. By penalizing the coefficients using the L_2-norm, they can be shrunken but not set exactly to zero [178]. In order to render a model more interpretable (in the sense of simulatability, see Figure 4.3), we might pursue the goal of reducing several of the coefficients exactly to zero. Therefore, Robert Tibshirani introduced the L_1-penalized ordinary least squares estimate, also known as least absolute shrinkage and selection operator (lasso) [179]. The lasso constraint takes the following form

$$\sum_{j=1}^{P} |\beta_j| \leq d, \tag{4.20}$$

where d determines the amount of shrinkage. However, there are still some limitations of the lasso method. We refer the reader to [178] for a thorough discussion. Two of the arguments presented therein are the following: Firstly, if a group of variables with high pairwise correlations exists, the lasso tends to select only one of them instead of the whole group. Secondly, if there are more features than samples ($P > N$), the lasso can select N features at most, which can lead to the neglect of important features. As a remedy, Zou and Hastie suggested the elastic net contraint, which is a convex combination of the ridge and lasso constraints [178]:

$$\alpha \sum_{j=1}^{P} |\beta_j| + (1-\alpha) \sum_{j=1}^{P} \beta_j^2 \leq t, \ \alpha \in [0,1], \tag{4.21}$$

given a threshold t. For the two edge cases ($\alpha = 0$ or $\alpha = 1$), this constraint degenerates to the ridge and lasso constraint, respectively. Otherwise, the estimator inherits properties of both methods: the elastic net constraint can set coefficients to zero exactly like the lasso but also shrinks coefficients like in ridge regression [178].

4.3.4 Tree-based methods

One of the most inherently interpretable, i.e., transparent, methods from ML are decision trees since they mimic the human decision-making process in many aspects: they consist of internal nodes representing decision problems, branches representing the selectable decisions, and leaves representing the final outcome of the complete decision process [126]. Thus, they generate relatively simple if-then rules for decision-making. In the following, we will first briefly describe the construction of a single decision tree before elaborating on more complex tree-based methods. Note that we limit the descriptions of tree-based methods to quantitative features and response variables. Yet, tree-based methods can work with qualitative feature and response variables as well.
Once again, let $\mathbf{X} \in \mathbb{R}^{N \times P}$ be the feature matrix and $\mathbf{y} \in \mathbb{R}^N$ be the correspondingly ordered response vector. The central model assumption of tree-based methods is that the space in which our samples live can be divided into a set of simple, non-overlapping box-type regions, with each region corresponding to one common response value. Thus, building a decision tree must entail the identification of suitable regions $R_1, \ldots, R_L$ using some loss function. Let $\hat{y}_{R_l}$ represent the predicted value of the l-th box, e.g., the mean response of the samples in this box. If we employ the residual sum of squares as a loss function, the goal of decision tree building is to find boxes $R_1, \ldots, R_L$ that minimize [126]

$$\sum_{l=1}^{L} \sum_{i \in R_l} (y_i - \hat{y}_{R_l})^2. \tag{4.22}$$

A popular estimator of Equation 4.22 is a greedy algorithm known as binary recursive splitting [126]. The algorithm starts with determining a feature variable X_j from $X_1, \ldots, X_P$ and a cutpoint s that achieves the greatest reduction in error as given below. Let $R_{\text{left}}(j, s) = \{i | x_{ij} < s\}$ and $R_{\text{right}}(j, s) = \{i | x_{ij} \geq s\}$ denote the two emerging regions from a putative split. Then, we seek to minimize

$$\sum_{i \in R_{\text{left}}(j,s)} (y_i - \hat{y}_{R_{\text{left}}})^2 + \sum_{i \in R_{\text{right}}(j,s)} (y_i - \hat{y}_{R_{\text{right}}})^2. \tag{4.23}$$

This procedure is repeated recursively for each region until a tree is fully grown or some stopping criterion is fulfilled. For instance, we may stop tree splitting as soon as a minimal number of samples per leaf is reached. If we want to predict the response value of a new sample x, we can trace a path from the first node (the root) to a leaf and predict the value of the corresponding region.

Bagging and random forests

Decision trees, as described above, exhibit a high variance and are thus prone to overfitting. As a remedy, Leo Breiman proposed following an ensemble approach called bagging, also known as bootstrap aggregating [180]. Bootstrapping is a resampling method in which several new data sets are generated from an original dataset by drawing samples with replacement. In bagging using decision trees, we first generate B bootstrapped data sets from the original one and build a tree for each data set. Then, we average the predictions of the B individual trees to determine the overall prediction (aggregation step). Later, Leo Breiman suggested further reducing variance by decreasing the correlation between the trees by investigating a random subset of features as splitting candidates instead of the whole feature space. This algorithm is known as random forests [181], and we thoroughly investigate it below. Suppose $\mathbf{X} \in \mathbb{R}^{N \times P}$ is the model matrix, $\mathbf{y} \in \mathbb{R}^N$ the response vector, and B the number of trees in the forest. We first draw B bootstrap data sets. For each data set $b \in \{1, \ldots, B\}$, we build one decision tree. For each tree, we start tree building in the root node that contains all bootstrap samples (i.e., data points that are part of one bootstrap data set b) and repeat the following steps until some stopping criterion, such as the minimal number of samples per leaf, is fulfilled:

1. For each current leaf node that does not yet satisfy the stopping criterion, we draw $m < P$ features without replacement from the set of features.

2. For each drawn feature, we assess the quality of its putative splitting points, i.e., we determine by how much a binary division of the samples at each specific splitting point can improve the used error measure.
   To this end, let v be the current node and let $\delta(v)$ represent all bootstrap samples that belong to this node. Note that $\delta(v)$ can contain some of the initial samples twice and that we treat them as unique samples of the corresponding tree for the respective equations. Let y^v be the known response vector for the bootstrap samples falling into node v and let w_n^v be sample weight that

reflects the desired importance of sample n. Typically, all samples are weighted equally, and $w_n^v = \frac{1}{|\delta(v)|}$. However, the weights can be set to meet user-defined properties, which we will discuss when we introduce SAURON-RF in Chapter 6.
The predicted response of our current node v is constant for each sample in that node and given by

$$\forall i \in \delta(v) : \hat{y}_i^v = \sum_{n \in \delta(v)} w_n^v \cdot y_n^v \tag{4.24}$$

The remaining error in that node can be measured using the MSE between the known response y^v and the predicted response $\hat{y}^v$

$$\text{MSE}(y^v, \hat{y}^v) = \sum_{n \in \delta(v)} w_n^v \cdot (y_n^v - \hat{y}_n^v)^2. \tag{4.25}$$

For each splitting point of a feature, we can assess its quality by the improvement of the error by splitting node v into the two nodes v_r and v_l

$$\begin{aligned} &w_{\text{an}}(v) \cdot (\text{MSE}(y^v, \hat{y}^v) - \\ &w_{\text{ch}}(v_r) \cdot \text{MSE}(y^{v_r}, \hat{y}^{v_r}) - \\ &w_{\text{ch}}(v_l) \cdot \text{MSE}(y^{v_l}, \hat{y}^{v_l})). \end{aligned} \tag{4.26}$$

Here, $w_{\text{an}}(v)$, $w_{\text{ch}}(v_r)$, and $w_{\text{ch}}(v_l)$ are node-specific weights for the ancestor and child nodes, respectively, which can for example represent the fraction of samples assigned to a node. In particular, $w_{\text{an}}(v) = \frac{|\delta(v)|}{N}$, $w_{\text{ch}}(v_r) = \frac{|\delta(v_r)|}{|\delta(v)|}$, and $w_{\text{ch}}(v_l) = \frac{|\delta(v_l)|}{|\delta(v)|}$ are typical choices for a regression forest [182].

3. Finally, we can use the feature and splitting point for which Equation 4.26 is maximized to divide the samples into two groups. The respective splitting criterion of the feature then represents an internal node of the tree, and the two groups become the children of this node.

For a new sample $x \in \mathbb{R}^P$, the prediction of a single tree b can then be calculated as the average of the response values in the reached leaf node. Let μ be the leaf node that is reached by sample x, then the prediction of a single tree can be calculated as described above (cf. Equation 4.24)

$$\hat{f}_b(x) = \sum_{n \in \delta(\mu)} w_n^{\mu} \cdot y_n^{\mu} \quad . \tag{4.27}$$

The random forest prediction can subsequently be obtained by averaging the predictions of all of the trees. Similar to the sample-specific weights, the trees can be assigned weights that quantify their importance for the prediction and the formula for the prediction is given by

$$\hat{f}(x) = \sum_{b=1}^{B} w_b(x) \cdot \hat{f}_b(x) \quad . \tag{4.28}$$

Here, $w_b(x)$ is the tree-specific weight, which is set to $\frac{1}{B}$ in a conventional random forest to obtain the simple average of the trees. When we introduce SAURON-RF in Chapter 6, we will propose other weighting schemes.
In comparison to an ordinary decision tree, random forests lack interpretability. Thus, augmenting them with post hoc explanations in the form of feature importance values is common. One possibility to measure feature importance is via error reduction. Here, the importance of a particular feature is given by the (normalized) average of error reductions over all splits that feature was involved in [183].

Boosting

Another approach able to improve the predictive performance of decision trees is known as boosting and was proposed by Robert Schapire in 1990 [184]. Instead of fitting many trees to bootstrapped data sets of the original data set, the boosting technique relies on fitting trees sequentially on data sets modified through previously trained trees. Thus, the first tree is trained on the original data set. All ensuing trees are fit using the residuals, i.e., the deviation of the predicted values from the true values, of the current boosted forest. The boosting algorithm can be briefly summarized as follows: Suppose $\mathbf{X} \in \mathbb{R}^{N \times P}$ is the model matrix, $\mathbf{y} \in \mathbb{R}^{N}$ the response vector, and B the number of trees in the forest. Moreover, let d be the number of splits in each tree and let $\mathbf{r} \in \mathbb{R}^N$ denote the current residuals. Initially, we set $\hat{f}(x) = 0$ and $\mathbf{r} = \mathbf{y}$. Tree boosting consists of the following three steps that are executed until B trees are fit [126]:

1. We fit a tree to the current training data given by $(\mathbf{X}, \mathbf{r})$ resulting in the model $\hat{f}^b(x)$

2. We update the total model $\hat{f}(x)$ by adding a scaled version of $\hat{f}^b$:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{4.29}$$

   Here, λ is called shrinkage parameter. It is a small positive value representing the learning rate.

3. We update the residuals by calculating

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i), \quad \forall i \in \{1, \dots, N\}. \tag{4.30}$$

The final prediction of the boosted forest can be determined by calculating

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{4.31}$$

4.3.5 Neural networks

In the introductory section of this chapter, we already mentioned that artificial neural networks have been part of machine learning since its beginning in the 1950s. Probably, they are the most versatile and flexible already existing ML method. They can be utilized to solve problems from all four ML realms alike (for examples see [185, 186, 187, 188]) and are thought to have universal approximation capability [144], which means that they are flexible enough to safely assume that model uncertainty disappears when using them. However, their strength can simultaneously be considered their Achilles' heel: their flexibility results from model complexity in terms of (hyper)-parameters, which is coupled to relatively high estimation uncertainty. Thus, fitting neural networks is computationally expensive. Moreover, it typically requires extensive data sources to alleviate estimation uncertainty. Historically speaking, neural networks were even replaced by less computationally expensive ML methods for a certain period [189]. Owing to technological advances in hardware and software and the availability of suitable large data sources, they, however, experienced a revival in the last decade. During the genesis of this thesis, we repeatedly investigated neural networks for performing drug sensitivity prediction. However, their results were not surpassing those of less computationally expensive ones such as random forests. In the last chapter of this thesis, we will discuss the reasons for this observation (cf. Chapter 9). Since neural networks are not the focus of this thesis, we limit our description to specific aspects required to understand the parts of this thesis where they are nevertheless applied.

Feedforward neural networks in supervised ML

Generally, neural networks can be described as graphs consisting of nodes representing neurons and edges representing their connections. In Figure 4.4, we show an example of a simple fully-connected feedforward neural network as employed for supervised learning. Feedforward neural networks can be interpreted as a special form of directed acyclic graphs with the following topology: they comprise an input layer that consists of nodes corresponding to predictor variables, one or several consecutive hidden layers that transform and forward the information from one layer to the next, and an output layer representing the desired response variable(s). Note that we restrict our description to a single output node in the following. The network is called feedforward because there is an unidirectional flow of information from the input layer to the output layer, and thus, no feedback between layers occurs [190, 191]. Typically, each edge has an associated weight, and each node in the hidden and output layer has a node bias. The edge weights and the node biases are known as the parameters of a neural network that we have to determine by model training. Moreover, each node transforms the received signal from the previous layer by application of a (non-linear) activation function.

Mathematically speaking, we can think of a (fully-connected feedforward) neural network as a nesting of (non-)linear functions and formulate them as described in the following. To this end, we use $X_1, \ldots, X_P$ to refer to the predictor variables and Y

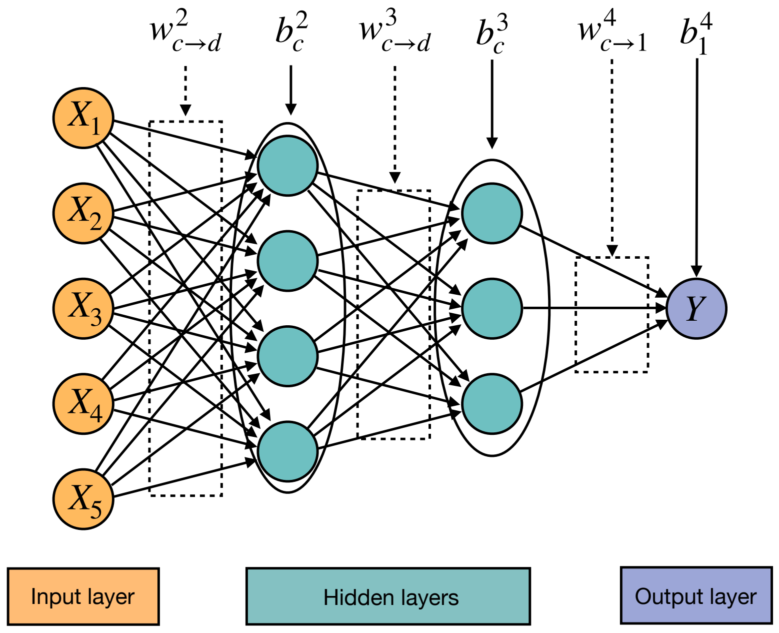


Figure 4.4: Simple fully-connected feedforward neural network. This figure depicts an example network architecture for a simple feedforward neural network with four input nodes and one output node.

to refer to the response variable. Let L be the number of layers in the network, and let K_l denote the number of nodes in layer l. Moreover, we use the variable name b^l_c to refer to the bias of node c in layer l and $w^l_{c\to d}$ to refer to the weight of an edge connecting node c of layer $l-1$ to node d of layer l. Additionally, we are given some non-linear function σ, known as activation function, that is identical across all nodes. One such activation function is the ReLU function $\sigma = \max(0, x), \sigma : \mathbb{R} \to \mathbb{R}^+_0$. Indeed, it could be a different function for each node of the neural network, but this is typically not investigated. Given these variables, we assume that our response variable Y can be expressed as

$$Y = \sigma(b^L_1 + \sum_{k=1}^{K_{L-1}} w^L_{k\to 1} \cdot \mathrm{out}^{L-1}_k) + \epsilon. \tag{4.32}$$

Here, the prediction, i.e., the output of the last node, is defined recursively depending on the outputs of the nodes in the previous layer (out_k^{L-1}). Generally speaking, it holds that

$$\text{out}_k^{L-1} = \begin{cases} X_k, \text{ if } L-1=1, \text{ i.e., the previous layer is the input layer} \\ \sigma(b_k^{L-1} + \sum_{m=1}^{K_{L-2}} w_{m\to k}^{L-1} \cdot \text{out}_m^{L-2}), \text{ otherwise.} \end{cases} \tag{4.33}$$

Network training

Like all previously discussed ML models, neural networks can be trained by minimizing an error function such as the mean-squared error. To this end, the training of neural networks involves the following three steps

1. Parameter initialization: Initially, the weights and biases of the neural network have to be fixed to some numerical values. There exists a variety of initialization methods (e.g., Glorot initialization [192] or He initialization [193]), yet their discussion is out of the scope of this thesis.

2. Network application and error calculation: The network is applied to the training data, resulting in the network predictions for all training samples. The performance of the current neural network is determined by comparing the predictions to the known actual values within an error function.

3. Parameter adaptation: Since the goal of the neural network training is the minimization of the error, we have to adapt the weights and biases so that the error is reduced. Typically, the so-called gradient descent algorithm is employed for this purpose [190]. Roughly speaking, we calculate the gradient of the error function, i.e., the first derivative of the error function with respect to all model parameters (weights and biases), to determine the direction of the steepest ascent. Then, each parameter is updated by slightly moving in the direction of its negative partial derivative, i.e., the direction of the steepest descent. In practice, the computational cost of computing the gradient is reduced by the application of the backpropagation algorithm, which is based on the idea of recursively applying the chain rule of calculus [194, 190].

The latter two steps are repeated until some stopping criterion is satisfied.

4.4 Reliability estimation via conformal prediction

The term *conformal prediction* was coined by Vovk, Gammermann, and Shafer in 2005 to describe a procedure that delivers statistically confident predictions for an arbitrary learning algorithm [195, 196]. To achieve confidence in predictions (for new samples previously unseen by the learning algorithm), their procedure relies on a specific similarity of the new sample to the training samples, which they apparently denote as *conformity* [195]. In Section 4.2, we discussed that creating trust in ML is a critical challenge for (healthcare) applications. In this section, we will outline how conformal prediction (CP) can be employed for this purpose.

Recall that the predictions delivered by ML models can readily be used to assess the overall model performance in terms of conventional error measures such as mean-squared error as long as the true response is known. While the evaluation of performance measures certainly indicates some form of reliability, we cannot use it to tell if a prediction for a previously unseen sample with an unknown response will be close to its actual but unknown value, which is what we refer to as reliability (cf. Section 4.4). Conformal prediction is a reliability estimation framework that can sit on top of various ML methods given that they provide a notion of predictive (un)certainty [197]. For random forest classifiers, such a notion of certainty can be represented by the proportion of trees that voted for the predicted class. Intuitively speaking, the more trees vote for a particular class, the more likely it becomes that this prediction is correct. Similarly, we may use some form of quantile regression as a notion of certainty for regression models, e.g., for random forest regressors. The CP framework then converts this notion into a mathematical rigorous certainty guarantee. For a user-specified maximal allowed error rate α, CP constructs a (preferably non-trivial) valid prediction set (classification) or interval (regression), which then contains the true value with a certainty of almost exactly $1 - \alpha$. Note that it is furthermore possible to interpret these sets or intervals via p-values assigned to the members. While we did not pursue this approach in this thesis, the interested reader can refer to [196] for more information.

In the following, we first introduce a general conformal prediction procedure that employs a notion of (un)certainty in a score function, also referred to as (non-)conformity measure, to convert it into a rigorous (un)certainty guarantee by delivering valid prediction sets and intervals. After describing the conformal predic-

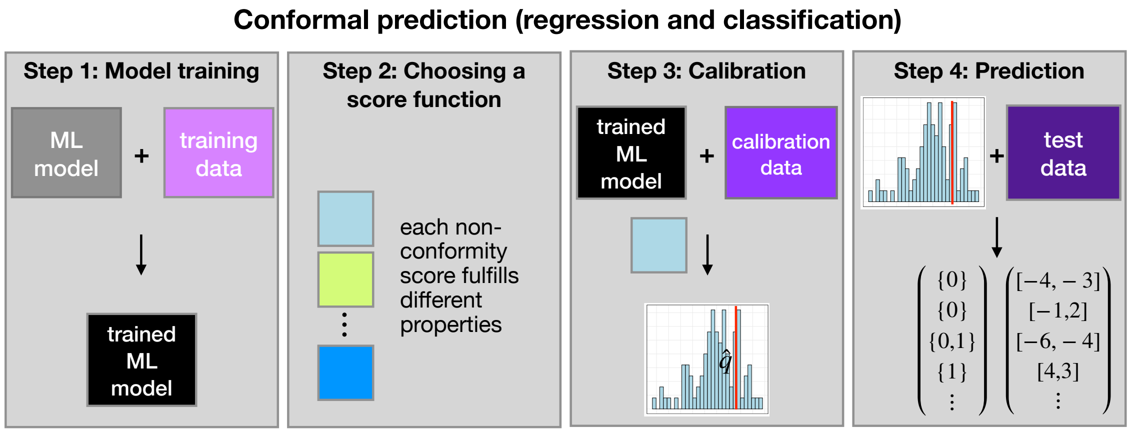


Figure 4.5: CP procedure. This figure depicts a typical workflow of a conformal prediction procedure.

tion algorithm, we present the score functions that we evaluated in Chapter 8 in the context of reliability estimation for anti-cancer drug sensitivity prediction.

4.4.1 Conformal prediction procedure

Training supervised ML models generally includes partitioning the complete data set into a disjoint training and test data set (cf. Section 4.3.1). While the training data set usually serves for the training of the parameters of a particular ML model, the test set is used to evaluate the performance of this model on data previously unseen by the model. Conformal prediction needs a third disjoint data set, the so-called calibration data set employed to calculate statistics on the (un)certainty of the model. For our application case, let $\mathbf{Z} = (\mathbf{X}, \mathbf{y})$ be the complete data set with $\mathbf{X} \in \mathscr{X}$ as feature matrix and $\mathbf{y} \in \mathscr{Y}$ as correspondingly ordered response vector. Note that $\mathbf{y}$ can be discrete or continuous. Let $\mathbf{Z}_{\text{train}}$, $\mathbf{Z}_{\text{cal}}$, and $\mathbf{Z}_{\text{test}}$ be the disjoint training, calibration, and test set, respectively. Moreover, let N_{train}, N_{test}, and N_{cal} denote the number of samples in each of these data sets, and let $\alpha \in [0, 1]$ be the desired maximal error rate. Then, CP can be divided into the ensuing four steps [197]

1. First, we train the chosen ML model using $\mathbf{Z}_{\text{train}}$.

2. Then, we define a score function $s(x, y)$ that is based on the given notion of (un)certainty by the model.

3. Next, we apply the trained model to $\mathbf{Z}_{\text{cal}}$, and calculate one score for each calibration sample. Based on the resulting score distribution, we derive a threshold $\hat{q}$ that corresponds to the allowed error rate α.

4. Lastly, we calculate the corresponding scores for $\mathbf{Z}_{\text{test}}$ and use $\hat{q}$ to form intervals (regression) or sets (classification).

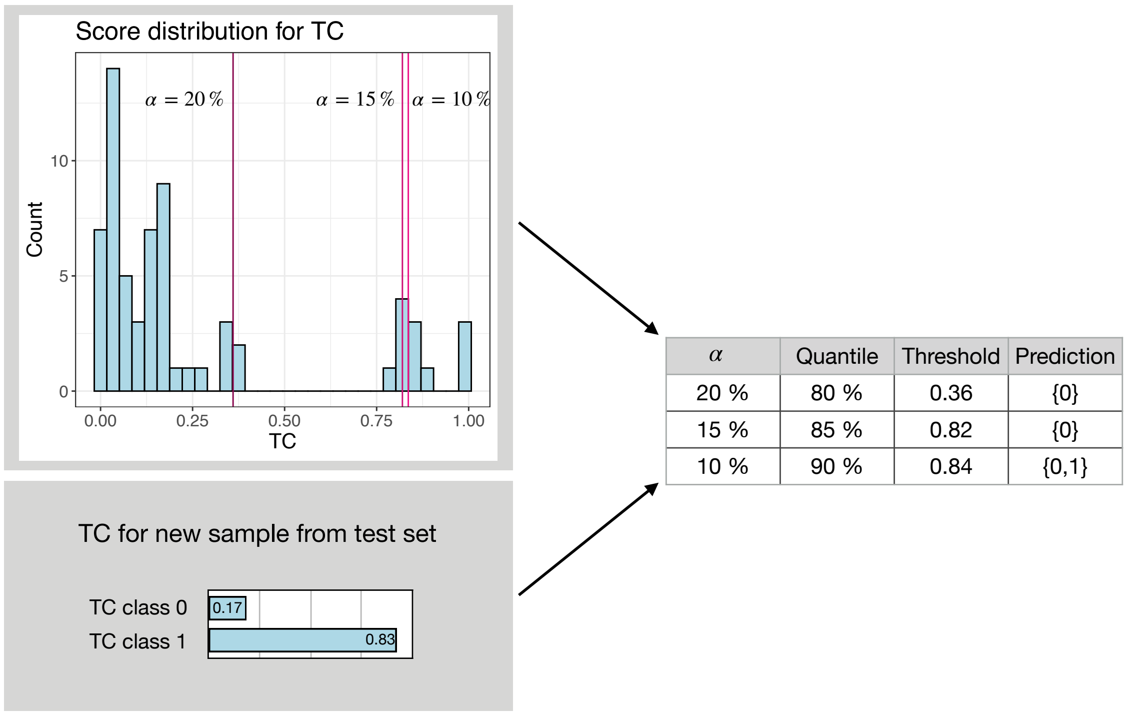
In Figure 4.5, we visualize the CP procedure. By performing CP as outlined above, we construct intervals or sets that contain the true response with a probability of almost exactly $1 - \alpha$, which are also called valid prediction intervals or sets. More specifically, let $\mathcal{C}(x_i)$ represent this interval or set for $x_i \in \mathbf{X}_{\text{test}}$, CP guarantees marginal coverage (certainty) [197]

$$1 - \alpha \leq \Pr(y_i \in \mathcal{C}(x_i)) \leq 1 - \alpha + \frac{1}{N_{\text{cal}} + 1}. \tag{4.34}$$

This equation is referred to as marginal coverage property of CP since the certainty (coverage) is averaged (marginalized) over the randomness in the test and calibration data points [197]. For this guarantee to hold, we do not need to assume that our random variables are drawn iid. It suffices that they are drawn exchangeable [197], i.e. their underlying joint probability distribution is invariant to finite permutations. From Equation 4.34, we can also deduce that the more calibration samples are available, the lower the upper boundary becomes, i.e., the coverage would become exactly $1 - \alpha$ for $N_{\text{cal}} \to \infty$. Indeed, the relationship between N_{cal} and the observed coverage can be described analytically. We refer to [198, 197] for in-depth information on this issue.
Instead of marginal coverage, we would usually like to guarantee conditional coverage, meaning that we guarantee the coverage for a particular sample, i.e., we would like to guarantee

$$\Pr(y_i \in \mathcal{C}(x_i) | x_i) \geq 1 - \alpha. \tag{4.35}$$

While it is impossible to achieve conditional coverage with CP in all possible scenarios according to Vovk [198], it can be approximated with appropriate scores [197]. Therefore, we also assessed our models and the ensuing score functions in that respect in Chapter 8.



α	Quantile	Threshold	Prediction
20 %	80 %	0.36	{0}
15 %	85 %	0.82	{0}
10 %	90 %	0.84	{0,1}

Figure 4.6: True-class (TC) score example. This figure exemplifies how to apply CP to an unknown sample using the True-Class score with different maximal allowed error rates.

4.4.2 Classification scores

As mentioned above, CP consists of four steps. In particular, step two requires defining a score function based on the notion of (un)certainty given by the model. The choice of score function heavily influences the quality of results [197]. Angelopoulos and Bates [197] thoroughly discuss a variety of criteria that can play a role in selecting the best score function for different application cases. In the following, we will briefly describe the score functions evaluated for reliable SAURON-RF presented in Chapter 8. We start with classification scores and finally present a regression score based on quantile regression. For the following descriptions, let $\mathbf{d}$ denote a discrete response vector potentially containing k different classes, $\mathbf{y}$ denote a continuous response vector and $\mathbf{X}$ be the feature matrix.

True-class (TC) score. Arguably, the most simple scoring function that Angelopoulos and Bates [197] represents the probability of misclassifying a sample. Given a sample x of class d, it is defined as

$$s_{\mathrm{TC}}(x,d) = 1 - \hat{\Pr}(d|x) \quad . \tag{4.36}$$

Here, $\hat{\Pr}(d|x)$ is an estimate of the probability $\Pr(d|x)$ based on the trained classifier. For a random forest, $\hat{\Pr}(d|x)$ is the proportion of trees that voted for the true class d of the calibration sample x. The True-class score results in high values if the true class of sample x had a low probability and vice versa. As described in Step 3 of the CP procedure, we calculate this score for each sample x_j in $\mathbf{Z}_{\mathrm{cal}}$, resulting in a score distribution. Based on this distribution, we derive the threshold $\hat{q}$ that tells us which classes to add to our prediction set to satisfy the marginal coverage property in Equation 4.34. In particular, we calculate $\hat{q}$ as a modified (1-α)-quantile of the distribution. We must modify the usual $(1-\alpha)$-quantile to account for the finite number of calibration samples N_{cal}. Thus, we determine $\hat{q}$ as the $\frac{\lceil (N_{\mathrm{cal}}+1)(1-\alpha)\rceil}{N_{\mathrm{cal}}}$-quantile. Note that $\frac{\lceil (N_{\mathrm{cal}}+1)(1-\alpha)\rceil}{N_{\mathrm{cal}}}$ can be larger than 1. In that case, we cannot satisfy the desired maximal error rate α for the given number of calibration samples N_{cal}. For a new sample x_i, we do not know the true class. Hence, we calculate the score for all classes and add those with a score smaller or equal to $\hat{q}$ to the prediction set, i.e.,

$$\mathcal{C}(x_i) = \{c_l | s_{\mathrm{TC}}(x_i, c_l) \leq \hat{q}, \quad \forall l \in \{1,\dots,k\}\} \tag{4.37}$$

In Figure 4.6, we exemplify how to apply the TC score to an unknown sample using different values of α.

Summation (Sum) score. Angelopoulos and Bates propose another score function based on ideas from [199, 200]. We call this score function Summation score since it builds on the concept of summing up the probabilities of all classes until the true class is reached. The Summation score for a sample x with true class d is calculated as follows

1. Firstly, we generate a prediction for sample x and sort the resulting estimated probabilities for all classes $c_l \in \{1,\dots,k\}$ decreasingly from highest to lowest probability. For random forest classifiers, this probability is again given by the

proportions of samples that voted for a class. W.l.o.g., let $u = [c_1, \dots, c_k]$ be this sorted list for the sample x.

2. Secondly, we add up the probabilities of all classes in this sorted list until the true class d of sample x is reached:

$$s_{\text{Sum}}(x, d) = \sum_{o=c_1}^{d} \hat{\Pr}(o|x) \tag{4.38}$$

Roughly speaking, the behaviour of the Summation score can be summarized as follows. If the Summation score results in high values, it was either because the correct class was predicted with high probability or the sample was misclassified and several class probabilities had to be summed up until the true class was reached. If the score is comparably low, the correct class was predicted with a low probability. Once again, we calculate the score for each sample in $\mathbf{Z}_{\text{cal}}$ to obtain the score distribution. To subsequently derive the threshold $\hat{q}$ that we need to decide whether to include a class into our prediction set for a new sample, we calculate the adjusted $(1-\alpha)$-quantile $\hat{q}$ as described for the True-class score. For a new sample x_i, we do not know the true class. However, we can use $\hat{q}$ to identify all classes that need to be added to the prediction set $\mathcal{C}(x_i)$ by performing the two steps above in a slightly modified manner. At first, we sort the predicted class probabilities from highest to lowest. Again, w.l.o.g. let $u_i = [c_1, \dots, c_k]$ be this sorted list for sample i. We obtain $\mathcal{C}(x_i)$ by adding all classes until, in sum, their predicted class probabilities exceed $\hat{q}$

$$\mathcal{C}(x_i) = \{c_l \,| l \in \{1, \dots, \max\{l' : \sum_{u=1}^{l'} \hat{\Pr}(c_u|x_i) < \hat{q}\} + 1\}\} \tag{4.39}$$

Mondrian (Mon) score. The Mondrian score is a type of class-conditional CP in which the marginal coverage from Equation 4.34 is extended to hold for each available class, i.e., the predicted sets for a new sample x_i from the test set should fulfil [201, 197, 202]

$$\Pr(d_i \in \mathcal{C}(x_i) | d_i = c_l) \geq 1 - \alpha, \quad \forall l \in \{1, \dots, k\}. \tag{4.40}$$

Instead of generating one score distribution in the calibration step of CP, the general idea of Mondrian CP is to perform the calibration step in each class separately. Thus, using the True-class score, Mondrian CP can be conducted as follows: we calculate s_{TC} for each sample from the calibration data set and divide the resulting distribution into k sub-distributions, one distribution for each class. We then determine the modified $(1-\alpha)$-quantile for each distribution, resulting in k thresholds $\hat{q}^l, l \in \{1, \dots, k\}$. For a new sample x_i, we do not know the true class. Thus, we calculate the score for each class and add a class c_l to the prediction set $\mathcal{C}(x_i)$ if it fulfills $s_{\text{TC}}(x_i, c_l) \leq \hat{q}^l$.

4.4.3 Regression score

As outlined in Section 4.4.1, regardless of whether classification or regression is performed, the CP procedure can be applied as long as an appropriate score function is provided. Romano et al. developed a CP method based on quantile regression [203]. In principle, we can already employ quantile regression itself to provide an estimate of the certainty of the regression: we can train one model $\hat{f}_{\frac{\alpha}{2}}$ that predicts the $\frac{\alpha}{2}$-quantile and another model $\hat{f}_{1-\frac{\alpha}{2}}$ that predicts the $1-\frac{\alpha}{2}$-quantile and expect the interval $[\hat{f}_{\frac{\alpha}{2}}, \hat{f}_{1-\frac{\alpha}{2}}]$ to contain the true response with $1-\alpha$ certainty. However, we do not know how accurate the predicted intervals are because they were calculated on the training data set. Thus, Romano et al. define a score function, which we also call Quantile (Qu) in the following, that quantifies whether the samples from the calibration data set were within the signified quantile interval as often as expected. For a sample x from the calibration data set with its known response y, this score function represents the signed distance between y and the nearest interval boundary

$$s_{\text{Qu}}(x, y) = \max \begin{cases} \hat{f}_{\frac{\alpha}{2}}(x) - y \\ y - \hat{f}_{1-\frac{\alpha}{2}}(x) \end{cases} \tag{4.41}$$

The sign of this score function is positive if y is outside of the interval and negative if y is within the interval. Again, we calculate s_{Qu} for each sample of the calibration data set and receive a score distribution on which we determine $\hat{q}$ as the modified $(1-\alpha)$-quantile of the distribution. If the quantile regression achieves the desired coverage, $\hat{q}$ will be approximately 0, and the predicted interval for a new sample x_i will remain unaltered. Otherwise, the interval will be widened ($\hat{q} > 0$) or narrowed ($\hat{q} < 0$), i.e., the predicted interval is

$$\mathcal{C}(x_i) = [\hat{f}_{\frac{\alpha}{2}}(x_i) - \hat{q}, \hat{f}_{1-\frac{\alpha}{2}}(x_i) + \hat{q}]. \tag{4.42}$$

5 Rule identification using integer linear programming

In Chapter 4, we outlined which requirements a drug sensitivity prediction approach should meet at best to become trustworthy. Interpretability, i.e., the ability of a model to generate human comprehensible decisions, plays a major role in this specific area and, more generally, in the healthcare domain [204]. However, many ML models suffer from a lack of inherent interpretability (cf. Table 4.1 in Chapter 4). For example, (deep) neural networks, which recently gained much attention within the drug sensitivity prediction community (cf. Table 5.1), often deliver well-performing models with respect to error measures. Yet, their interpretation is impeded by complex model structures.
To address the demand for model interpretability, Knijnenburg et al. developed a classifier named LOBICO [27]. LOBICO is an integer linear programming formulation aiming to deliver Boolean logic-based rules as output. These rules then specify under which conditions a sample will be sensitive or resistant to a particular drug. In principle, Knijnenburg et al. build on work by Kamath et al. , who show how to solve the Boolean function synthesis problem (BFSP) with integer linear programming [205]. Boolean function synthesis is concerned with logically combining input variables such that an output variable is explained. For drug sensitivity prediction, the input is given by the model matrix that for each sample contains binary values for a set of variables (features) while the output is the binary drug response. Knijnenburg et al. use mutation data of 60 genes as input and a binarized version of IC50 values as output. To determine a binary drug response, they introduced a novel heuristic outlier procedure inspired by the previous observation of Garnett et al. that most cell lines will not be responsive to a (targeted) anti-cancer treatment [14]. This approach leads to a binarization threshold that Knijnenburg et al. then leverage to retain some of the continuous information in their model via sample-specific weights. Since the space of possible logic combinations grows rapidly when increasing the size of a logic formula or the number of employed input features, LOBICO suffers from runtime issues. Furthermore, LOBICO and most approaches for drug sensitivity prediction do not consider a priori pharmacogenomic knowledge

in terms of known biomarkers of drug response (cf. Table 5.1). While easily interpretable models reflecting molecular mechanisms of drug sensitivity are desirable, overly small models may not suffice to mirror the complexity of biological processes involved in cancer development and anti-cancer drug resistance. Hence, larger yet easily interpretable models are required to analyse cancer cell sensitivity.
To this end, we propose MERIDA (MEthod for Rule Identification with multi-omics DAta), a new ILP formulation based on the LOBICO approach. Analogous to LOBICO, we rely on an ILP formulation to synthesize a Boolean function. While LOBICO is principally able to fit any logic function to the given data, MERIDA is only allowed to infer a restricted Boolean formula, reducing the runtime tremendously. Consequently, MERIDA can build more comprehensive rules and simultaneously enables the consideration of more input features. In addition to using a predefined form of the Boolean formula, we also investigated the influence of other weight functions on the runtime of LOBICO and MERIDA. Our analyses demonstrate that different weight functions can further reduce the runtime while delivering almost identical rules as output. Finally, we integrate biomarkers of drug response into the Boolean formulas and gauge their effect on the prediction quality. In order to generate comprehensive lists of sensitivity/resistance biomarkers, we leveraged information from various well-established cancer-related databases: IntOGen ([22]), COSMIC ([23]), CIViC ([24]), OncoKB ([25]), and the Cancer Genome Interpreter (CGI) ([26]). However, a priori knowledge might not only be obtained from databases but can also be newly generated by interpretable models such as MERIDA. Therefore, we also applied MERIDA iteratively, i.e., we added features detected in previous runs as prior knowledge into the next run. Our results show that the iterative approach not only improves the statistical performance but also identifies more comprehensive sets of putative sensitivity biomarkers.
The remainder of this chapter is structured as outlined in the following: We first explain the ILP formulation of LOBICO. Afterwards, we introduce our novel approach, MERIDA, highlighting differences and commonalities to LOBICO. Lastly, we present the results of a case study using the GDSC data set that demonstrates the superiority of MERIDA over LOBICO.

Table 5.1: Comparison between different supervised ML methods with respect to used base method and biomarker integration. In this table, we compare various supervised ML approaches for drug sensitivity prediction. We list their underlying base methodology and information on whether they incorporated knowledge on a priori biomarkers for drug response prediction.

Name and author	Base method	Biomarker integration
Menden et al., 2013 [152]	neural network	✗
Zhang et al., 2015 [153]	similarity network	✗
LOBICO by Knijnenburg et al., 2016 [27]	integer linear program	✗
Stanfield et al., 2017 [166]	similarity network	✗
SRMF by Wang et al., 2017 [154]	matrix factorization	✗
HARF by Rahman et al., 2017 [155]	random forest	✗
HNMDRP by Zhang et al., 2018 [167]	similarity network	✗
Matlock et al., 2018 [156]	random forest, neural network, K nearest neighbour	✗
KRL by He et al., 2018 [157]	kernelized rank learning	✗
RWEN by Basu et al., 2018 [158]	elastic net	✗
CDRscan by Chang et al., 2018 [159]	neural network	✗
QRF by Fang et al., 2018 [31]	random forest	✗
NCFGER by Liu et al., 2018 [160]	similarity network	✗
DeepDR by Chiu et al., 2019 [161]	neural network	✗
Deep-Resp-Forest by Su et al., 2019 [168]	random forest	✗
netBITE by Oskooei et al., 2019 [162]	random forest	✗
Deng et al., 2020 [163]	neural network	✗
PathDSP by Tang et al., 2021 [164]	neural network	✗
MERIDA by Lenhof et al., 2021 [21]	integer linear program	✓
GraphDRP by Nguyen et al., 2022 [165]	neural network	✗
SAURON-RF by Lenhof et al., 2022 [29]	random forest	✗

Authors' contributions
This chapter is based on my publication *MERIDA: a novel Boolean logic-based integer linear program for personalized cancer therapy* [21] in terms of content and text. Hans-Peter Lenhof and I conceived the initial idea of developing a drug sensitivity prediction model using integer linear programming (ILP), which is the basis of the MEthod for Rule Identification in multi-omics DAta (MERIDA). A first version of MERIDA was then presented in my master's thesis (2018), which was mainly supervised by Lara Schneider. Until its publication in 2021 [21], this method has undergone substantial development including the incorporation and analysis of a larger drug sensitivity data set, the introduction of a substantially refined feature-selection and a priori knowledge annotation strategy, as well as changes in the ILP model equations itself. I implemented the software, conducted the computational experiments, and drafted the manuscript for this publication. Together with the remaining authors, I also analyzed and evaluated the results.

5.1 Using logic models to predict drug sensitivity in cancer

MERIDA (MEthod for Rule Identification in multi-omics DAta) aims to build easily interpretable logic rules that specify whether a cell line is sensitive or resistant to a drug. To this end, we decided to pursue the approach by Knijnenburg et al., who employ integer linear programming for synthesizing a Boolean function from a binary model matrix and a binarized drug response vector. By reducing the search space of allowed logic combinations, MERIDA can investigate a considerably higher input feature space and generate larger rules than LOBICO. In addition, we enabled the integration of prior knowledge in terms of known biomarkers of drug response, increasing the comprehensiveness of our output rules even more. In the following, we first discuss LOBICO before introducing MERIDA.

5.1.1 LOBICO

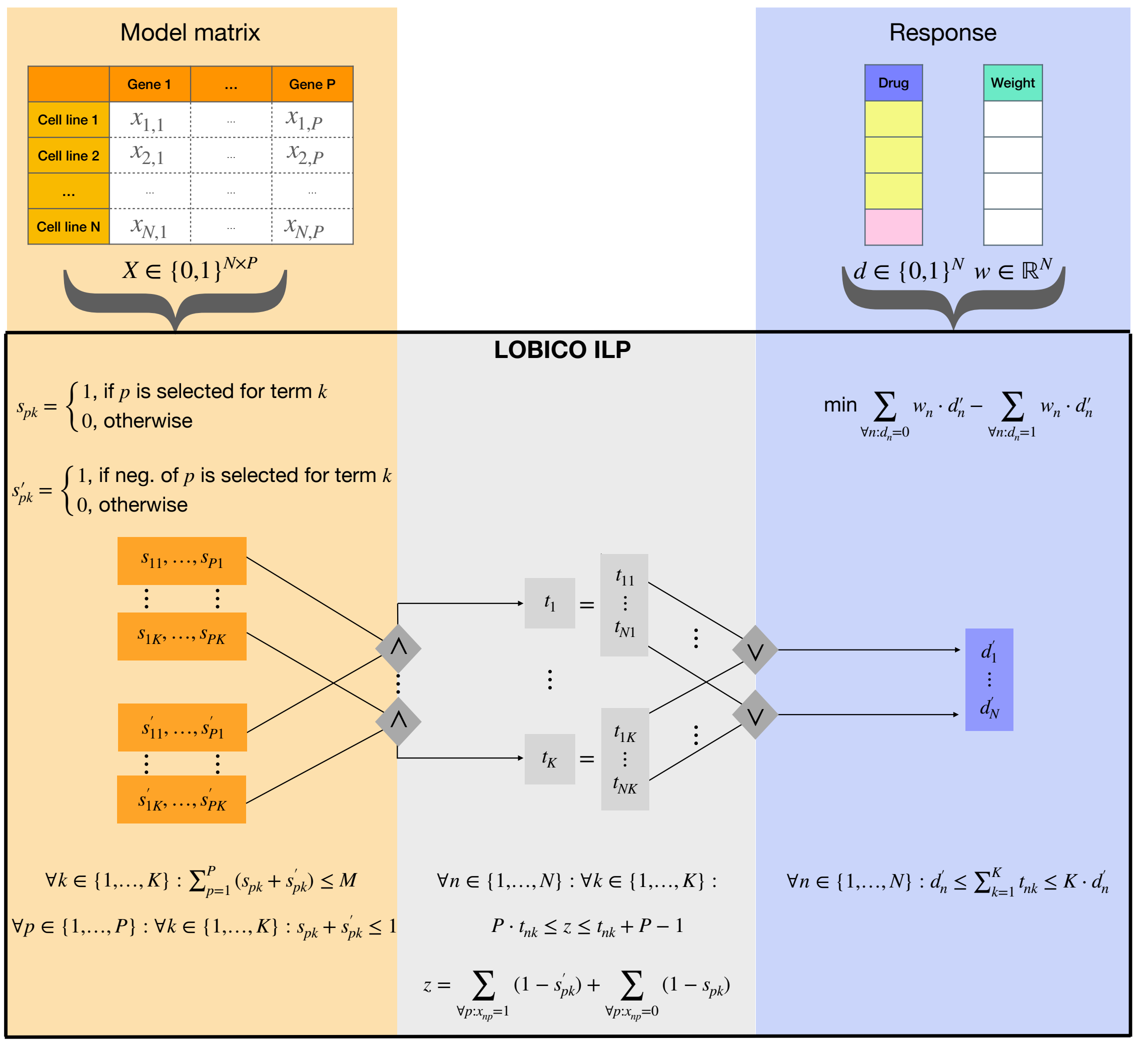
With LOBICO, Knijnenburg et al. aimed to address the demand for interpretability in drug sensitivity prediction. They decided to achieve this goal by formulating an integer linear program for Boolean function synthesis.

Discretization of drug sensitivity measurements

However, drug sensitivity prediction using the prevailing drug sensitivity measures, e.g., IC50 or AUC, is inherently a regression task. Thus, Knijnenburg et al. had to discretize the continuous drug response to apply a logic modelling approach. Motivated by the previous observation of Garnett et al. [14] that most cell lines will be non-responders to a specific (targeted) drug, they developed a heuristic outlier procedure for distinguishing sensitive cell lines from resistant ones. Briefly, they first gather all IC50 values for a specific drug and perform kernel density estimation. The highest point of the kernel density is then interpreted as the mean of the distribution of resistant cell lines, which they subsequently model as a normal distribution. By evaluating the corresponding cumulative normal distribution, they then derive the drug-specific threshold t. Roughly speaking, the threshold t corresponds to an input value of the cumulative normal distribution for which this cumulative normal distribution evaluates to a small value. Cell lines with an IC50 value below the threshold are called sensitive, the remaining ones are called resistant.
After binarization of the response, Knijnenburg et al. argue that the additional information such a continuous value may possess should not be discarded. Therefore, they introduce the sample-specific weights as an additional input. We will describe these weights in the ensuing section.

LOBICO ILP

The ILP formulation of LOBICO is based on work by Kamath et al. [205] who formulate the Boolean function synthesis problem as an ILP. Briefly, the goal of Boolean function synthesis is to find a Boolean function that satisfies constraints imposed by an (incompletely) specified truth table. In case of drug sensitivity prediction, the incompletely specified truth table is defined by the binary model matrix



Model matrix
Gene 1
...
Gene P
Cell line 1
Cell line 2
...
Cell line N
$x_{1,1}$
$x_{1,P}$
$x_{2,1}$
$x_{2,P}$
$x_{N,1}$
$x_{N,P}$
$X \in \{0,1\}^{N \times P}$
Response
Drug
Weight
$d \in \{0,1\}^N$ $w \in \mathbb{R}^N$
LOBICO ILP
$s_{pk} = \begin{cases} 1, \text{ if } p \text{ is selected for term } k \\ 0, \text{ otherwise} \end{cases}$
$s'_{pk} = \begin{cases} 1, \text{ if neg. of } p \text{ is selected for term } k \\ 0, \text{ otherwise} \end{cases}$
$s_{11}, \dots, s_{P1}$
$s_{1K}, \dots, s_{PK}$
$s'_{11}, \dots, s'_{P1}$
$s'_{1K}, \dots, s'_{PK}$
$\wedge$
$t_1 = t_{11} \dots t_{N1}$
$t_K = t_{1K} \dots t_{NK}$
$\vee$
$d'_1 \dots d'_N$
$\min \sum_{\forall n: d_n = 0} w_n \cdot d'_n - \sum_{\forall n: d_n = 1} w_n \cdot d'_n$
$\forall k \in \{1,\dots,K\} : \sum_{p=1}^{P} (s_{pk} + s'_{pk}) \le M$
$\forall p \in \{1,\dots,P\} : \forall k \in \{1,\dots,K\} : s_{pk} + s'_{pk} \le 1$
$\forall n \in \{1,\dots,N\} : \forall k \in \{1,\dots,K\} :$
$P \cdot t_{nk} \le z \le t_{nk} + P - 1$
$z = \sum_{\forall p: x_{np} = 1} (1 - s'_{pk}) + \sum_{\forall p: x_{np} = 0} (1 - s_{pk})$
$\forall n \in \{1,\dots,N\} : d'_n \le \sum_{k=1}^{K} t_{nk} \le K \cdot d'_n$

Figure 5.1: LOBICO ILP formulation. This figure depicts the LOBICO ILP approach by Knijnenburg et al. [27]. In particular, it is shown how the model matrix **X** and the response vector **d** become integrated in the ILP formulation for the Boolean function synthesis. Moreover, the connections between the variables are visualized.

$\mathbf{X} \in \{0,1\}^{N \times P}$, which represents the P variables of N samples that should be combined in the logic formula, and the corresponding binary model output $\mathbf{d} \in \{0,1\}^N$ denoting the desired evaluation of the logic formula for all samples. Knijnenburg et al. then search for a Boolean function of a given size that best reflects the assumed underlying Boolean relationship between **X** and **d**. However, two main differences exist between classical Boolean function synthesis with ILPs and LOBICO: Knijnenburg et al. allow mismatches between the Boolean function and the response, and they weight each sample with a factor that mimics its importance. The ILP for this task can be formulated as shown in the following. As introduced earlier, let $\mathbf{X} \in \{0,1\}^{N\times P}$ be the model matrix and $\mathbf{d} \in \{0,1\}^N$ be the response vector. Here, N is the number of samples, and P is the number of features. Then, Knijnenburg et al. search for a Boolean function in disjunctive normal form (DNF). Note that any Boolean function can be expressed in DNF. A Boolean function is in disjunctive normal form if it consists of a disjunction (OR-gate) of conjunctions (AND-gate) of literals (atomic variables or their negation). To this end, let K be the number of disjunctive terms and M be the maximal number of conjunctions allowed per disjunctive term. Note that these two values are hyperparameters that must be set during model training. Furthermore, let s_{pk} denote the selection variable of a feature $p \in \{1,\ldots,P\}$ in the disjunctive term $k \in \{1,\ldots,K\}$, i.e.,

$$s_{pk} = \begin{cases} 1 \text{ if feature } p \text{ is in the } k\text{th disjunctive term} \\ 0 \text{ if feature } p \text{ is not in the } k\text{th disjunctive term} \end{cases} \tag{5.1}$$

Analogously, let $s^{'}_{pk}$ be the selection variable for the negation of the feature $p \in \{1,\ldots,P\}$ in the disjunctive term $k \in \{1,\ldots,K\}$, i.e.,

$$s^{'}_{pk} = \begin{cases} 1 \text{ if the negation of feature } p \text{ is in the } k\text{th disjunctive term} \\ 0 \text{ if the negation of feature } p \text{ is not in the } k\text{th disjunctive term} \end{cases} \tag{5.2}$$

In each disjunctive term, we may only allow either s_{pk} or s_{pk}' to be present since otherwise, the logical formula would be led ad absurdum. Thus, we need the following constraint to ensure the latter:

$$\forall p \in \{1,\ldots,P\} : \forall k \in \{1,\ldots,K\} : s_{pk} + s^{'}_{pk} \leq 1 \tag{5.3}$$

Furthermore, we require that each disjunctive term contains at most M elements:

$$\forall k \in \{1,\ldots,K\} : \sum_{p=1}^{P} (s_{pk} + s^{'}_{pk}) \leq M \tag{5.4}$$

Let now t_{nk} be a set of auxiliary variables representing the disjunctive terms $k \in \{1,\ldots,K\}$ for all samples $n \in \{1,\ldots,N\}$. In order for our constraints to depict a logical formula in DNF for each sample $n \in \{1,\ldots,N\}$, each disjunctive term needs to consist of conjunctions of literals. This can be mathematically expressed as follows using logic AND gates:

$$\forall n \in \{1,\dots,N\} : \forall k \in \{1,\dots,K\} :$$
$$P \cdot t_{nk} \leq \sum_{\forall p : x_{np}=1} (1 - s'_{pk}) + \sum_{\forall p : x_{np}=0} (1 - s_{pk}) \leq t_{nk} + P - 1 \tag{5.5}$$

Consequently, one disjunctive term $k \in \{1,\dots,K\}$ is now given by the N-dimensional vector $\mathbf{t}_k$, i.e., $\mathbf{t}_k = (t_{1k},\dots,t_{Nk})^T$. Finally, our synthesized Boolean function needs to represent the disjunction of the $\mathbf{t}_k$. Thus, we generate the N-dimensional output vector $\mathbf{d}'$ via a logic OR gate between the $\mathbf{t}_k$:

$$\forall n \in \{1,\dots,N\} : d'_n \leq \sum_{k=1}^{K} t_{nk} \leq K \cdot d'_n \tag{5.6}$$

This output vector is the prediction based on the currently used logic formula and should resemble our actual response vector $\mathbf{d}$ as closely as possible. This means that we have to find the logic formula that reduces some error function most. To this end, LOBICO minimizes the weighted sum of incorrectly inferred samples

$$\text{minimize} \sum_{\forall n : d_n = 0} (w_n \cdot d'_n) - \sum_{\forall n : d_n = 1} (w_n \cdot d'_n). \tag{5.7}$$

If the true response d_n is 0 and the prediction d'_n is 1, the error increases by w_n, while the contribution to the error is 0 if d'_n is also 0. If the true response d_n is 1 and the prediction d'_n is 0, the contribution to the error is 0, while the error decreases by w_n if d'_n is also 1. Hereby, w_n is a sample-specific importance factor, which allows us to put emphasis on particular cell lines. Given the continuous response vector $\mathbf{y}$, these weights correspond to the absolute distance of a particular value y_n from the threshold t and can be calculated as

$$w_n = \frac{|y_n - t|}{2 \cdot \sum_{\forall m : d_m = d_n} |y_m - t|} \tag{5.8}$$

In the following, we call this weight function *linear*. We provide an overview of the variable chaining of LOBICO in Figure 5.1.
Apart from this core ILP formulation, Knijnenburg et al. provide two constraints

for controlling the sensitivity and specificity of the derived Boolean formula. More specifically, these constraints exclude any Boolean function as a solution that does not satisfy a predefined minimal requirement on the sensitivity or specificity, which can, however, ultimately lead to infeasibility of the ILP. Let Sens_{min} be the desired minimal sensitivity and Spec_{min} be the desired minimal specificity. The two constraints are given by

$$\sum_{\forall n:d_n=1} (w_n \cdot d_n^{'}) \geq \text{Sens}_{\text{min}} \cdot \sum_{n=1}^{N} (w_n \cdot d_n) \tag{5.9}$$

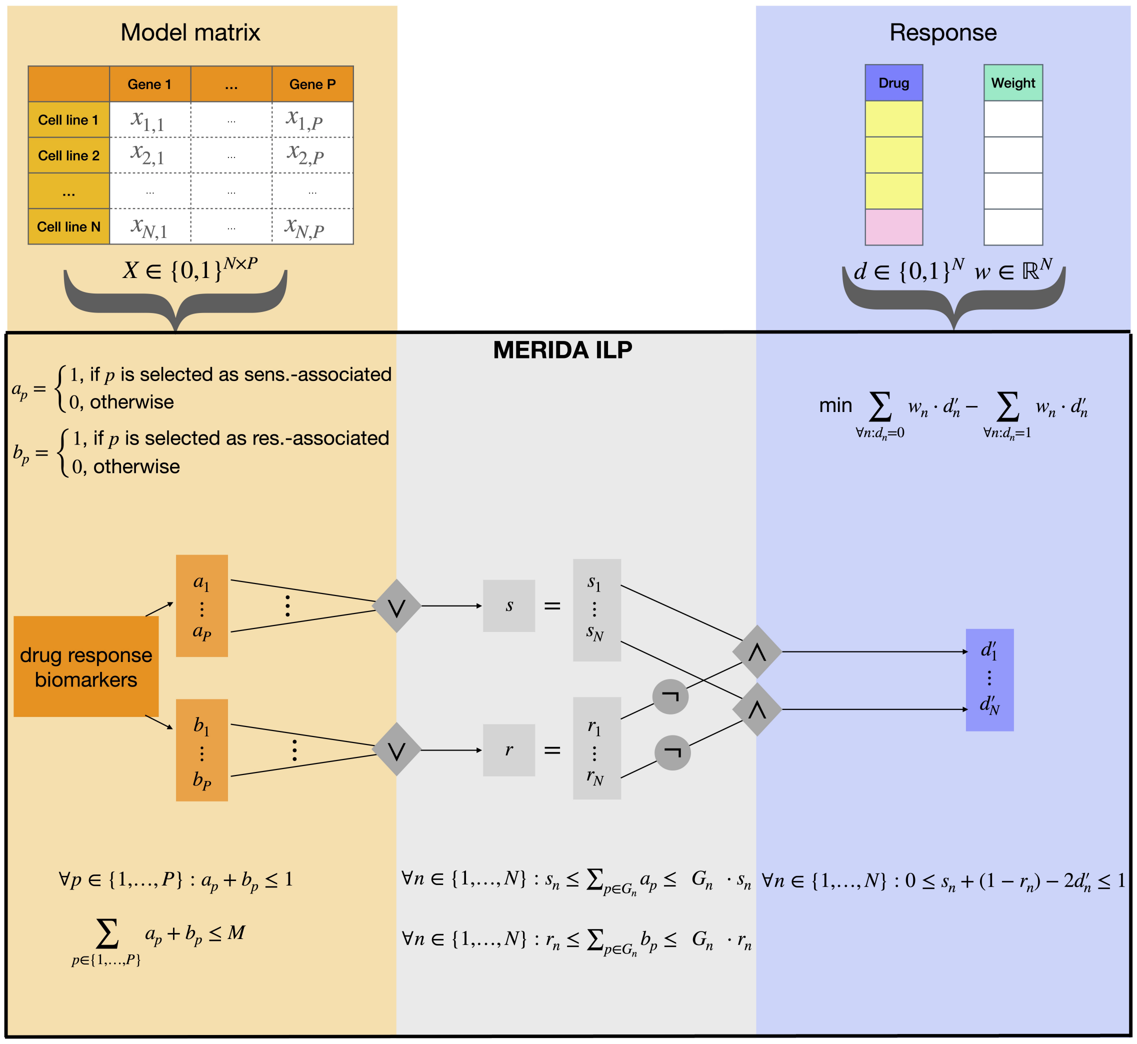
and

$$\sum_{\forall n:d_n=0} (w_n \cdot (1 - d_n^{'})) \geq \text{Spec}_{\text{min}} \cdot \sum_{n=1}^{N} (w_n \cdot (1 - d_n)) \tag{5.10}$$

If all sample-specific weights equal 1, $\sum_{n=1}^{N}(w_n \cdot d_n)$ is the number of positives and $\sum_{\forall n:d_n=1}(w_n \cdot d_n^{'})$ the number of true positives. Likewise, $\sum_{n=1}^{N}(w_n \cdot (1 - d_n))$ is the number of negatives and $\sum_{\forall n:d_n=0}(w_n \cdot (1 - d_n^{'}))$ is the number of true negatives. Hence, the constraints exactly represent a minimum requirement for sensitivity and specificity, respectively. If the sample-specific weights are set differently, Knijnenburg et al. refer to these equations as continuous versions of sensitivity and specificity that factor in the relative importance of samples.

5.1.2 MERIDA

Similar to LOBICO, the ILP formulation of MERIDA aims to deliver well-interpretable logic rules explaining drug sensitivity and resistance. While Knijnenburg et al. did not restrict their rules to adhere to a specific form, traversing the full space of possible logic combinations becomes extremely computationally expensive, which limits the applicability of the approach. Thus, we propose a modified version of the ILP that tremendously reduces the runtime, allowing for the investigation of larger input feature sets and larger rules. Moreover, we enable the integration of a priori knowledge in form of known biomarkers of drug response, which increases the comprehensiveness of our rules even more.
Briefly, we model the prediction vector $\mathbf{d}^{'}$ as a logic combination of only two terms.



Model matrix
Gene 1
Gene P
Cell line 1
Cell line 2
Cell line N
$X \in \{0,1\}^{N \times P}$
Response
Drug
Weight
$d \in \{0,1\}^N$ $w \in \mathbb{R}^N$
MERIDA ILP
$a_p = \begin{cases} 1, \text{if } p \text{ is selected as sens.-associated} \\ 0, \text{otherwise} \end{cases}$
$b_p = \begin{cases} 1, \text{if } p \text{ is selected as res.-associated} \\ 0, \text{otherwise} \end{cases}$
$\min \sum_{\forall n: d_n = 0} w_n \cdot d'_n - \sum_{\forall n: d_n = 1} w_n \cdot d'_n$
drug response biomarkers
$\forall p \in \{1,\ldots,P\} : a_p + b_p \leq 1$
$\sum_{p \in \{1,\ldots,P\}} a_p + b_p \leq M$
$\forall n \in \{1,\ldots,N\} : s_n \leq \sum_{p \in G_n} a_p \leq G_n \cdot s_n$
$\forall n \in \{1,\ldots,N\} : r_n \leq \sum_{p \in G_n} b_p \leq G_n \cdot r_n$
$\forall n \in \{1,\ldots,N\} : 0 \leq s_n + (1 - r_n) - 2d'_n \leq 1$

Figure 5.2: MERIDA ILP formulation. In this figure, we depict MERIDA using the same schematic display format we employed for the LOBICO ILP in Figure 5.1 . For MERIDA, we show how the model matrix **X**, the response vector **d**, and the known biomarkers of drug response are integrated in the ILP formulation. Moreover, the connections between the variables are visualized.

One term (**s**) represents features that may cause or explain drug sensitivity, and one term (**r**) represents features that may cause or explain drug resistance. Within each term, we connect the features via a disjunction, while the two terms are joined using a conjunction between the sensitivity-associated term and the negation of the resistance-associated term. In the ensuing section, we present the constraints and the objective function of the corresponding ILP. In Figure 5.2, we visualize MERIDA using the same schematic display format we employed for LOBICO in Figure 5.1.

Let $\mathbf{X} \in \{0,1\}^{N\times P}$ once again be the model matrix and $\mathbf{d} \in \{0,1\}^N$ the binary response vector. For MERIDA, we decided to synthesize only Boolean formulas adhering to the scheme described above. The logic rule should only consist of two terms: one term representing all sensitivity-associated alterations and one representing all resistance-associated alterations. Consequently, we need two types of selection variables that indicate whether a feature $p \in \{1,\dots,P\}$ is part of the former or latter set, i.e.

$$a_p = \begin{cases} 1 & \text{if feature } p \text{ is selected as sensitivity-associated} \\ 0 & \text{else} \end{cases} \tag{5.11}$$

and

$$b_p = \begin{cases} 1 & \text{if feature } p \text{ is selected as resistance-associated} \\ 0 & \text{else} \end{cases} . \tag{5.12}$$

We ensure that a feature cannot be part of both sets simultaneously by using the following constraint

$$a_p + b_p \leq 1, \quad \forall p \in \{1,\dots,P\}. \tag{5.13}$$

Moreover, we restrict the total number of features to be selected to M:

$$\sum_{p\in\{1,\dots,P\}} a_p + b_p \leq M. \tag{5.14}$$

Note that M is the only hyperparameter of our model. The selection variables now need to be combined into the logic formula. Firstly, we join the putative sensitivity-associated features a_p into the binary vector $\mathbf{s}$ of dimension N via a logic OR

$$s_n \leq \sum_{p\in G_n} a_p \leq |G_n| \cdot s_n, \quad \forall n \in \{1,\dots,N\} . \tag{5.15}$$

Here, G_n is the set of features that is altered in cell line n. The logic OR ensures that s_n is equal to 1 iff at least one sensitivity-associated alteration is contained in G_n, i.e. $\sum_{p\in G_n} a_p \geq 1$. More specifically, the lower boundary ensures that s_n cannot be equal to 1 if $\sum_{p\in G_n} a_p = 0$ and the upper boundary ensures that s_n cannot be

equal to 0 if $\sum_{p \in G_n} a_p \geq 1$
Analogously, we join the putative resistance-associated features b_p into the binary vector $\mathbf{r}$ of dimension N:

$$r_n \leq \sum_{p \in G_n} b_i \leq |G_n| \cdot r_n, \quad \forall n \in \{1, \ldots, N\} \tag{5.16}$$

Finally, assuming that a cell line is only sensitive to a drug if a sensitivity-inducing but no resistance-inducing alteration is present, we combine the vectors $\mathbf{s}$ and $\mathbf{r}$ to the binary prediction vector $\mathbf{d}^{'}$ by a logic AND between the vector components of $\mathbf{s}$ and the negation of $\mathbf{r}$:

$$0 \leq s_n + (1 - r_n) - 2 \cdot d'_n \leq 1, \quad \forall n \in \{1, \ldots, N\}. \tag{5.17}$$

Here, d'_n will be equal to 1 iff the cell line n is predicted to be sensitive and 0 otherwise. Finally, we then employ the error function from LOBICO as objective function to obtain the prediction vector $\mathbf{d}^{'}$ that resembles $\mathbf{d}$ most

$$\min \sum_{\forall n : d_n = 0} w_n \cdot d'_n - \sum_{\forall n : d_n = 1} w_n \cdot d'_n \tag{5.18}$$

While LOBICO uses only linear weights, we also consider quadratic and cubic weight functions that more strongly emphasize the distance of the continuous drug response value y_n from the binarization threshold t

$$w_n = \frac{|y_n - t|^v}{2 \cdot \sum_{\forall y_m : y_m = y_n} |y_m - t|^v} \tag{5.19}$$

with $v \in \{2, 3\}$. The usage of a quadratic or cubic weight function can be advantageous. In our experiments (cf. Section 5.4.1), we show that these alternative weighting schemes significantly reduce the runtime.

Inclusion of drug response biomarkers

We already discussed (cf. Chapters 3 and 4) that data from multi-omics measurements are usually very high-dimensional while the number of samples is comparatively low. In computer science, this problem is known as the curse of dimensionality and severely impacts the search for relevant alterations using machine learning. To counteract this issue, we decided to implement a literature-driven feature selection and annotation strategy in front of our method, the details of which will be presented in-depth in Section 5.2. Briefly summarized, we filter the alterations of the used data types for known cancer genes and annotate them with their assumed oncogenicity and known relationships to drug response. In this paragraph, we focus on why the annotation with known biomarkers of drug response is especially desirable and how easily this can be accomplished for MERIDA or similar logic modelling approaches. For various alterations, it is already known whether they confer sensitivity or resistance to a certain drug [25]. However, other (yet unknown) alterations may annihilate their expected effect. Apart from that, many sensitivity or resistance determinants are still assumed to be unknown, especially rarer variants. Phenomena like these severely impede the search for relevant features based on purely computational methods, e.g., ML techniques, given the high dimensionality of the feature space. Often, it is even barely possible to rediscover known biomarkers. By directly incorporating known drivers of drug response, an algorithm can more easily learn patterns truly related to drug response.
For MERIDA, integrating known sensitivity- or resistance biomarkers is fairly simple: the a priori knowledge imposes constraints on the corresponding selection variables, e.g., if it is known that a mutation is responsible for sensitivity to the investigated drug, the corresponding selection variable a_p can be set to 1. If it is known that a feature is related to drug resistance, the corresponding selection variable b_p can be set to 1. Importantly, allowing the incorporation of a priori biomarkers in this form enables an iterative application of MERIDA, where knowledge generated in previous runs of MERIDA can serve as a priori knowledge for the next run.

5.2 Data preparation

For the analyses presented in this chapter, we employed various publicly accessible resources: The cancer cell line data was downloaded from the Genomics of Drug Sensitivity in Cancer (GDSC) website [19]. Here, we use the pre-processed mutation data (whole exome sequencing with Agilent Sure- SelectXT Human All Exon 50Mb bait set) from Release 6.1, the copy number data (Affymetrix SNP6.0 Array) from Release 6.1, the gene expression data (Affymetrix Human Genome U219 Array) from Release 7.0, and drug data from Release 8.0, which can be divided into two sub-datasets depending on the experimental assay type (GDSC1: Syto60 and resazurin assay, GDSC2: CellTiterGlo). As briefly mentioned in the previous paragraph, we developed and implemented a literature-driven feature selection and annotation strategy, for which we leveraged information from established cancer-related databases, i.e., IntOGen (Release 2016.5) [22], COSMIC (Version v86) [23], CIViC (Release 01-Oct-2018) [24], OncoKB (Version v1.16) [25], and CGI (Version 2018/01/17[26], as well as curated information from a publication by Sanchez-Vega et al. [206]. In the following, we describe how these different data types contribute to the generation of interpretable logic models for drug sensitivity prediction.

5.2.1 Drug response vectors

In this chapter, we concentrate our analyses on mTOR pathway inhibitors since the mTOR pathway is known to play a pivotal role in cancer development and progression [206, 68] and, thus, it is already relatively well investigated. In particular, for various mTOR inhibitors, there is already knowledge on drug response biomarkers (cf. Appendix Tables B.1 - B.11), which we can directly integrate into our logic rules and improve the interpretability of MERIDA. In total, we investigated 37 different mTOR inhibitors. If a drug was screened for GDSC1 and GDSC2, we decided to use the newer GDSC2 database if there were sufficient cell lines available ($>$700 cell lines). Since the presented models rely on binary data to generate Boolean rules for drug sensitivity prediction, we have to binarize the given continuous logarithmized IC50 values from the GDSC. By applying the heuristic outlier procedure described in Section 5.1.1, we obtain drug-specific thresholds that can be used to construct one binary response vector per drug, specifying for each cell line whether it is sensitive (1) or resistant (0). In Figure 5.3, the resulting sensitive-to-resistant ratio is depicted for all drugs in GDSC1. The analogous plot for GDSC2 can be found in the Appendix B.1. It is clearly visible that for most drugs the resistant cell lines outnumber the sensitive cell lines, i.e., almost all drug data sets exhibit a high class imbalance. On average, 10% of cell lines are sensitive to a particular drug treatment. To investigate the performance of MERIDA when there is almost no class imbalance, we decided to analyze the four drugs with the highest number of sensitive cell lines in GDSC1 and GDSC2 (the p53(R175) mutant reactivator NSC319726 (GDSC1), the rRNA synthesis inhibitor CX-5461 (GDSC1), the selective PARP1/2 inhibitor Niraparib (GDSC2), and the PARP inhibitor Talazoparib (GDSC2)) in addition to the 37 mTOR inhibitors. The drug-specific threshold cannot only be used to partition the data but can help retain some of the continuous information from the IC50 values in the binarized data. As explained in Section 5.1.1, Knijnenburg et al. suggest employing the distance from the threshold as weights reflecting the importance measure of the cell lines. Since the sum of the weights for the sensitive cell lines is equal to the sum of the weights for the resistant cell lines, this can be interpreted as a direct countermeasure against the class imbalance.

5.2.2 Input feature matrix

Generally, the binary input feature matrices of the drugs consist of gene expression, copy number alteration, and mutation features. Each feature is represented by a binary vector indicating whether each considered cell line is affected by the corresponding aberration (1) or not (0). We constructed the input feature matrices taking into account the mentioned cancer-related databases containing information on oncogenicity and known association to drug response of alterations. We will now describe how we integrate this knowledge.

Gene expression features: The gene expression features of the matrix are obtained by performing the following steps:

- From the available genes of the GDSC gene expression matrix, we consider only those that belong to the IntOGen cancer driver gene list (459 genes).

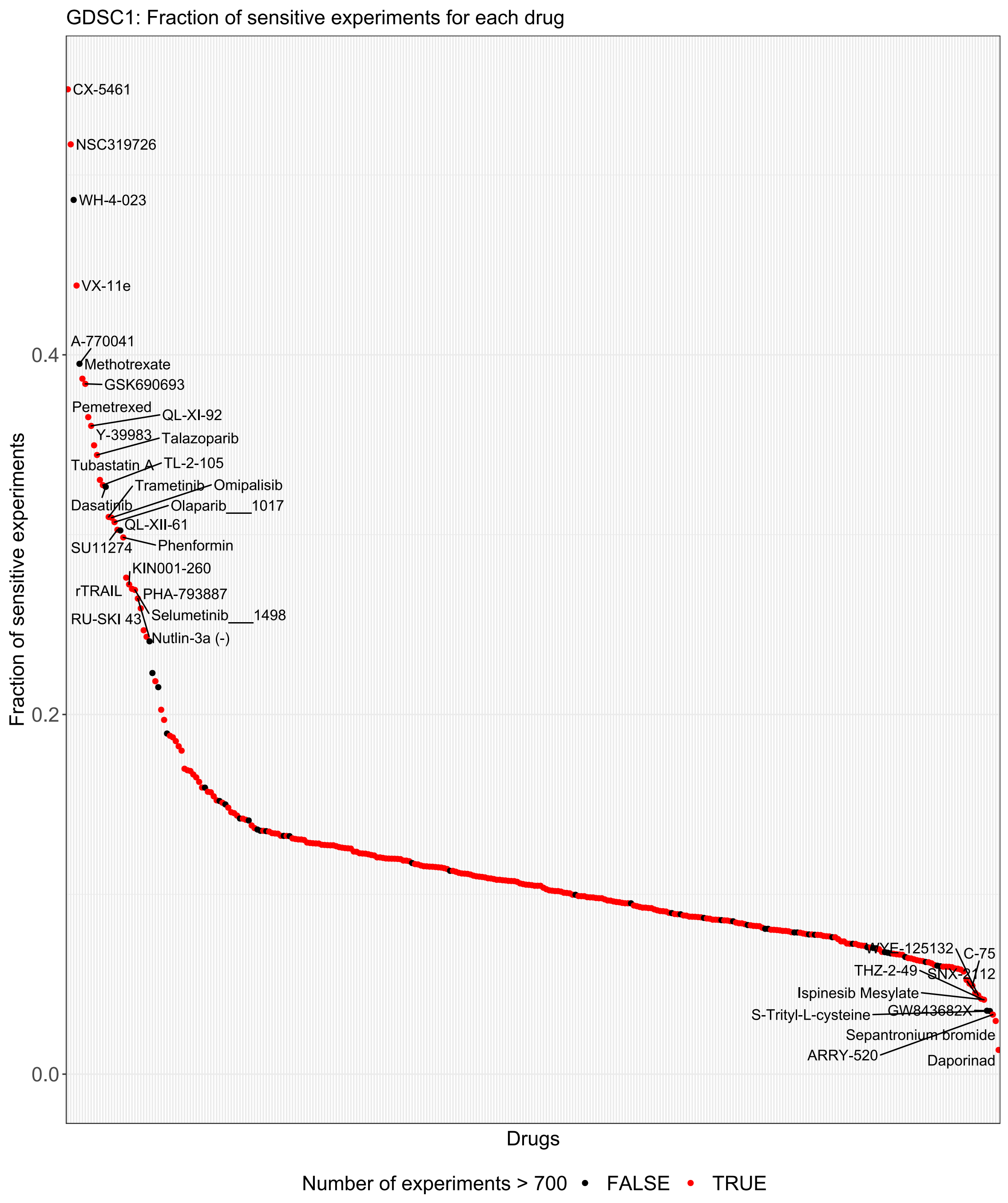


Figure 5.3: Sensitive-to-resistant ratio in GDSC1. In this figure, we depict the fraction of sensitive cell lines for each drug from GDSC1 when binarizing the logarithmized IC50 values with the procedure suggested by Knijnenburg et al. [27]. The coloring indicates whether a drug has been screened with more than 700 cell lines (red) or not (black).

- For each of the selected genes, two binary vectors are added to the feature matrix: one binary vector that specifies whether the gene is up-regulated in the considered cell lines (1) or not (0) and one binary vector that specifies whether the gene is down-regulated (1) or not (0). The binarization was accomplished by the calculation of gene-wise z-scores and selection of the top 5% up- and down-regulated cell lines per gene. Note that we used only the training cell lines to determine the sample mean and standard deviation and also used these values to obtain the z-score for the test set.

In summary, we obtained up to 918 gene expression features.

Copy number features: Similar to the gene expression data, we also filter the GDSC CNV data: we consider only genes that are listed in the copy number driver list by Sanchez-Vega et al. [206] (140 genes). For each of these genes, two binary feature vectors are added to the feature matrix: one vector that represents for each cell line whether a copy number gain is present (1) or not (0) and an analogous vector for copy number loss. Hence, this leads to a CNV feature list with up to 280 elements.

Mutation features: The GDSC mutation data is given as point mutations of genes within the cell lines. However, using each point mutation as a feature would lead to a sparse, high-dimensional matrix, posing a problem to ML methods. Thus, we consider only genes contained in the IntOGen driver list (459 genes) and gather all mutations assigned to these genes in a filtered mutation list. For each gene in this list, we combine mutations with similar functional annotations into four composite vectors representing four oncogenicity states: oncogenic gain-of-function, oncogenic loss-of-function, neutral, and status unknown. We use the annotations from CGI, CIViC, COSMIC, and OncoKB to define and derive the composite features by performing the steps described below:

- We annotate the alterations from the filtered mutation list with an oncogenicity status from the four mentioned states.

- Afterwards, all mutations with the same annotation in one particular gene are merged into one binary feature vector. Consequently, the resulting feature vector contains a 1 iff at least one of the alterations is present in a cell line and 0 otherwise.

By doing so, we consider up to 1836 mutation features. If frameshift or truncating mutations present in our samples have not been annotated in one of the databases, we add these as extra features to our input matrix since we consider them deleterious.

5.2.3 Integration of drug response biomarkers

In the previous paragraphs, we outlined how to integrate the oncogenicity of alterations into the input feature matrix. In this paragraph, we explain how to incorporate available prior knowledge on sensitivity or resistance biomarkers for a particular drug. To this end, we have to perform additional processing steps for each drug separately. After filtering the CNV and mutation data using the list by Sanchez-Vega et al. and the IntOGen driver list, respectively, we define a sensitivity status for the remaining alterations. This sensitivity status can have one of the following states: sensitive (alteration is predictive of a positive drug response), not sensitive (alteration has been shown not to be predictive for a positive drug response), resistant (alteration is predictive of negative drug response), not resistant (alteration has been shown not to be predictive for a negative drug response) and status unknown (cf. Appendix Tables B.1 - B.11 for an overview of the alterations with known drug response association per drug). For all alterations with unknown status, we construct the feature matrix as described in the previous section. For all other alterations, we can, however, define and construct composite features, i.e., we merge all alterations with the same state into a binary feature vector. This vector contains a 1 iff at least one of the alterations with the same state is present in the considered cell line and 0 otherwise. Suppose at least one of the databases more generally stated that oncogenic loss-of-function, oncogenic gain-of-function, truncating mutations, or frameshift mutations of a gene are linked to drug sensitivity or resistance. In that case, we add all alterations of a sample with a matching label in the corresponding gene to the composite features. Note that all features contributing to a drug response composite feature are not integrated into the oncogenicity-associated composite features.

5.2.4 Model training

In this chapter, we showcase the capabilities of MERIDA. To this end, we generally focus on a thorough comparison of MERIDA to its direct competitor, LOBICO, with respect to runtime, statistical performance, and selected biomarkers. However, we also benchmark MERIDA against baseline classifiers, i.e., k nearest neighbours (KNN) [207] and random forests (RF) [181], regarding statistical performance. For most experiments, we use the data and data processing steps as outlined above (cf. Section 5.2). However, the runtime analysis was conducted on a simplified data set solely prepared and used for this purpose. In the following, we describe the preparation and training procedure of all methods in detail.

Preparation of runtime analysis

We performed the runtime benchmarking between MERIDA and LOBICO on the Rapamycin drug data set and the mutation data of GDSC Version 6.1. The Rapamycin data set of GDSC Version 6.1 comprises IC50 values for roughly 350 cell lines that we binarized using the LOBICO threshold provided in the publication by Iorio et al. [19], in which this version of the GDSC data set was introduced, and LOBICO was already applied to generate logic rules explaining drug sensitivity and resistance. For this analysis, we decided to construct a simple input feature matrix containing mutation features only. To this end, we established a gene-wide mutation status, i.e. we call a gene mutated (feature value of 1) iff at least one mutation is present and not mutated (feature value of 0) otherwise. This means that we do not distinguish between different mutations or classes of mutations for this analysis. Since we wanted to assess the influence of the number of input features on the time needed to fit a model, we compiled 16 data sets with varying feature set sizes in the range from 25 to 400 features by a step size of 25. We used the IntOGen cancer driver list to filter and prioritize the genes through the frequency of mutations within a gene as provided by IntOGen [22]. We tested MERIDA and LOBICO with different hyperparameters including the three different weight functions (linear, quadratic, and cubic) and different number of input features. We performed the calculations on a compute server with four Intel(R) Xeon(R) CPU E5-4657L v2 processors with 2.40GHz clock rate. IBM ILOG CPLEX Optimization Studio V12.6.2 for C++ was employed to formulate and solve the ILP. CPLEX was run using 32 cores and a deterministic parallel mode. Each experiment was repeated 10 times if not prohibited by a high runtime.

Preparation of statistical performance and biomarker analysis

Except for the runtime analysis, we benchmarked MERIDA using the data processing and annotation scheme described in Section 5.2. As mentioned in that section, we focused our analyses on mTOR pathway inhibitors since, for many of them, comprehensive knowledge of drug response biomarkers exists, which we can directly incorporate into MERIDA. To assess the effect of these biomarkers on prediction performance, we generally investigated three analysis settings:

- Setting 1: the available drug response biomarkers are not included in the input feature matrix, i.e., the matrix is constructed as if there was no biomarker information
- Setting 2: the available drug response biomarkers are part of the input feature matrix as a specific composite feature (cf. Section 5.2.3), and the value of the corresponding ILP feature variable is fixed in advance of solving the ILP (cf. Section 5.1.2)
- Setting 3: the available drug response biomarkers are part of the input feature matrix, but the value of the corresponding feature variable is determined by the (ILP) model

We can merely carry out analyses for Setting 1 if there is no a priori biomarker knowledge for a particular drug. We report the statistical performance during cross-validation (CV) and on a test set. To this end, we divide each drug data set consisting of the input feature matrix and the binarized drug response into a training (80%) and a test (20%) set by performing a stratified sampling with respect to the cancer site. This partition into training and test set is identical for all methods. On the training set, we then conduct a 5-fold CV to determine the best hyperparameters. Note that the CV fold division can vary between methods. We then trained a final model on the complete training data employing the best hyperparameter combination from CV and applied this model to the test set.

5.3 Implementation and data deposition

The implementation of LOBICO and MERIDA was accomplished with C++14 and a CMake-based build system. Using standard software engineering techniques, we kept the implementation modular so that it could be easily extended with novel ILPs for Boolean function synthesis. We employed IBM ILOG CPLEX Optimization Studio V12.6.2 for C++ to formulate and solve the ILPs. The resulting C++ program can be called from the console. It requires a simple configuration file in txt format containing model hyperparameters and file paths as input and delivers a custom txt file with Boolean rules and performance information as output. We deposited the C++ code alongside example files on our publicly available GitHub: `https://github.com/unisb-bioinf/MERIDA.git`.
Apart from the core ILP models, we also filed the implementation of the feature annotation and selection strategy and the implementation of the baselines methods on this GitHub repository. The feature annotation and selection strategy was implemented with python. The heuristic outlier procedure for binarising the drug response values is written in R, and the baseline methods were implemented using the caret package [208].

5.4 Results

The number of already known cancer-associated genetic and molecular variants is enormous [22, 23, 24, 25]. Apparently, they not only determine cancer development and progression but also influence therapy responsiveness. Yet, due to the complexity of this disease, many contributing factors and their interactions still remain to be elucidated. With MERIDA, we pursued the goal of developing an approach able to handle this variety of features in order to discover logical rules that may serve as a basis for hypothesis generation concerning the anti-cancer drug response

of tumours. While MERIDA is similar to LOBICO, our method has a significantly reduced runtime, which allows it to handle large feature sets and construct more comprehensive rules. In addition to that, MERIDA directly accounts for known drug response biomarkers if desired.
In the following sections, we will first discuss the runtime advantages of our method in comparison to LOBICO. Then, we will examine the statistical performance of the methods. Here, we also show that MERIDA performs significantly better than baseline machine learning methods such as random forests and k-nearest neighbors. Lastly, we show that MERIDA is able to identify biomarkers for drug sensitivity.

5.4.1 Runtime analysis

At first, we applied LOBICO and MERIDA to the specially prepared runtime data sets to compare their runtime with respect to the number of input features and varying model hyperparameters. As model hyperparameters, we investigated the three different weight functions as well as the maximally allowed number of selectable features, i.e., the parameter M of MERIDA (cf. Section 5.1.2) and the parameters K and M of LOBICO (cf. Section 5.1.1). In Figure 5.4, we exemplarily plotted the results of MERIDA models with $M \in \{2, 4, 8\}$ compared to a 4-feature sized model ($K = 2, M = 2$) of LOBICO, which is the largest model with both $M > 1$ and $K > 1$ that we could solve in a reasonable amount of time.
In Figure 5.4, we can observe that the runtime of both MERIDA and LOBICO is dependent on the used weight function. The original linear weight function of LOBICO consistently has the highest runtime for all tested parameter combinations. For a 4-feature-sized model ($M = 1, K = 4$, $M = 4, K = 1$, $M = 2, K = 2$), the runtime of LOBICO can, on average, across all runtime data sets be reduced by a factor of 2.17 and 3.47 by employing a quadratic or cubic weight function, respectively. For MERIDA, the use of a quadratic or cubic weight function is also advantageous. MERIDA with a quadratic or cubic weight function is, on average, 1.35 and 1.65 times faster than the linear version. The direct comparison between LOBICO and MERIDA furthermore reveals that the runtime of MERIDA is not only rising more slowly with an increasing number of input features but is also considerably lower. For the linear weight function, for example, MERIDA can obtain a 4-feature sized model ($M = 4$) 3.61 times faster than LOBICO's $K = 1, M = 4$ model, 71.79 times

faster than LOBICO's $K = 4, M = 1$ model, and 641.97 times faster than LOBICO's $K = 2, M = 2$ model on average across the different input matrix sizes. By using the quadratic or cubic function for MERIDA, further speed-ups up to a factor of 1147.6 can be achieved. The most extreme values for speed-up can be observed when comparing LOBICO's $K = 2, M = 2$ model (linear weight function) with MERIDA's $M = 4$ model (cubic weight function) for the input matrix size of 400 features. Here, a speed-up factor of 5775 could be observed. To conclude, MERIDA can consider more features in the input matrix and construct larger models. Moreover, since MERIDA has only one hyperparameter (M), fewer models need to be fit during tuning in comparison to LOBICO with two hyperparameters (K and M).
For MERIDA, we additionally analyzed whether the runtime can be improved by iteratively increasing the model size instead of fitting a model of a particular size in one shot. The details of this analysis can be found in the Appendix B. Briefly, we perform the iterative extension of models as described in the following: we first select the best model among models with $M \in \{1, \ldots, 6\}$ using Youden's J (sensitivity + specificity -1) as selection criterion. Then, we interpret the selected features as a priori biomarkers and utilize them as the basis for the next, larger model. We repeat this procedure by iteratively enlarging M by 4 until the desired model size is reached. By this iterative application of MERIDA, we can generate more comprehensive models with larger values of M significantly faster than in one shot (by a factor of 25 on average). Additionally, the resulting models deliver similar features to those generated in one shot.

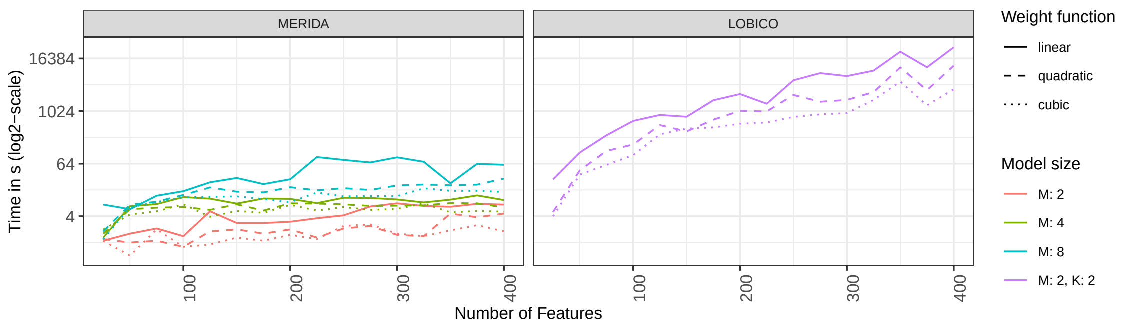


Figure 5.4: Results of runtime analysis. The runtime analysis was conducted on a small data set (350 samples) with varying feature set sizes in the range from 25 to 400 features and each experiment was repeated 10 times if not prohibited by high runtime. The figure depicts the mean runtime (plotted on a log2-scale) of LOBICO and MERIDA for different numbers of input features, weight functions, and hyperparameters.

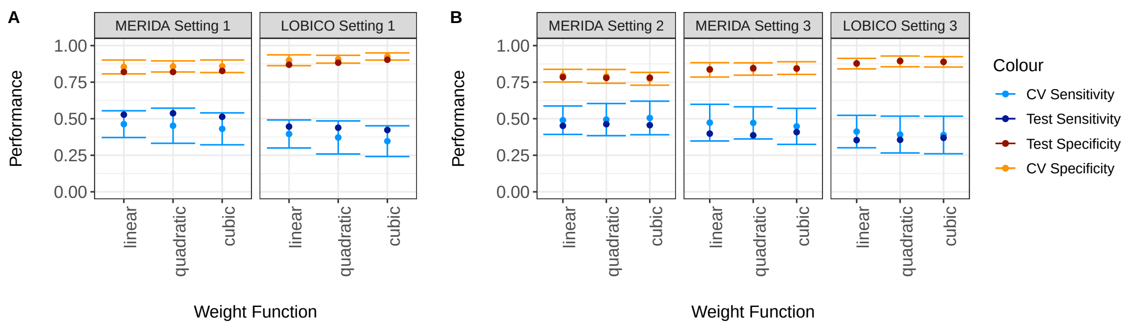


Figure 5.5: Comparison between LOBICO and MERIDA. In this figure, we present the averaged statistical performance across all drugs (A) without a priori knowledge and (B) with a priori knowledge. Shown is the mean performance and standard deviation during cross validation as well as the test error.

5.4.2 Statistical performance analysis

As outlined in Section 5.2, we mainly analyzed 37 mTOR pathway inhibitors since these are a relatively well-studied class of drugs with which we can assess the capabilities of MERIDA, particularly concerning integrating drug response biomarkers. Since there is an unfavourable ratio between sensitive and resistant cell lines of 1:10 for almost all of them, we additionally opted for investigating the four drugs with the highest number of sensitive cell lines: NSC319726, CX-5461, Niraparib, and Talazoparib. In the following, we first present a direct comparison between LOBICO and MERIDA. Afterwards, we briefly discuss the performance of MERIDA in comparison to two baseline classifiers, i.e., KNN and RF.

Unfortunately, fitting LOBICO models was extremely time-consuming even for 2-feature-sized models (cf. Section 5.4.1 and Appendix B), which is why we limited our comparison of LOBICO to MERIDA to six mTOR inhibitors (Rapamycin, Temsirolimus, Omipalisib, AZD8055, Dactolisib, and Voxtalisib) in addition to the four drugs with the best class balance. In Figure 5.5, the corresponding resulting statistical performances are depicted for the three settings introduced in Section 5.2.4: the available drug response biomarkers are not included in the input feature matrix (Setting 1), the available drug response biomarkers are part of the input feature matrix as a specific composite feature and the value of the corresponding ILP feature variable is fixed in advance of solving the ILP (Setting 2), the available drug response biomarkers are part of the input feature matrix but the value of the corresponding feature variable is determined by the ILP (Setting 3). A di-

rect comparison between MERIDA and LOBICO in Settings 1 and 3 shows that MERIDA has a higher average sensitivity across all drugs and lower average specificity. Since there is a high class imbalance in favour of the resistant cell lines for many of the drugs, the high sensitivity, i.e., correct identification of the sensitive cell lines, indicates a more balanced overall model fit for MERIDA. The inclusion of the sensitivity biomarkers (Setting 2) should, in general, improve the sensitivity of MERIDA even more. Actually, the a priori knowledge improves the average CV sensitivity while decreasing CV specificity. The average test sensitivity also slightly increases. However, when considering the test performances individually (per drug), we observed that it improved for Rapamycin, CX-5461, and Talazoparib, while it did not for Temsirolimus and Dactolisib. In Section 5.4.3, we explore why the a priori knowledge seems less beneficial than expected.
Next, we tested whether the performance can be improved when integrating the most informative features from previously calculated models as a priori knowledge into new models. Here, we added the features from the current best model as a priori knowledge to the next model to be fitted. To this end, we fixed the corresponding ILP variables to 1 and then solved the ILP again. The best model was selected based on Youden's J (sensitivity + specificity -1). We observe that by iterative repetition of this process, the performance can be improved for the tested drugs compared to the first application of our method (cf. Appendix Figure B.3). For example, the model after the fourth iteration performs best for Niraparib across all tested settings.
We also compared MERIDA to the two baseline classifiers RF and KNN on the complete set of investigated drugs (41) using the receiver operating characteristic (ROC) and sensitivity as model selection criteria during CV. In Figure 5.6, we exemplarily depict the performance for RF with sensitivity as the model selection criterion. The corresponding plot using the ROC as a selection criterion can be found in Appendix Figure B.10. Moreover, analogous plots for KNN are also shown in Appendix B. Generally, our results for RF and KNN confirm that standard machine learning approaches do not account for the class imbalance without specific countermeasures such as upsampling or sample-specific weighting. We will further delve into this topic in Chapter 6. Clearly, MERIDA addresses the class imbalance issue by calculating sample-specific weights. By comparing the results of KNN and RF to those for MERIDA shown in Figures 5.7 and 5.8, it can, thus, be seen that

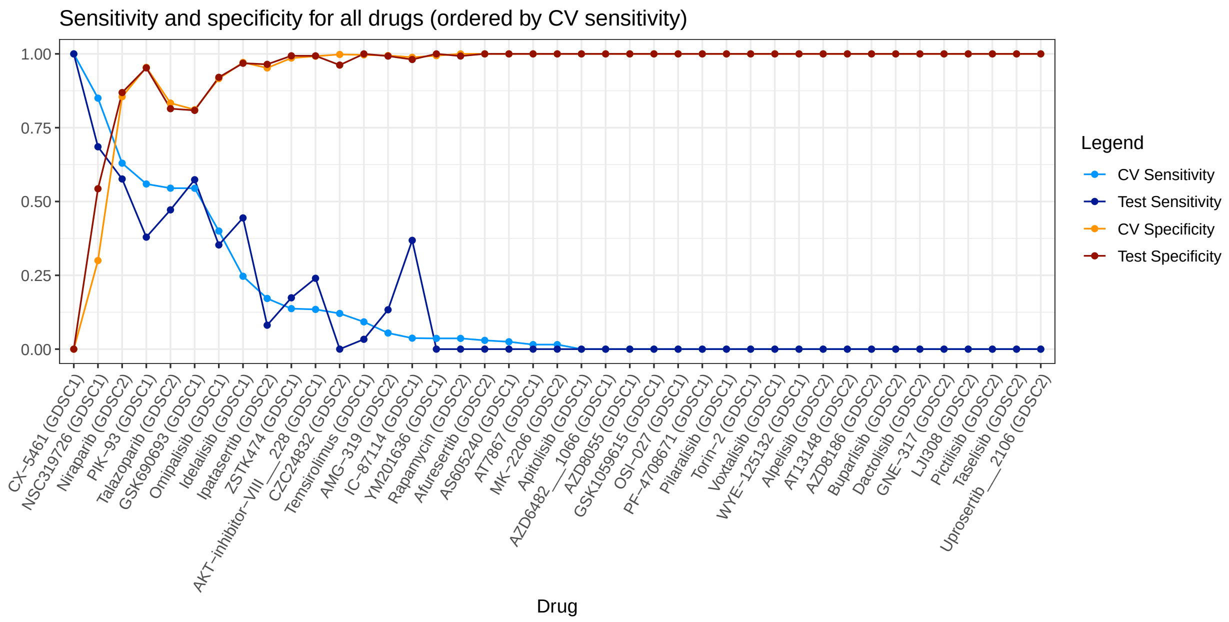


Figure 5.6: RF performance using sensitivity as selection criterion. This figure depicts the statistical performance of the best RF model in Setting 1 for all 41 drugs using sensitivity as selection criterion.

MERIDA outperforms RF and KNN. Regarding the different weight functions, we could not identify one with the overall best performance. However, on average, the linear weight function generates the most balanced fits, and the cubic function generates the least balanced ones. Considering the fact that both the cubic and quadratic weight functions usually have a tremendously lower runtime, we argue that the quadratic weight function should be preferred.
The analyses thus far dealt with the full pan-cancer data set from the GDSC. Since it is well-known that the cancer type plays a role in drug response, we also carried out analyses for sub-groups of cell lines. When dividing the GDSC data set into single sites, many of them will be relatively small, exacerbating the curse of dimensionality. Therefore, we opted for two groups of cell lines with a considerable amount of cell lines: haematological (i.e., leukaemia, lymphoma, and myeloma) and non-haematological cancer cell lines. The former comprises 160 cell lines, and the latter comprises approximately 700 cell lines. After dividing the two groups, we prepared the data sets as described in Section 5.2.4. As expected, the statistical performance decreased for both groups in comparison to the pan-cancer analysis

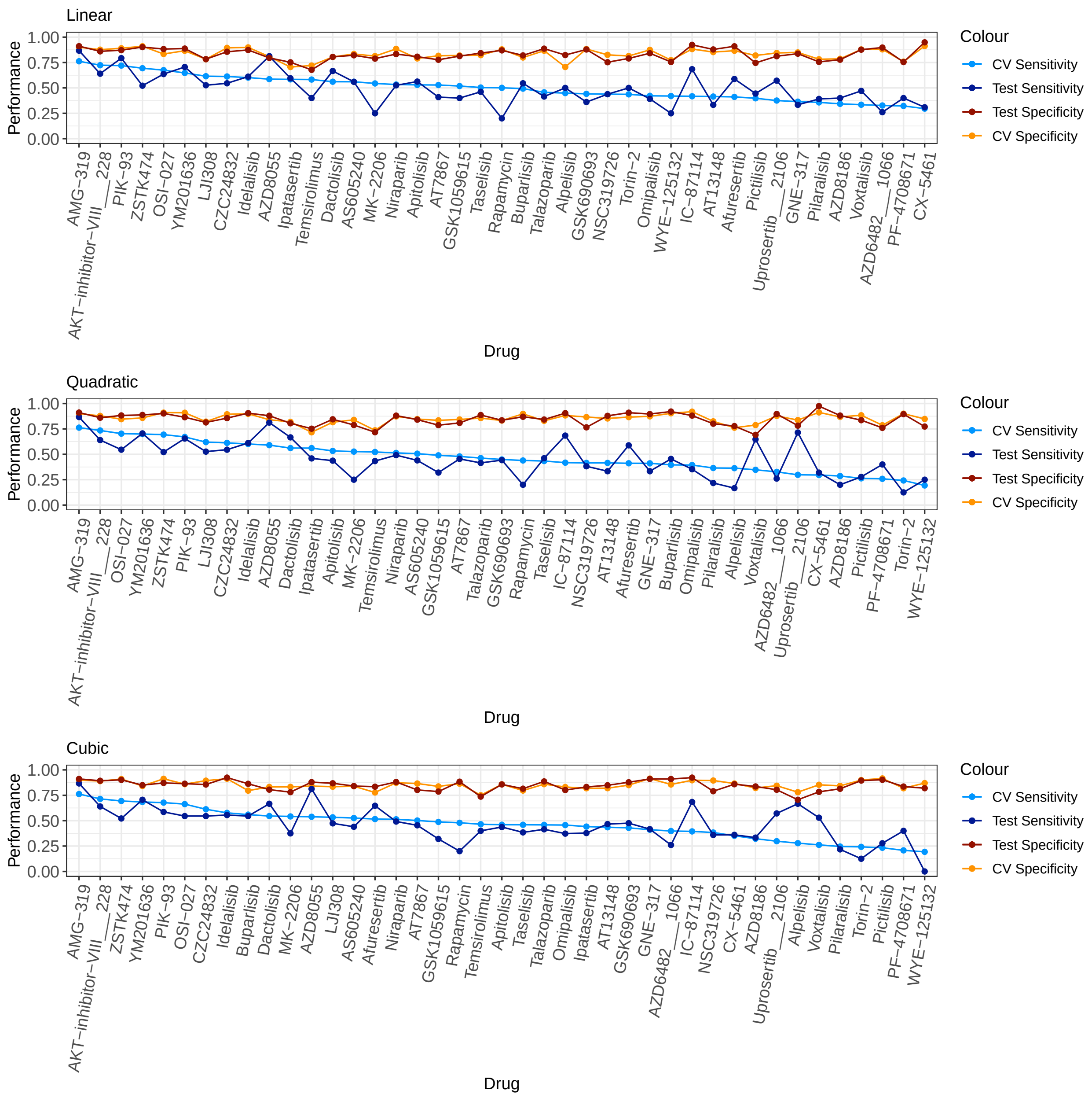


Figure 5.7: MERIDA performance in Setting 1 for all 41 drugs. This figure depicts the statistical performance of the best MERIDA model in Setting 1 for all 41 drugs using the three different weight functions.

(cf. Appendix B), a phenomenon that may be attributed to the reduced number of cell lines.

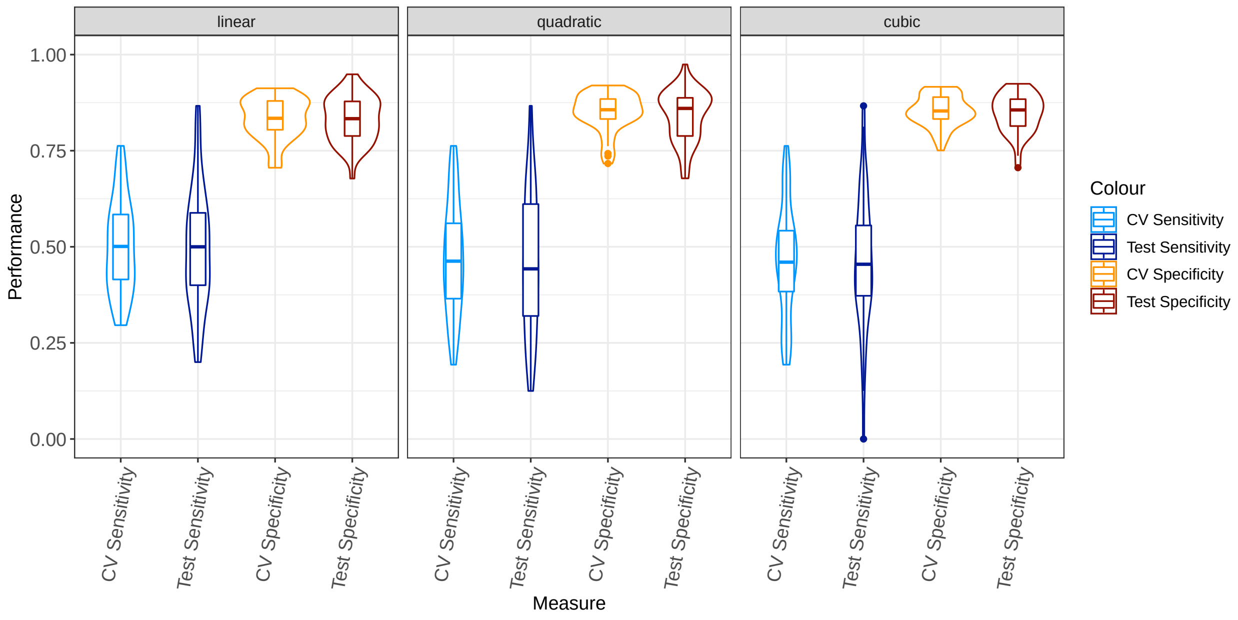


Figure 5.8: Averaged MERIDA performance in Setting 1. This figure depicts the averaged statistical performance of the best MERIDA model in Setting 1 for all 41 drugs using the three different weight functions.

5.4.3 Selected biomarkers

Lastly, we wanted to assess the relevance of the selected features of MERIDA. To this end, we compare them to the feature sets generated by LOBICO for the 10 investigated drugs in the previous section and then provide an interpretation in the corresponding biological context.
When analyzing the similarity of the selected feature sets between LOBICO and MERIDA for each drug separately, it becomes apparent that these sets significantly resemble each other (see Appendix Figure B.29 and Table B.25, Fisher's p-value < 0.05). However, since MERIDA allows larger models, it detects additional biomarkers that may influence drug sensitivity. We also analyzed the similarity of the models between the different weight functions. We observed that the selected feature sets are very similar (cf. Appendix Figures B.30 and B.31 and Tables B.23 and B.24), which reinforces the formerly given advice to opt for the quadratic weight function when performing an analysis.
To gain insight into the biological relevance of the selected features, we started by simultaneously considering the resulting models of all mTOR pathway inhibitors. One sensitivity feature consistently selected across almost all mTOR inhibitors is

low expression of the tight junction protein 1 (TJP1), a member of the membrane-associated guanylate kinase (MAGUK) family of proteins. It plays a vital role in cell-cell communication, and it has recently also been shown to be implicated in anti-cancer drug sensitivity [209]. To test whether this observation is a general feature of sensitive cell lines rather than an mTOR pathway inhibitor-specific feature, we performed an enrichment analysis with the GeneTrail 3 C++ library [210] as follows: For each drug, we sorted the cell lines by decreasing sensitivity, i.e. by increasing logarithmized IC50 values. As a category, we define all cell lines for which TJP1 expression is low (same z-score-based definition as used for the binarization of the gene expression values, cf. Section 5.2) and carry out an enrichment analysis, i.e. for each drug, we test for an enrichment of this category at the top or bottom of the cell line list using a Kolmogorov-Smirnov test. Interestingly, TJP1 seems to be a key sensitivity determinant for the vast majority of drugs (258/320 drugs in GDSC1, 156/175 drugs in GDSC2). We repeated this type of analysis for all of the selected gene expression features and could identify several of the selected features as being also of broad importance to a variety of drugs (for further details and the results of this analysis, please refer to Appendix Tables B.26 and B.27). For example, NCKAP1 low expression (239/320 drugs in GDSC1, 156/175 drugs in GDSC2) and PTPRF low expression (205/320 drugs in GDSC1, 150/175 drugs in GDSC2) also seem to influence the sensitivity to various drugs. NCKAP1, whose low expression was detected as sensitivity-associated for AZD8055 by our analysis, is part of the WAVE complex that regulates actin filament organization. Moreover, NCKAP1 has been shown to promote tumor progression in specific mice melanoma cells [211]. With this potential oncogenic role, NCKAP1 expression might be a useful marker for the malignancy grade of cell lines that influences the susceptibility to drug treatments. Similarly, PTPRF, which MERIDA identified as a sensitivity factor of Rapamycin, was recently found to promote tumor progression by activating WNT signaling in colorectal cancer [212]. Thus, low expression of PTPRF could classify cell lines as less malignant and, as a consequence, might improve treatment outcome prediction.

When explicitly investigating the results for Setting 3, in which the ILPs can decide on whether to select the a priori knowledge features of drug response, we observe that neither LOBICO nor MERIDA opts for these variables (cf. Appendix Section B for all output rules). This indicates that from an ML perspective, their integration is

not favourable. We wanted to identify putative causes for this lack of performance. Therefore, we exemplary investigated how the drug response biomarkers distribute across the Rapamycin sensitivity scale. Indeed we find that literature biomarkers do not accumulate at the top (sensitive end) of this cell line list. On the contrary, these alterations usually do not affect the most sensitive cell lines. Furthermore, some predictive biomarkers from the literature seem to be more informative than others. For example, the well-studied PTEN loss and certain PTEN loss-of-function mutations seem to be rather predictive for the sensitivity of Rapamycin. In contrast, other features such as STK11 loss or FBXW loss are predominantly present in the resistant group, although there exists literature evidence that these alterations support sensitivity as well [213], [214]. There are various explanations for this phenomenon: There can be differences in the strength of the predictive biomarkers due to differing roles of distinct biomarkers in biological pathways. Furthermore, the cell lines are usually affected by various mutations that typically influence several biological pathways, which can, in turn, downgrade the importance of a single predictive biomarker.
A group of cell lines that repeatedly stood out in our analyses, i.e., the haematological cell lines, may be responsible for some of the abovementioned results. In particular, we found that the haematological cell lines are enriched at the top - the most sensitive end - of almost all sorted drug response lists (cf. Appendix Figures B.13 and B.14 for mTOR inhibitors). Thus, it is not unlikely that some of the identified biomarkers reflect the heightened susceptibility of haematological cell lines to anti-cancer treatments rather than actual drug responsiveness. This explanation also coincides with similar observations by Sharifi-Noghabi et al. [215].

5.5 Discussion

A major goal of personalized medicine in cancer is optimizing anti-cancer drug treatment based on measurements of different genetic and molecular characteristics of cancer cells. Here, the ability of a model to provide interpretable decisions is of utmost importance and poses a challenge to current ML methods. Not only should the model structure and results be amenable to human reasoning, but the current body of knowledge should also be considered.
To this end, we devised MERIDA, an integer linear programming formulation de-

rived from LOBICO, a similar method by Knijnenburg et al. While LOBICO can fit any Boolean formula to a given data set, we decided to restrict the space of permissible logical functions to reduce the runtime, which should enable MERIDA to fit larger models and integrate more input features. Indeed, we showcase that reducing the space of allowed logic combinations accelerates the runtime of the corresponding branch-and-cut algorithm tremendously. Additionally, our proposed weighting schemes as importance measures for the cell lines in the objective function improved the runtime of MERIDA (and LOBICO) even further. Meanwhile, the selected features of LOBICO and MERIDA are similar for the small models LOBICO can fit. Due to the reduced runtime, MERIDA, however, can handle considerably larger input feature sets and construct larger models. Despite the reduced space of logic combinations, the statistical performance of MERIDA is similar to or superior to LOBICO. In particular, MERIDA achieves superior results with respect to the statistical sensitivity measure, which is of particular importance for unbalanced data sets with a low amount of true positives.
We also investigated another option for improving the prediction models: integrating prior knowledge. While this could be implemented for any logical model, the systemic integration of predictive biomarkers has not been conducted for previously published logical models. This knowledge can stem from biomarker databases or be newly acquired by our own method. By integrating known biomarkers of drug response into our models, we can guide them towards learning why a known effect does not occur and identify drivers in samples with no already known drug response association. Indeed, by using knowledge from biomarker databases, we could improve the statistical performance of some of the drugs. Additionally, we could enhance the statistical performance by iteratively running our new method and adding biomarkers (features) identified by previous runs to the next model. Here, our results indicate that the iterative application provides similar models compared to the one-shot approach with a significant speed-up.
Nevertheless, the statistical performance should still be improved. An important factor influencing the performance is the choice of the used features. Feature selection or dimension reduction is usually indispensable to counteract the curse of dimensionality present for such large multi-omics data sets as provided by the GDSC. We decided to do a literature-driven feature selection with curated cancer driver lists to focus on alterations that are most likely involved in therapeutic responsiveness.

However, by restricting our models to known cancer-associated genes, we neglect the importance and discovery of novel drug-gene associations. In Chapter 6, we will pursue the opposite strategy and select features based on their statistical association with the drug response, and in the subsequent chapter, i.e., Chapter 7, we more generally gauge the influence of feature selection and dimension reduction techniques on the performance of various ML methods. Apart from the feature selection or dimensionality reduction technique, incorporating additional feature types, such as epigenomic data, may improve the performance and interpretability of the results.
In this chapter, we devised a classifier whose resulting logical models can be used to identify effective drugs for a given cell line or tumour. However, a central task in personalized medicine is to find the best drug or a suitable combination of drugs for a specific cell line. To this end, the effective drugs have to be prioritized, which can be achieved by performing regression. In Chapter 6, we start addressing this issue by developing a simultaneous regression and classification method. In Chapter 8, we then combine this method with a novel drug sensitivity measure and a reliability estimation framework to achieve reliable drug prioritization.

6 Simultaneous regression and classification random forests

In Chapter 5, we already discussed that the high specificity of (targeted) anti-cancer agents leads to a severe underrepresentation of sensitive cell lines in the GDSC cancer cell line panel. Consequently, classifiers that do not take any countermeasures are prone to poor statistical performance for sensitive cell lines. Yet, the correct identification of sensitive cell lines is essential since they represent cases of treatment success. To counteract this class imbalance in MERIDA, we followed the approach by Knijnenburg et al. [27] and pursued the strategy of integrating sample weights that reflect the importance of the cell lines. In particular, we and Knijnenburg et al. opted for sample weights reflecting continuous sensitivity values.

While the class imbalance problem is commonly discussed and addressed in the ML literature, the analogous regression problem, i.e., regression imbalance, has hardly been investigated [28]. The regression imbalance problem is defined as a systematic under- or overestimation of values at the edges of a distribution, while these more extreme values represent cases of particular interest. We can easily translate this concept to the drug sensitivity prediction domain: the sensitive cell lines, i.e., cell lines with a high sensitivity to a drug, are of utmost importance, yet regressors would systematically underestimate their response since the majority of cell lines has a low sensitivity. The first contribution of this work is to demonstrate that regression imbalance indeed occurs for a multitude of ML methods, i.e., neural networks, random forests, boosting trees and the elastic net, in the context of drug sensitivity prediction. These methods learn to predict drug sensitivity values around the mean drug response well but fail to deliver reasonable predictions for the highly sensitive cell lines.

In the drug sensitivity prediction literature, only a few approaches directly tackled this problem (cf. Table 6.1): Basu et al. [158] proposed an iterative response-weighted elastic net, where in each iteration more weight is placed on the extreme values at the leftmost end (high sensitivity) of the response distribution. To define this leftmost end, they employ an approximation for the lower limit of the 90% confidence interval. However, the lower limit of the 90% confidence interval does not

necessarily correspond to the set of sensitive cell lines for each drug.
For random forests, the tendency to predominantly predict values around the mean was already described as well [156, 216]. Although not initially designed to circumvent this issue, the HARF (Heterogeneity-Aware Random Forests) approach by Rahman et al. [155] can nevertheless help. By integrating cancer types as classes into the leaf nodes of a conventional random forest regressor, Rahman et al. perform cancer type prediction. This prediction is subsequently used to weight the trees for the regression task. As a consequence, HARF delivers continuous predictions around the mean of each integrated cancer type. The success of this procedure depends on the used cancer types: if cancer types substantially differ with respect to their average drug responses, the HARF method results in accurate predictions and outperforms conventional regression random forests [155]. Thus, for each drug, Rahman et al. consider only cancer types with sufficiently different average drug responses and neglect all other available cancer types. In practice, they usually even limit their analyses to two cancer types and, hence, dismiss large portions of the data.
To directly address regression and class imbalance, we propose our novel approach SAURON-RF (SimultAneoUs Regression and classificatiON Random Forests). More generally speaking, SAURON-RF simultaneously predicts continuous and discrete drug sensitivity values by exploiting the mechanics behind HARF. In particular, we suggest three main extensions (cf. Figure 6.1):

1. Instead of using cancer types, we propose to employ a partition into sensitive and resistant cell lines as a class assignment for SAURON-RF. Hence, SAURON-RF inherently meets the demand for distinct average drug responses per class without the disadvantage of sample exclusion.

2. We suggest counteracting class imbalance and regression imbalance by using either upsampling techniques or calculating sample-specific weights.

3. We introduce alternatives to the tree weighting scheme by Rahman et al. to further improve the regression performance.

In the following, we start with describing the HARF concept. Afterwards, we give a comparative overview of our novel approach, SAURON-RF. Lastly, we present the results of a comprehensive study on the GDSC data set, in which we demonstrate that regression imbalance is present for a variety of ML regression methods and that SAURON-RF effectively counteracts not only regression imbalance but also class imbalance.

Table 6.1: Comparison between different supervised ML methods for drug sensitivity prediction. In this table, we state whether a method performs regression, classification, or both and whether the data-inherent imbalance between sensitive cell lines and resistant cell lines has been considered for model design.
* Rahman et al. [155] and Matlock et al. [156] do not explicitly address class or regression imbalance. However, the approach by Rahman et al. can be employed for this purpose and the approach by Matlock et al. has a positive effect on the performance of the sensitive cell lines.

Name and author	Supervised technique	Class imbalance addressed	Regression imbalance addressed
Menden et al., 2013 [152]	regression (neural network)	✗	✗
Zhang et al., 2015 [153]	regression (dual-layer integrated drug-cell line similarity network)	✗	✗
LOBICO by Knijnenburg et al., 2016 [27]	classification (integer linear program delivering Boolean rules)	✓	✗
Stanfield et al., 2017 [166]	classification (cell line and drug proximity networks)	✗	✗
SRMF by Wang et al., 2017 [154]	regression (similarity regularized matrix factorization)	✗	✗
HARF by Rahman et al., 2017 [155]	regression (random forest augmented with cancer types)	✗	(✓)*
HNMDRP by Zhang et al., 2018 [167]	classification (similarity networks)	✗	✗
Matlock et al., 2018 [156]	regression (model stacking)	✗	(✓)*
RWEN by Basu et al., 2018 [158]	regression (response-weighted elastic net)	✗	✓
CDRscan by Chang et al., 2018 [159]	regression (convolutional neural networks)	✗	✗
QRF by Fang et al., 2018 [31]	regression (quantile regression random forest)	✗	✗
NCFGER by Liu et al., 2018 [160]	regression (neighbor-based collaborative filtering with global effect removal)	✗	✗

Continued on next page

Table 6.1 – continued from previous page

Name and author	Supervised techn.	CI addressed	RI addressed
DeepDR by Chiu et al., 2019 [161]	regression (neural networks)	✗	✗
Deep-Resp-Forest by Su et al., 2019 [168]	classification (deep cascaded forest)	✗	✗
netBITE by Oskooei et al., 2019 [162]	regression (biased tree ensemble)	✗	✗
Deng et al., 2020 [163]	regression (neural network)	✗	✗
PathDSP by Tang et al., 2021 [164]	regression (neural network)	✗	✗
MERIDA by Lenhof et al., 2021 [21]	classification (integer linear program delivering Boolean rules)	✓	✗
GraphDRP by Nguyen et al., 2022 [165]	regression (neural network)	✗	✗
SAURON-RF by Lenhof et al., 2022 [29]	regression and classification (simultaneous regression and classification random forest)	✓	✓

Authors' contributions

This chapter is based on my publication *Simultaneous regression and classification for drug sensitivity prediction using an advanced random forest method* [29] in terms of content and text. The idea of the SAURON-RF methodology was coined by me. For the corresponding publication, I also implemented the SAURON-RF software and drafted the manuscript, while the therein presented computational experiments were performed by me and Lea Eckhart. In particular, Lea Eckhart implemented the feature selection method, and conducted the comparison experiments with methods other than SAURON-RF. Moreover, the design of the study was yielded from Lea Eckharts master's thesis, which I supervised. All authors discussed the results and commented on the manuscript.

6.1 Simultaneous regression and classification with random forests

Our novel approach, SAURON-RF (SimultAneoUs Regression and classificatiON Random Forest), aims to tackle the drug sensitivity prediction problem by using regression random forests simultaneously for regression and classification. To this end, we pursue a similar strategy to HARF [155], i.e., we train regression random forests and augment them with class information. However, instead of using cancer types as classes, we use the drug-specific division into sensitive and resistant cell lines introduced in Chapter 5. We first discuss HARF before we describe our novel approach, SAURON-RF, in detail.

6.1.1 HARF

The functioning of the HARF algorithm can be briefly summarized as described below. The algorithm starts with the training of a canonical regression random forest on a given model matrix $\mathbf{X} \in \mathbb{R}^{N \times P}$ and the continuous response vector $\mathbf{y} \in \mathbb{R}^N$ with N being the number of samples and P being the number of features (cf. Chapter 4, Section 4.3.4 for a description of RF). Let B be the number of trees of this regression RF. Rahman et al. now additionally use the cancer types of the training samples to assign each leaf of a tree to the class that represents the relative majority (mode) of the training samples in the corresponding leaf node. To this end, suppose $C = \{c_1, \ldots, c_k\}$ is the set of different classes and $\mathbf{d} \in C^N$ is the vector with class assignments for all samples. By assigning each leaf node to the majority class of its training samples, each leaf, and, consequently, each tree and the complete random forest can classify a sample x: the class prediction of a tree is given by the class prediction of the reached leaf, and the class prediction of the RF is provided by the mode over the class predictions of all trees. This classification can now be employed to modify the weight of the trees for the final continuous prediction. To calculate the continuous prediction of the forest, Rahman et al. use only the trees for which the predicted class is equal to the class prediction of the forest. We can express this using tree-specific weights. The tree-specific weights $w_b(x)$ can be calculated by dividing an indicator variable $I_b(x)$, which is 1 if the mode of the leaf of tree b is equal to the mode of the forest and 0 otherwise, by the sum of these indicator variables over all reached leaves in all trees, i.e., the weight $w_b(x)$ is

$$w_b(x) = \frac{I_b(x)}{\sum_{\beta=1}^{B} I_\beta(x)} \tag{6.1}$$

In the following, we refer to this tree weighting as *binary tree weighting* scheme (binary t.w.). Finally, the continuous prediction of the regression random forest is given by the weighted average of all trees. To this end, let $\hat{f}_b(x)$ be the regression prediction of a single tree b for a sample x. Then, the final prediction is given as

$$\hat{f}(x) = \sum_{b=1}^{B} w_b(x) \cdot \hat{f}_b(x) \quad . \tag{6.2}$$

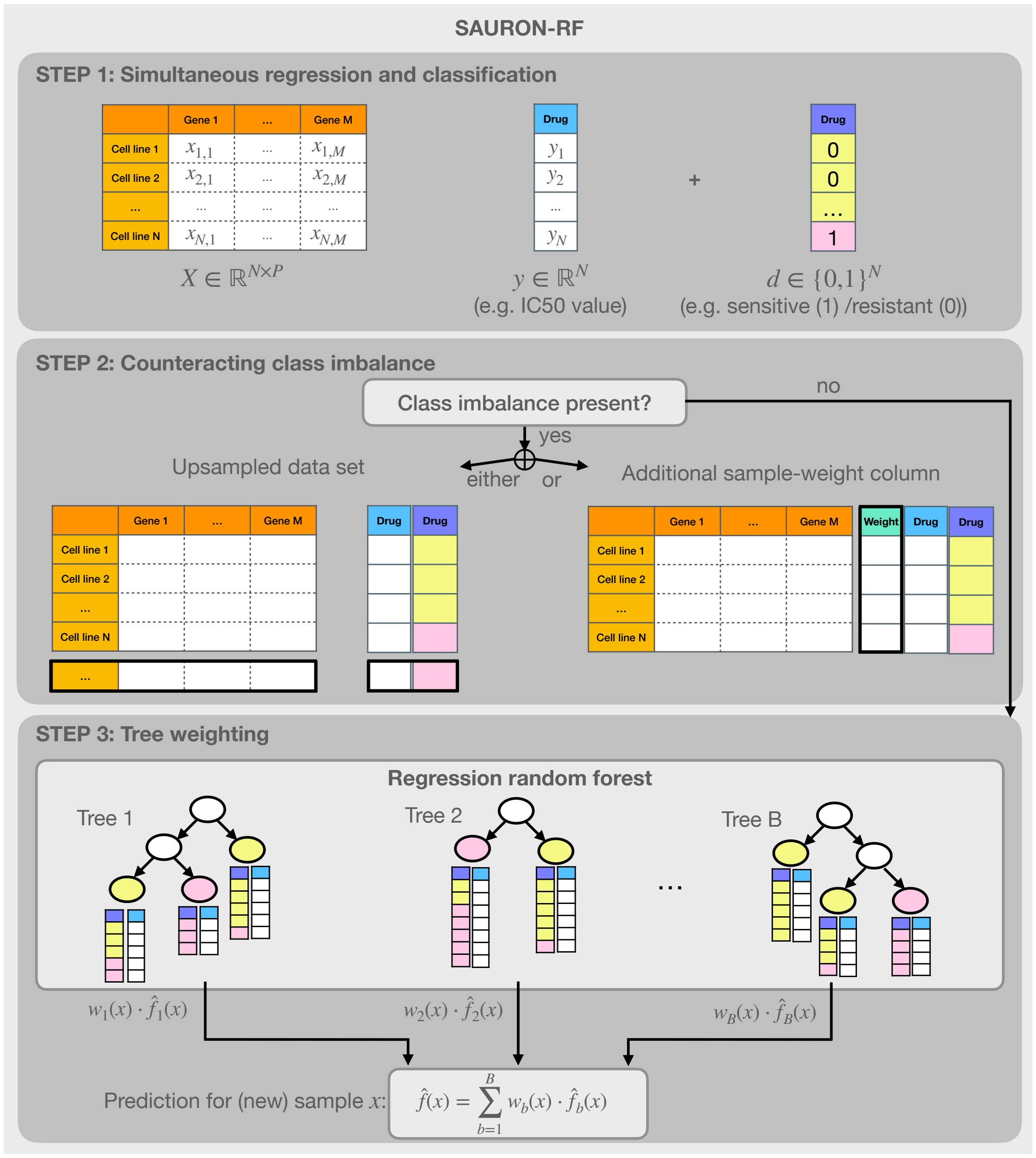

SAURON-RF
STEP 1: Simultaneous regression and classification
Gene 1
Gene M
Cell line 1
Cell line 2
Cell line N
$x_{1,1}$
$x_{1,M}$
$x_{2,1}$
$x_{2,M}$
$x_{N,1}$
$x_{N,M}$
$X \in \mathbb{R}^{N \times P}$
Drug
y_1
y_2
y_N
$y \in \mathbb{R}^N$
(e.g. IC50 value)
+
Drug
0
0
1
$d \in \{0,1\}^N$
(e.g. sensitive (1) /resistant (0))
STEP 2: Counteracting class imbalance
Class imbalance present?
no
yes
either
or
Upsampled data set
Additional sample-weight column
Weight
STEP 3: Tree weighting
Regression random forest
Tree 1
Tree 2
Tree B
$w_1(x) \cdot \hat{f}_1(x)$
$w_2(x) \cdot \hat{f}_2(x)$
$w_B(x) \cdot \hat{f}_B(x)$
Prediction for (new) sample x:
$\hat{f}(x) = \sum_{b=1}^{B} w_b(x) \cdot \hat{f}_b(x)$

Figure 6.1: SAURON-RF workflow. The figure depicts the three-step workflow of SAURON-RF. The SAURON-RF procedure starts with a preprocessing phase, in which the dimensionality of the input gene expression matrix is reduced, e.g., by our minimum-redundancy-maximum-relevance feature selection, and the binary drug response (sensitive/resistant) is derived from the logarithmized IC50 values. In the second step, SAURON-RF addresses class imbalance by either performing upsampling or using sample-specific weights. In the last step, a regression random forest is trained from the given continuous IC50 values and the gene expression matrix. Then, each leaf node is assigned to the majority class of its corresponding samples. Thereby, the regression random forest can also be used to classify samples by majority vote over the trees. For a new sample x, a continuous prediction is calculated by multiplying the tree-specific weights $w_b(x)$ with the tree-specific predictions $\hat{f}_b(x)$.

6.1.2 SAURON-RF

Our approach, SAURON-RF, relies on the basic functioning behind HARF. However, we propose three major modifications that can be summarized as follows (see Figure 6.1): instead of performing a cancer type classification by augmenting an RF regressor with class types, we propose augmenting the RF regressor with a division into sensitive and resistant cell lines to simultaneously perform anti-cancer drug response classification and regression. This direct coupling of the drug response classification and regression task then allows us to counteract the observed regression imbalance for the sensitive cell lines with techniques borrowed from classification. Finally, we suggest using a different tree weighting scheme than Rahman et al. to further improve the performance of the RF.

Augmenting RF with division into sensitive and resistant cell lines: To this end, let N again be the number of samples, P be the number of features, and $\mathbf{X} \in \mathbb{R}^{N \times P}$ the corresponding model matrix. Suppose $\mathbf{y} \in \mathbb{R}^N$ is the continuous response vector for the training of the weighted regression random forest. From the continuous drug response, we can derive a discretized version by comparison of the continuous values with a discretization threshold t as introduced in Chapter 5, dividing the cell lines into sensitive (1) and resistant (0) ones. Suppose $\mathbf{d} \in \{0,1\}^N$ is the corresponding binary response vector. Note that we restrict the following description of our method to the binary case, while the method can be easily extended to multiple classes (cf. Chapter 8, Section 8.1.1).

Sample-specific weights: One possibility to counteract class imbalance is using sample-specific weights in the training procedure of the RF. We tested several possibilities to set these weights. A simple way is to utilize the proportion between the majority and the minority. W.l.o.g., let 0 be the majority and 1 the minority class. Moreover, suppose that N_{Ma} is the number of samples in the majority class and N_{Mi} is the number of samples in the minority class. The *simple sample weights* (simple s.w.) for all samples $x_i, i \in \{1, \dots, N\}$ can be calculated as

$$w_i^* = \begin{cases} 1, & \text{if sample } i \text{ belongs to the majority} \\ \frac{N_{\text{Ma}}}{N_{\text{Mi}}}, & \text{if sample } i \text{ belongs to the minority} \end{cases} \tag{6.3}$$

In Chapter 5, we also discussed the use of alternative weight functions to this simple weight function, i.e., weight functions that emphasize samples based on the distance from the threshold t, such as

$$w_i^* = \frac{|y_i - t|^g}{2 \cdot \sum_{\forall n \in \{1,\ldots,N\}:d_n = d_i} |y_n - t|^g} \tag{6.4}$$

with $g \in \{1, 2\}$. Based on the exponent, we refer to them as *linear* and *quadratic*. Once these initial sample weights have been determined, they are carried through the training procedure of the RF: These weights are directly incorporated in the node-specific sample weights, which themselves are part of the prediction of the response of a node, the error measure, and the splitting criterion (cf. Chapter 4, Section 4.3.4). Let $\delta(v)$ be the set of bootstrap samples that belong to the node v. The node-specific sample weight for a sample x_i in this node v can be calculated as follows

$$w_i^v = \frac{w_i^*}{\sum_{n \in \delta(v)} w_n^*} \quad . \tag{6.5}$$

Alternatives to sample-specific weights: As an alternative to calculating sample-specific weights to assign samples a higher importance during training, we propose to perform upsampling prior to the training of the forest. In particular, we suggest upsampling the minority class by random drawing with replacement from the set of sensitive samples until the number of samples from the minority and majority class is equal. We call this method *upsampling*. We also investigated an upsampling scheme, which we call *proportional upsampling*, where we consider the distance from the threshold t to determine how often a sensitive sample should be duplicated. To this end, we calculate the linear sample-specific weights (see Equation 6.4) for the sensitive cell lines. Then, we multiply these weights by 2 such that they sum up to 1. For each sample, this value then reflects the percentage of importance. We then determine the desired number of copies by multiplying this importance by the total number of resistant samples.

Tree-specific weights: Once the RF regressor is trained, we also employ the classes to modify the continuous predictions. We assign each leaf to the weighted majority of its training samples using the initial sample weights, which directly enables classification: for a sample x, the class prediction of a tree is given by the prediction of the reached leaf, and the class prediction of the RF is provided by the mode over the class predictions of all trees. Based on this classification, the trees can obtain weights that can be incorporated in the continuous prediction of the regression random forest (cf. Equation 6.2). For SAURON-RF, we suggest a variety of different weighting schemes reflecting the data-inherent class distribution. For the tree weighting scheme referred to as *binary sensitive (binary sens.)*, we use the HARF weights (cf. Equation 6.1) for the samples predicted to be sensitive, while we use the usual random forest prediction, i.e., tree weight $\frac{1}{B}$, for the samples predicted to be resistant. For the tree weighting scheme called *majority tree weighting*, we first determine for each leaf the weighted fraction of samples that agree with the class prediction of the forest. For a particular tree, we use this fraction in the reached leaf normalized by the sum of all fractions as tree weight. Let $I_n(x)$ be an indicator function that is 1 if the class of sample n is equal to the overall majority of trees and 0 if not. Let μ be the reached leaf node of tree b. With the fraction given by

$$\text{frac}_b(x) = \frac{\sum_{\forall n \in \delta(\mu)} I_n(x) \cdot w_n^*}{\sum_{\forall n \in \delta(\mu)} w_n^*} \tag{6.6}$$

the tree-specific weight is given as

$$w_b(x) = \frac{\text{frac}_b}{\sum_{\beta=1}^{B} \text{frac}_\beta(x)} \quad . \tag{6.7}$$

Finally, we also used a tree weighting scheme, which we call *majority sensitive (majority sens.)*, where we employ the majority tree weighting given in Equation 6.7 for the sensitive samples and the usual RF weight $\frac{1}{B}$ for the resistant samples.

6.2 Data preparation

We downloaded Release 8.3 (June 2020) of the Genomics of Drug Sensitivity in Cancer (GDSC) cancer cell line panel for the analyses presented in this chapter. In particular, we obtained the pre-processed gene expression matrix (Affymetrix Human Genome U219 Array) and the pre-processed drug sensitivity data in the form of logarithmized IC50 values (GDSC1 compounds: Syto60 and resazurin assay, GDSC2 compounds: CellTiter-Glo assay) from the GDSC website.

6.2.1 Processing of drug response values

In our analyses, we focused on the more recently published GDSC2 data set, which is based on an improved drug screening procedure and assay (cf. Chapter 3 for more details on the GDSC data set development). This version of the GDSC2 data set consists of 809 cell lines and 198 drugs. In this chapter, we use only drugs with more than 750 available cell lines, which results in 86 investigated drugs. For each drug, we consider only cell lines with complete information for drug sensitivity and gene expression.
Since SAURON-RF performs a simultaneous regression and classification analysis, it requires a continuous and discrete response vector as model input for training. As a continuous measure of drug response, we use the provided logarithmized IC50 values. As discussed in Chapter 5, the logarithmized IC50 values can be binarized using the heuristic outlier procedure introduced by Knijnenburg et al. [27]. For each drug, we apply this procedure to the logarithmized IC50 data to derive one threshold able to divide the cell lines into sensitive (1) or resistant (0) ones, which results in a binary drug-specific response vector.

6.2.2 Model training

To comprehensively benchmark SAURON-RF, we trained various ML models: elastic nets, neural networks, boosting trees, and numerous random forest approaches, including HARF. All methods were fit and tested on the same data that was prepared with the following scheme. For each drug, we divide the corresponding data set consisting of the triple of gene expression matrix, continuous drug response and binary drug response into a training (80%) and test data set (20%) by randomly drawing without replacement from the set of available cell lines. The training data set is then subject to a 5-fold CV split. Since the gene expression data is very high-dimensional, we implemented a feature selection that reduces the dimension of the expression matrix before serving as input for the machine learning methods. Consequently, the feature selection is performed on each CV training set and the complete training data set. We decided to use the greedy minimum-redundancy-maximum-relevance algorithm proposed by Kwak and Choi [30] as a feature selection method. An in-depth description of this algorithm is provided in Chapter 4. Briefly, the algorithm selects the top features that maximize the feature-response (gene - drug) mutual information (MI) while minimizing the feature-feature (gene - gene) MI. The optimal number of features is a hyperparameter, which can be determined during the CV. In our application case, the features (gene expression values) and the response (logarithmized IC50 values) are continuous. Hence, we discretized them for calculating the MI using equal width binning. A table with the used (hyper)parameters of the feature selection algorithm and all ML models is provided in Appendix Table C.30.

6.3 Implementation and data deposition

All random forest-based approaches were implemented using Python 3 and the sklearn RandomForestRegressor and RandomForestClassifier packages [182]. In particular, the RandomForestRegressor from sklearn also serves as the basis for our implementation of HARF and SAURON-RF. The corresponding Python script can be executed from the console with a single configuration file in JSON-format as input. The configuration file comprises information about standard random forest parameters such as the number of trees or the number of features per split and specific parameters introduced for SAURON-RF, e.g., the technique to counteract class imbalance or the desired tree weight. The script delivers several output files in a simple tab-separated txt-format containing the training and test set predictions as well as the evaluation of performance measures such as mean-squared error and accuracy. We deposited the code for our newly developed SAURON-RF algorithm (including a HARF implementation) alongside example files on our publicly available GitHub: `https://github.com/unisb-bioinf/SAURON-RF.git`.

Besides RF-based approaches, we also trained boosting trees, elastic nets and neural networks. Boosting trees and elastic nets were fit using the R packages gbm (Version 2.1.8) [217] and glmnet (Version 4.1.1) [218], respectively. The neural networks were fit using the Python framework Keras (Version 2.3.1) together with the GPU Version of Tensorflow (1.13.1) [219, 220].

6.4 Results

We applied a multitude of ML regression methods, i.e., elastic net, neural networks, boosting trees, and random forests, to the 86 drugs from the GDSC data set. To evaluate their regression performance, we calculated the mean squared error (MSE) and median squared error (median SE) averaged across all drugs. Additionally, we calculated the MSE and median SE for the set of sensitive and set of resistant cell lines separately to assess whether regression imbalance is present. Expectedly, all tested approaches achieved a similar overall regression performance with elastic nets and random forests having a slightly superior average performance with respect to mean-squared error (MSE) and median squared error (cf. Figure 6.2). Moreover, we observed that all methods predict values around the mean response of the cell lines (resistant cell lines) considerably well while especially the lower (highly sensitive) end of the sensitivity scale is misfitted to a varying degree, clearly indicating that regression imbalance is present independent of the ML method. In Figure 6.3A, we exemplary illustrate this circumstance for 5-Fluorouracil, a typical representative of high class imbalance within the GDSC (8.6% of sensitive cell lines). We can clearly discern that values around the mean of the resistant samples are sufficiently well predicted. Simultaneously, none of the sensitive samples has a reasonable prediction. Next, we determined the classification performance of all methods. To this end, we derived a discrete response from the continuous prediction by comparison to the drug-specific IC50 thresholds. As to be expected, we found that the classification performance for the sensitive cell lines, i.e., the sensitivity, is low, while the specificity is high. In summary, the classification performance of the models reflects their ability to predict the continuous response and vice versa. This can already be seen as indicative for positive effects on predictive performance by a coupling of regression and classification.

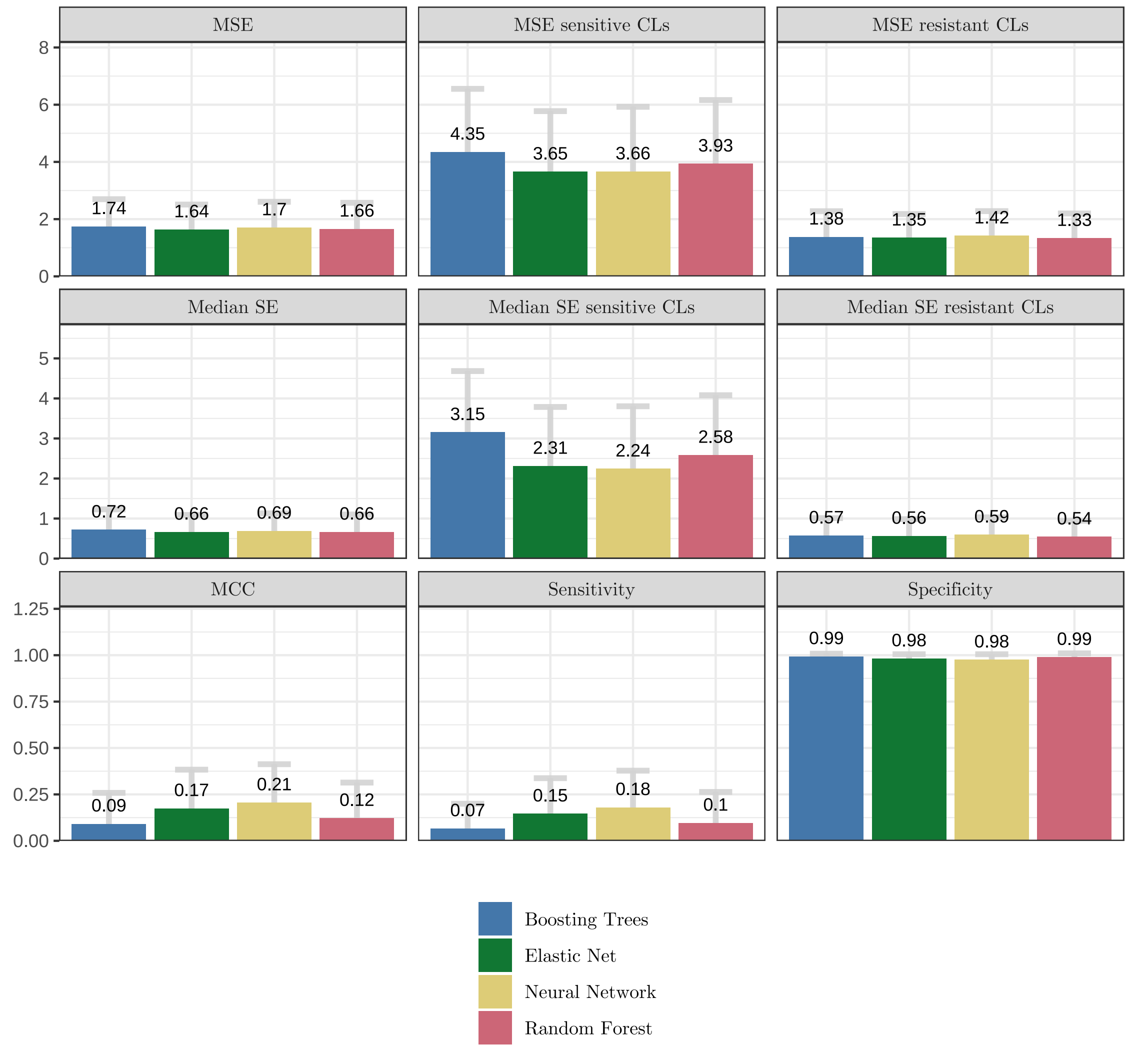


Figure 6.2: Average test set performance for different ML algorithms. In this figure, we compare boosting trees, elastic nets, neural networks, and random forests. We show the average test set performance across the 86 different drugs from the GDSC for 20 input features. We measured performance in terms of the mean-squared error (MSE), median squared error (median SE), Matthew's correlation coefficient (MCC), sensitivity, and specificity. Each measure was once calculated across all cell lines, the set of sensitive cell lines, and the set of resistant cell lines.

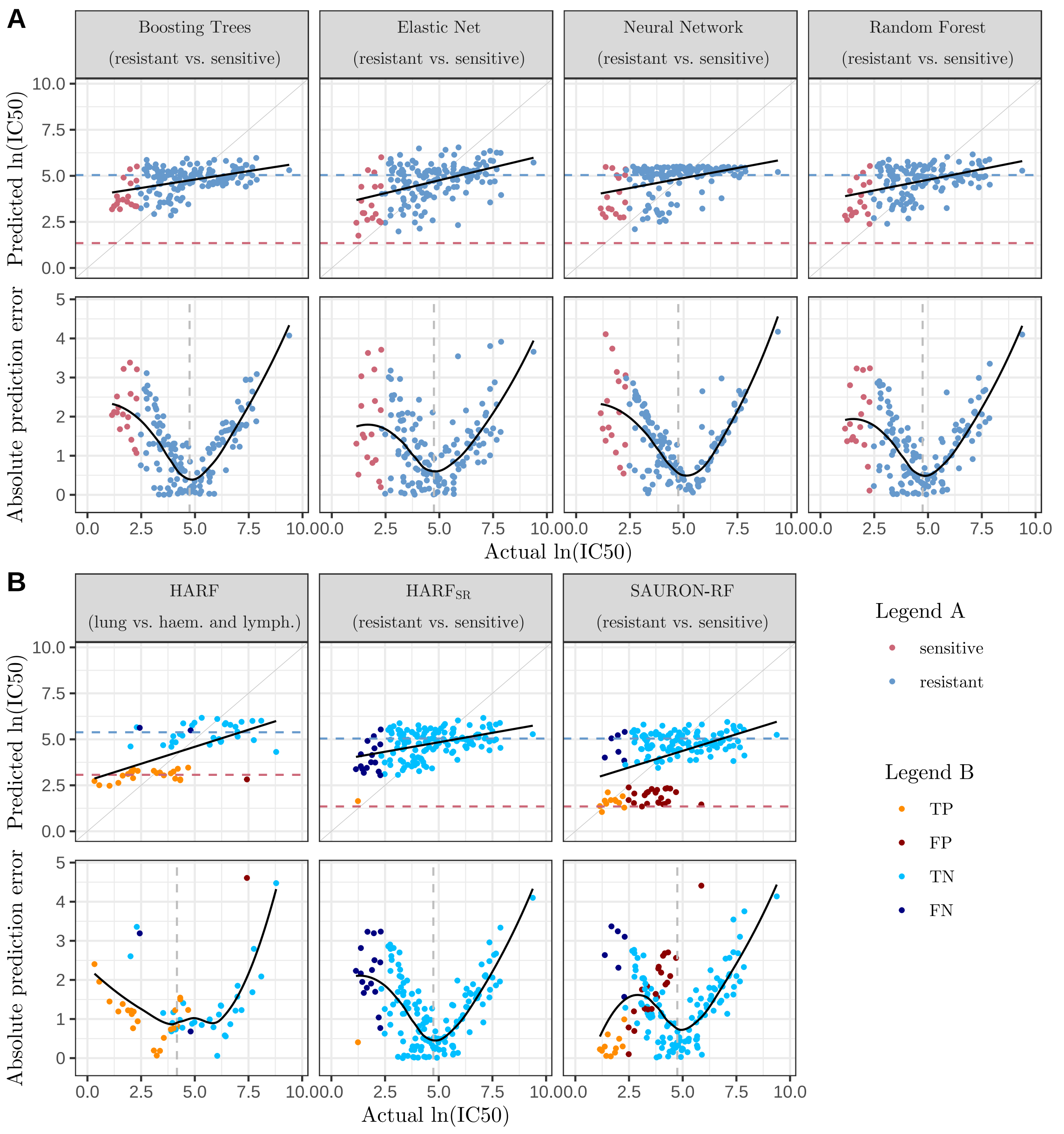


Figure 6.3: Regression performance of different ML methods for 5-Fluorouracil. This figure exemplifies the performance of different ML algorithms when applied to the 5-Fluorouracil data set of the GDSC database using 20 input features. In Figure A, we compare boosting trees, elastic net, neural networks and random forests. The upper row of Figure A shows the predicted IC50 values plotted against the actual IC50 values including a fitted regression line, which is shown as a solid black line. The mean IC50 of training samples for each investigated class is depicted as a horizontal dashed line. The lower row of Figure A shows the absolute prediction error. Here, the solid curve is a loess curve fitted to the error, the vertical dashed line gives the mean IC50 of all training samples. In Figure B, we depict analogous plots for a comparison between HARF (HARF applied to two cancer types with different average drug responses), HARF$_{SR}$ (HARF applied to our proposed class division), and SAURON-RF (SAURON-RF simple s.w., binary sens t.w.).

6.4.1 Development of SAURON-RF

The HARF approach by Rahman et al. performs simultaneous drug response regression and cancer type classification by integrating cancer types with distinct average drug responses into a random forest regressor trained on drug sensitivity data. More specifically, they use the cancer type distribution of the leaf nodes to modify the tree weights after regression (cf. Section 6.1.1). In Figure 6.3B, we show its performance for a typical application scenario: We applied HARF to 5-Fluorouracil using two cancer sites with distinct average drug responses. In particular, we opted for the two most abundant sites of origin of the GDSC data set as classes: haematopoietic/lymphoid cell lines (139 cell lines) and lung cell lines (135), resulting in a very balanced data set. As expected, the HARF predictions for the haematopoietic/lymphoid and lung cell lines are well-separated and concentrated around the mean of the respective class. However, there is still the tendency to misfit the upper and lower end of the prediction sale. Moreover, we had to exclude 532 of 806 available cell lines for 5-Fluorouracil in the GDSC, which is especially critical in data poor settings. Apart from that, this modeling approach is even infeasible if an investigated cancer type exhibits a heterogenous response distribution for a particular drug.

To address these issues, we suggest linking the regression and classification problems more closely by using the division into sensitive and resistant cell lines instead of cancer types. Consequently, we meet the demand for distinct average drug responses per class by definition and do not have to exclude samples to enforce this requirement artificially. We call this method HARF$_{SR}$. In Figure 6.3B, the performance of HARF$_{SR}$ is shown. As expected when considering the poor classification performance of RFs as shown in Figure 6.2, the substantial class imbalance with only 8.6% of sensitive cell lines negatively affects the classification performance of HARF$_{SR}$ because it identifies only one sensitive cell line correctly. Since the regression performance of the HARF approach relies on the classification performance, we also do not obtain reasonable continuous predictions for the sensitive cell lines.

Class imbalance is a common obstacle for ML algorithms, especially for drug sensitivity prediction [221, 222, 223, 224, 225, 21]. For most drug data sets in the GDSC, the sensitive cell lines are heavily underrepresented with an average sensitive-to-resistant ratio of 1:10. If we do not counteract class imbalance, the average sensitivity of classification random forests (cRF) is only 9% (cf. Figure 6.4). In order to improve the classification and subsequently the regression performance, we address the class imbalance. We proposed using various upsampling schemes and sample-specific weights (cf. Section 6.1.2). To enhance the regression performance even more, we suggest alternative tree-weighting schemes to the one by Rahman et al. (cf. Section 6.1.2). We call the resulting method SAURON-RF. Using 5-Fluorouracil as an example, it can be seen that SAURON-RF follows a promising principle (see Figure 6.3). In the ensuing sections, we benchmark SAURON-RF against multiple RF-based approaches.

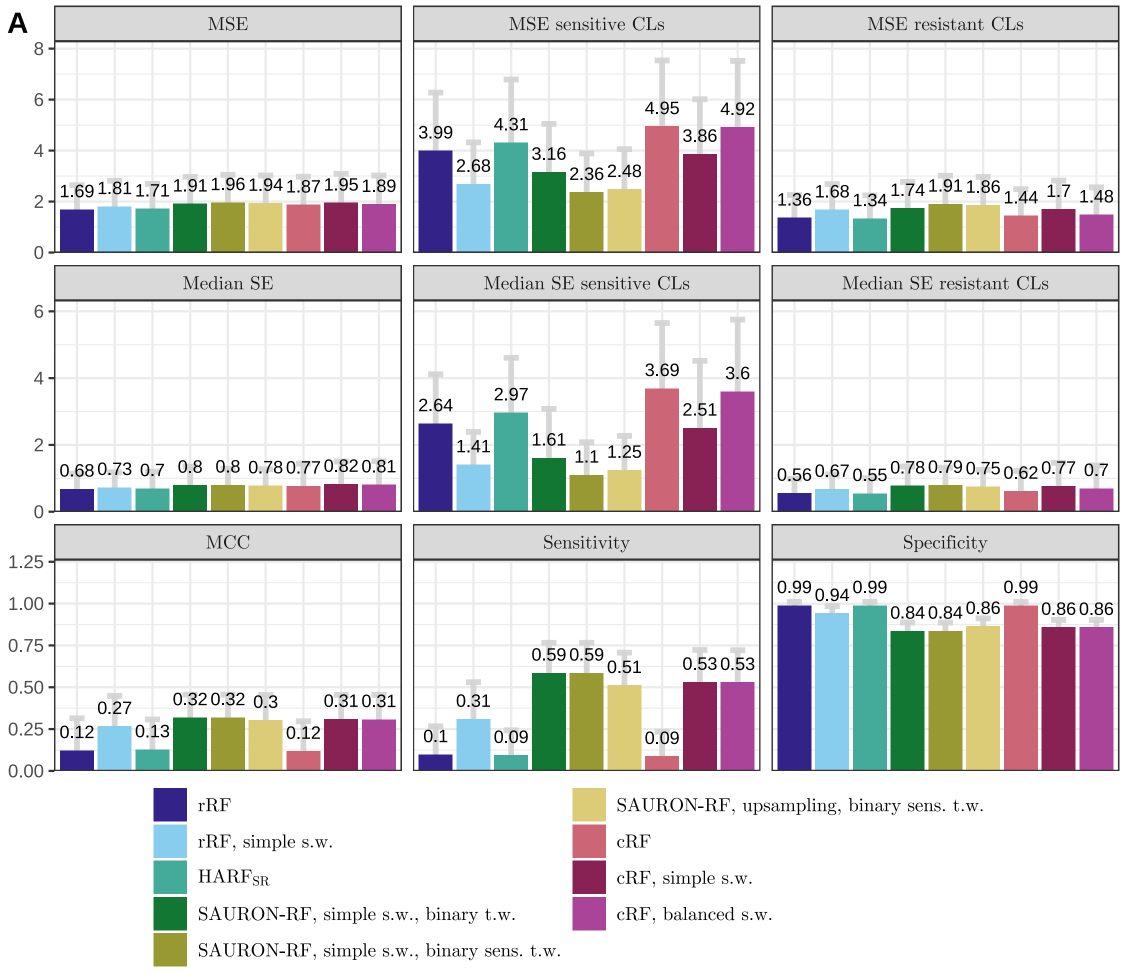
6.4.2 Performance comparison of SAURON-RF to regression and classification RFs

We applied various versions of standard regression RFs (rRFs), classification RFs (cRFs), HARF$_{SR}$, and our new approach SAURON-RF to 86 drugs of the GDSC. As described in the previous section, we employed the MSE and median SE to evaluate the regression performance. Additionally, we calculated sensitivity, specificity, and MCC to assess classification performance. While HARF$_{SR}$ and SAURON-RF can inherently perform classification and regression, we had to derive a discrete response for rRFs and a continuous response for cRFs. To determine class predictions for a standard regression RF, we compare its prediction to the drug-specific IC50 threshold. We derived the continuous response for cRFs as follows: For a new sample, we first determine the trees that agree with the majority vote of the forest. Then we average the continuous values of the training samples in the reached leaf nodes. Finally, we then average over the selected trees.

Figure 6.4 shows an overview of the test results for 20 gene expression input features obtained by our minimum-redundancy-maximum-relevance feature selection. We provide results for additional input feature set sizes in Appendix Figures C.32 - C.35. Generally speaking, those results strongly resemble the ones that will be presented in the following. As can be seen in Figure 6.4A, both rRF and HARF$_{SR}$ have a low overall MSE, while they perform very poorly for the sensitive cell lines, i.e. the average MSE for the sensitive cell lines is 2.9 times (rRF) and 3.2 times (HARF$_{SR}$) as high as the average MSE of the resistant cell lines. Moreover, the statistical sensitivity for both approaches is very low with 10% for rRFs and 9% for

HARF$_{SR}$. When counteracting class imbalance by employing simple sample weights (simple s.w.) as defined in Equation 6.3, the sensitivity of both methods increases substantially by 21% and 50%, respectively. In comparison, the specificity decreases only moderately by 5% (rRF) and 15% (HARF$_{SR}$). Simultaneously, the average MSE for the sensitive cell lines drops from 3.99 to 2.68 (-32.8%) for rRF and from 4.31 to 3.16 (-26.6%) for HARF$_{SR}$. In contrast, the MSE for the resistant cell lines rises from 1.36 to 1.68 (23,5%) for rRF and from 1.34 to 1.74 (29.8%) for HARF$_{SR}$. When combining the sample weights with the proposed tree weighting scheme (SAURON-RF, simple s.w., binary sens t.w.), the reduction in average MSE for the sensitive cell lines compared to rRF even amounts to 1.63 (-40.8%), and compared to HARF$_{SR}$ amounts to 1.95 (-45.2%). The MSE for the resistant cell lines increases by 0.55 (40.4%) for rRF and 0.57 (42.5%) for HARF$_{SR}$. In total, the MSE of the resistant and sensitive cell lines is more leveled for SAURON-RF: while the average MSE of the sensitive cell lines for rRF is 2.9 times as high as the average MSE of the resistant cell lines, the average MSE of the sensitive cell lines for SAURON-RF is only 1.2 times as high as the average MSE of the resistant cell lines. We note that upsampling improves the predictive performance for the sensitive samples slightly less than the proposed simple sample weights. The overall best performance in terms of classification and regression was achieved by SAURON-RF simple s.w., binary sens. t.w.

For this best-performing model, we also plotted the absolute gain in performance for the sensitive cell lines against the absolute loss of performance for the resistant cell lines compared to rRF on a per drug basis (cf. Figure 6.4B). For almost all drugs, the absolute performance gain of SAURON-RF for the sensitive cell lines outweighs the absolute performance loss for the resistant ones in terms of classification and regression. When we focus on the correctly identified sensitive cell lines, we note that the MSE value for SAURON-RF is even three times smaller than the MSE of all sensitive cell lines. Hence, our results also strongly indicate that further reducing the classification error would substantially improve regression performance.



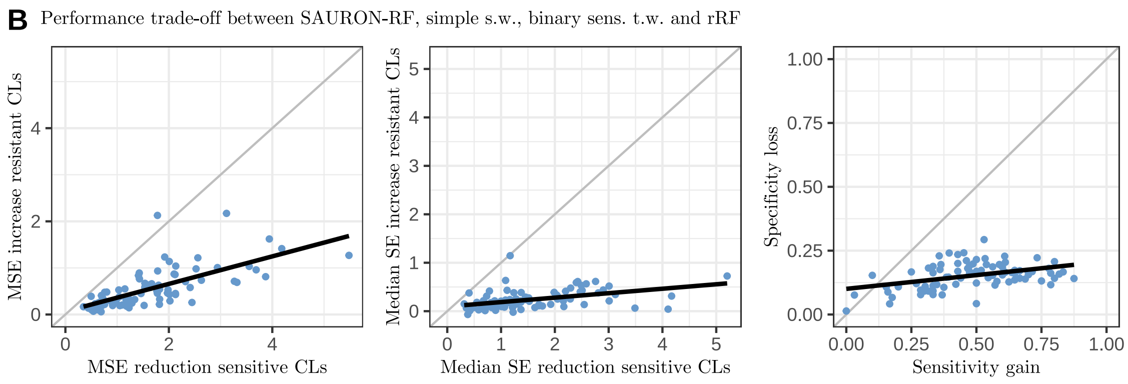


Figure 6.4: Random forest test set performance. In Figure 3A, we compare regression random forests (rRFs), classification random forests (cRFs), and HARF$_{SR}$ with different versions of our suggested approach SAURON-RF. We show the average test set performance across the 86 different drugs for 20 input features. Results for additional versions of SAURON-RF and other input feature set sizes can be found in the Appendix Figures C.32 - C.36. In Figure 3B, we depict for each of the 86 drugs the tradeoff between the absolute reduction in test error for the sensitive cell lines and the absolute increase in test error for the resistant cell lines when comparing our best-performing version of SAURON-RF (SAURON-RF simple s.w., binary sens. t.w.) with rRF.

To quantify whether our joint regression and classification approach can compete with a pure classification method, we compared the classification performance of SAURON-RF to classification RFs (cRFs). The cRF exhibits a similar classification performance to HARF$_{SR}$ and rRF: 99% specificity, 9% sensitivity, and an MCC of 0.12. When integrating our proposed simple sample weights into cRF (cRF, simple s.w.) or using the built-in class balancing of the cRF python implementation (cRF, balanced s.w.), the sensitivity and MCC improve. However, in terms of statistical sensitivity, SAURON-RF is superior, while still maintaining a similar specificity. Furthermore, SAURON-RF outperforms all tested cRF versions in the regression task for the sensitive cell lines.
Finally, we also compared SAURON-RF to a hierarchical RF approach, where we fit one classification RF on the complete data set and two regression RFs on the respective subsets of sensitive and resistant cell lines. The performance of this approach was also inferior to all versions of SAURON-RF presented in Figure 6.4 (see Appendix Figure C.36).

6.4.3 Evaluating predictive biomarkers

Typically, RFs lend themselves well to interpretation due to their method of origin, binary decision trees. While it is possible and relatively common to manually inspect a single decision tree in order to understand the importance of the features used for splitting, there is the need to aggregate the feature importances across all trees for RFs (cf. Chapter 4). To assess the capability of our method to select biomarkers that provide information on drug response, we investigated the feature importances provided by the in-built sklearn functionality. For each drug, we only considered models trained using our best-performing version of SAURON-RF (SAURON-RF, simple s.w., binary sens t.w.) and selected the ten drugs with the largest MCC during CV as visualized in Figure 6.5. For each of these 10 drugs, we then sorted the features of the best model and investigated their potential involvement in mechanisms of drug sensitivity or resistance through literature research. For most drugs (6/10), we could identify at least one feature with a known relation to drug response among the top five features. For the BCL2 inhibitor ABT737, for example, the most important features include its target gene BCL2, as well as MIR22HG, IDH2 and

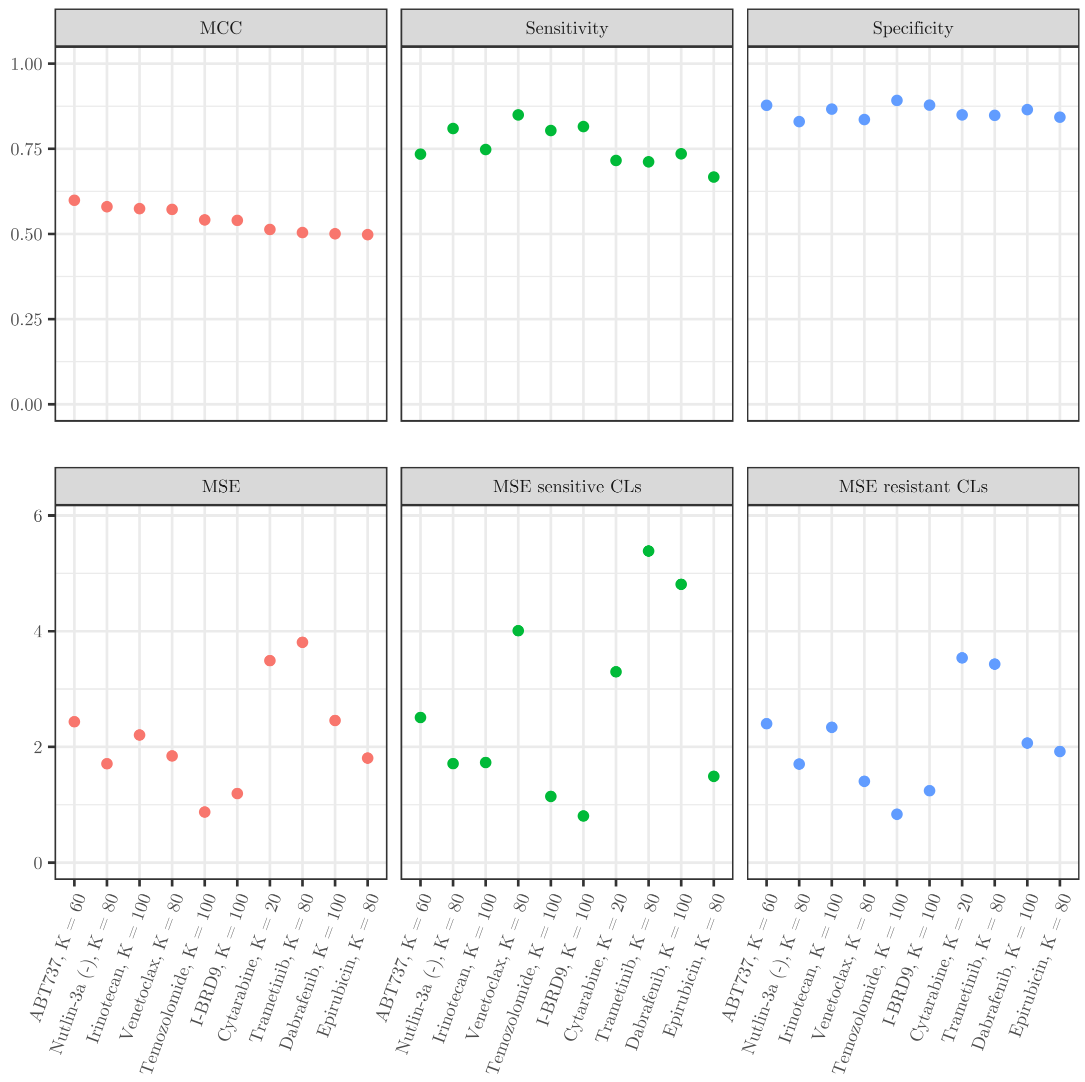


Figure 6.5: Top-performing models. This figure depicts the classification and regression performance for the 10 drugs with the highest CV MCC. We obtained this list by sorting the results for the best-performing SAURON-RF model (SAURON-RF, simple s.w., binary sens t.w.) for all drugs using the CV MCC. The hyperparameter K indicates the size of the input feature set.

BLVRB. While MIR22HG is involved in the downregulation of BCL2 [226, 227], mutations in IDH2 are associated with increased sensitivity to ABT737 [228, 229]. Low expression of BLVRB has been linked to increased sensitivity to the BCL2 inhibitor Obatoclax [230]. For the drug Nutlin-3a(-), which targets the p53 pathway, we identified its direct target gene MDM2 as the most important feature. Additionally,

we identified RPS27L – a downstream target of TP53 – that is activated in cells, which undergo apoptosis following Nutlin-3a(-) treatment [231]. Further important features included DDB2 and CYFIP2, whose expression has been shown to increase through Nutlin-3a(-) treatment [232, 233]. This could indicate the involvement of these genes in the mode of action of Nutlin-3a(-). We provide information on the selected features for the remaining drugs in Appendix Tables C.31 - C.40.
We also investigated whether we are able to identify features more commonly associated with drug sensitivity or resistance. For each feature, we determine the number of drugs for which it is selected and then average its feature importance. As shown in Figure 6.6, we could identify three genes that are not only selected for many drugs but have a high average feature importance as well: PPIC, SDC4 and DCBLD2. All three genes encode transmembrane proteins. PPIC is involved in protein folding [234], while SDC4 and DCBLD2 play a role in cellular signaling [235, 236]. DCBLD2 upregulation is associated with poor prognosis in various cancers, including glioblastoma and colorectal cancer [237, 238, 236] and has recently been linked to 5-Fluorouracil resistance [236]. However, the role of these genes in multi-drug resistance remains to be investigated.

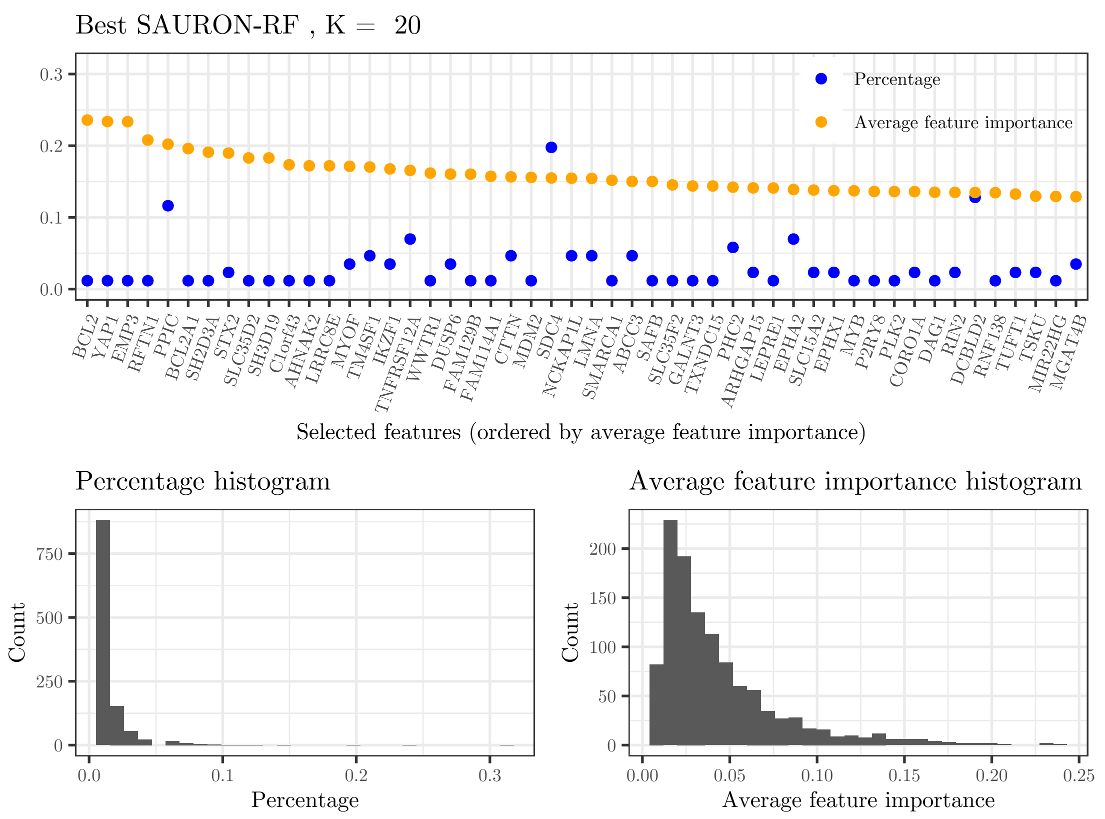


Figure 6.6: Feature importance for SAURON-RF using 20 input features. This figure shows the 50 features with the highest average feature importance for the best-performing SAURON-RF model using $K = 20$ input features per drug. The per feature average was calculated based on the drugs for which this feature was selected in the feature selection. The corresponding percentage of drugs is also depicted (blue dots).

6.5 Discussion

With our novel method, SAURON-RF, we aimed to address the regression imbalance problem in drug sensitivity prediction. More specifically, we pursued the goal to mitigate the negative effects that the skewed distribution of drug response values in favour of the drug-resistant cell lines exerts on the predictions of the sensitive cell lines.

To this end, we pursued the strategy of augmenting a drug response regression RF with information on the corresponding class partition into sensitive and resistant cell lines. This proceeding allowed us to address regression and class imbalance directly with commonly employed techniques for counteracting class imbalance during simultaneous regression and classification. We could substantially improve not only the classification performance for the sensitive cell lines but also the respective regression performance at the expense of a moderate loss in performance for the resistant cell lines. We have shown that jointly conducting regression and classification is superior to performing either of them alone. Moreover, we demonstrated that an accurate classification can substantially improve a subsequent regression. The practical implementation of this strategy could be best achieved using a combined regression and classification approach instead of a hierarchical approach. Therefore, the development and refinement of joint classification and regression approaches like SAURON-RF seem to be a promising field of future research.

Here, the ultimate goal would be the integration of SAURON-RF into the medical decision process as part of a comprehensive decision support pipeline such as ClinOmicsTrail^{bc} [239]. ClinOmicsTrail^{bc} helps to interpret molecular and clinical data of patients and, amongst others, assesses standard-of-care treatments, the pharmacokinetics of drugs, potential adverse effects, as well as known biomarkers for drug efficacy. SAURON-RF could complement this existing information by providing evidence for the on-label use of drugs and also giving insight into potential off-label treatment options.

However, we recognize several potential starting points for improving our method. Since we have shown that joint regression and classification is advantageous for both classification and regression compared to each of them alone, it could be beneficial for SAURON-RF to combine regression and classification splitting criteria in the tree-building process, e.g., as described in [240, 241]. Another important factor that influences the predictive performance of our method is the choice of input features and dimensionality reduction method. Currently, we conduct a maximum-relevance-minimum redundancy feature selection on gene expression data. While gene expression has been shown to be the most informative data type [17], the performance and interpretability of the machine learning models can, for example, benefit from the integration of (epi)genomics data [17], protein-protein interaction networks [162], pathway information [19], and pharmacogenomic a priori knowledge [21]. The process in which the features are selected or reduced has also already been shown to affect the performance of ML models [242]. In the following chapter, we aim to provide a more comprehensive benchmark for a diverse set of feature selection and dimensionality reduction techniques combined with various ML models (cf.

Chapter 7). An additional factor that influences the predictive performance of our approach is the class division that we actively leverage in model building. The binarization procedure by Knijnenburg et al. only represents a heuristic to estimate the division into sensitive and resistant samples, while the factual division is unknown. Since our analyses clearly indicate that the most sensitive samples, i.e., samples of potential treatment success, benefit from considering such a partitioning if they are underrepresented, it might be interesting to investigate different discretization schemes. Thus, we pursued this goal in Chapter 8.

To benchmark SAURON-RF against a variety of ML methods, we followed a very common approach: we assessed performance measures such as accuracy, sensitivity, specificity, MSE, and median SE. However, none of these measures estimates the reliability of predictions for previously unseen samples, which is, however, required to translate ML methods into clinical decision-making. There exist ML approaches that are specifically designed for such purposes, e.g., conformal prediction (see Chapter 4). We are going to investigate the usefulness of conformal prediction in Chapter 8.

In order to provide even more useful recommendations for decision support systems, our method should be able to perform drug prioritization, i.e., to deliver a list of recommendable drugs sorted by their efficiency. In principle, SAURON-RF can become leveraged for precisely this purpose by identifying effective drugs with the classification part and then ranking the effective ones using the regression part. Unfortunately, IC50 values are not comparable across drugs (cf. Chapter 3), which prevents us from sorting the drugs that we predict to be effective. In Chapter 8, we also address this shortcoming.

7 A benchmarking of dimensionality reduction techniques and machine learning methods

In the previous two chapters, we presented two novel approaches for drug sensitivity prediction: In Chapter 5, we introduced the rule-based classifier MERIDA, and in Chapter 6, we devised the random forest-based simultaneous regression and classification model SAURON-RF. MERIDA and SAURON-RF differ in many respects. In particular, they employ different methods for modelling drug sensitivity and counteract the curse of dimensionality using distinct dimensionality reduction (DR) techniques. To be more precise, MERIDA relies on literature-driven feature selection and creation, and SAURON-RF employs a data-driven approach based on the maximum-relevance-minimum-redundancy principle, i.e., a selection of features statistically associated with the response. We already discussed (cf. Chapter 4) that a dimensionality reduction step is typically indispensable before or during model training when investigating data sets with far more features than samples. Thereby, the estimation uncertainty (variance of the model) is reduced, and the ML models usually become more interpretable. Thus, such a step is integrated into most drug sensitivity methods, as shown in Table 7.1. Simultaneously, most authors do not report how they opted for a specific dimensionality reduction technique, i.e., whether they tested other dimensionality reduction techniques and how these performed in comparison. However, given the variety of DR techniques and ML models they could be combined with, the correct choice might strongly influence the ML model performance. Some benchmarking studies on drug sensitivity prediction in cancer aim to establish best practices. For example, Jang et al. systematically assessed the difference in performance when combining different types of input features, ML models, and drug response summary metrics [243]. However, they did not focus on comparing different DR strategies. Koras et al. compared several DR techniques [242], particularly literature-based feature selection using drug targets and target pathways and data-driven feature selection with elastic net and random forests, but did not systematically compare the combination with different ML methods. Moreover, they did not account for size differences in the investigated feature sets.

In this chapter, we present the results of a comprehensive benchmarking study on DR techniques and ML methods for drug sensitivity prediction in cancer. More specifically, we trained four different supervised ML methods (elastic net, random forest, boosting trees, neural networks) combined with nine different DR approaches (random, literature-based, variance, correlation, enrichment, minimum-redundancy-maximum-relevance, principal component analysis, pathway activity, autoencoder) resulting in more than 16,000,000 investigated models. We consulted different quality measures such as interpretability, statistical performance regarding error measures, and runtime for the comparative evaluation. Our analysis reveals that for most drugs, the elastic net outperforms all other methods in terms of statistical performance and runtime. The best-performing DR methods were principal component analysis and the minimum-redundancy-maximum-relevance approach.

In the following, we first describe the study design before presenting the results of our benchmarking study.

Table 7.1: Comparison between methods for drug sensitivity prediction with respect to dimensionality reduction. In this table, we compare various drug sensitivity approaches with respect to their integrated dimensionality reduction method. Moreover, we assessed whether the authors benchmarked the selected dimensionality reduction method against others.

Name and author	Methodology	DR technique	DR comparison
Menden et al., 2013 [152]	neural network	literature-based	✗
Zhang et al., 2015 [153]	dual-layer integrated drug-cell line similarity network	✗	✗
LOBICO by Knijnenburg et al., 2016 [27]	integer linear program delivering Boolean rules	literature-based	✗
Stanfield et al., 2017 [166]	cell line and drug proximity networks	literature-based	✗
SRMF by Wang et al., 2017 [154]	similarity regularized matrix factorization	matrix factorization	✗
HARF by Rahman et al., 2017 [155]	random forest augmented with cancer types	RELIEFF	✗
HNMDRP by Zhang et al., 2018 [167]	similarity networks	✗	✗
Matlock et al., 2018 [156]	model stacking	RELIEFF	✗

Continued on next page

Table 7.1 – continued from previous page

Name and author	Methodology	DR technique	DR comparison
KRL by He et al., 2018 [157]	kernelized rank learning	✗	PCA
RWEN by Basu et al., 2018 [158]	response-weighted elastic net	elastic net	✗
CDRscan by Chang et al., 2018 [159]	convolutional neural networks	literature-based	✗
QRF by Fang et al., 2018 [31]	quantile regression random forest	correlation, random forest feature importance	✗
NCFGER by Liu et al., 2018 [160]	neighbor-based collaborative filtering with global effect removal	✗	✗
DeepDR by Chiu et al., 2019 [161]	neural networks	autoencoder	PCA
Deep-Resp-Forest by Su et al., 2019 [168]	deep cascaded forest	random	✗
Dr.VAE by Rampášek et al., 2019 [129]	semi-supervised generative modeling based on variational autoencoders	literature-based, autoencoder	✗
netBITE by Oskooei et al., 2019 [162]	biased tree ensemble	feature biasing	✗
Deng et al., 2020 [163]	neural network	literature-based	✗
PathDSP by Tang et al., 2021 [164]	neural network	pathway enrichment	✗
MERIDA by Lenhof et al., 2021 [21]	integer linear program delivering Boolean rules	literature-based	✗
GraphDRP by Nguyen et al., 2022 [165]	neural network	unknown	✗
PPORank by Liu et al., 2022 [131]	deep reinforcement learning	literature-based	✗
SAURON-RF by Lenhof et al., 2022 [29]	simultaneous regression and classification random forest	min.-redundancy-max.-relevance	✗
reliable SAURON-RF by Lenhof and Eckhart et al., 2023 [33]	simultaneous regression and classification random forest using CP	min.-redundancy-max.-relevance	✗

Authors' contributions
The main research presented in this chapter has been conducted by Lea Eckhart while working at the corresponding publication called *A comprehensive benchmarking of machine learning algorithms and dimensionality reduction methods for drug sensitivity prediction.* In particular, Lea Eckhart also drafted the manuscript. I was involved in the study design, contributed by implementing some of the feature selection methods, and reviewed and edited the manuscript.

7.1 Study design

With this study, we dedicated ourselves to gauging the influence of DR techniques on ML methods for anti-cancer drug sensitivity prediction. Generally speaking, we aimed to represent the current supervised ML landscape in this research area with our selection of ML methods and DR techniques. Thus, we decided to investigate four popular ML algorithms, i.e., elastic net, random forest, boosting trees, and neural networks, each one combined with nine different DR techniques (random, literature-based, variance, correlation, enrichment, minimum-redundancy-maximum-relevance, principal component analysis, pathway activity, autoencoder). Similar to Chapter 5, we compare the resulting models concerning runtime, interpretability and statistical performance. In the ensuing sections, we briefly present the study design encompassing the employed data set, the model training, and the main ideas of the DR and ML methods used.

7.1.1 Data set

For the analyses presented in this chapter, we used Release 8.3 of the Genomics of Drug Sensitivity in Cancer (GDSC) database, the version also described and employed in the previous chapter. Once again, we downloaded the preprocessed gene expression matrix (Affymetrix Human Genome U219 Array) and drug-screening data (GDSC1 compounds: Syto60 and resazurin assay, GDSC2 compounds: CellTiter-Glo assay) from the GDSC website. In particular, we employ the provided logarithmized IC50 values as drug sensitivity measure. Similar to our analyses in the previous chapter, we opted to analyze the GDSC2 data set that is based on an improved drug screening procedure and drug screening assay (CellTiter-Glo) compared to the GDSC1 data set. While the GDSC2 data set contains drug response measurements for 198 drugs, we consider only those 179 drugs with drug response values for at least 600 cell lines.

7.1.2 Model training and testing

We generally model drug sensitivity prediction as a supervised learning task employing the gene expression matrix as the input feature matrix and the logarithmized IC50 values of each drug as the response vector. More specifically, we train drug-specific regression models using four different ML algorithms predicting the logarithmized IC50 values of cell lines from a reduced representation of the initial gene expression matrix. We obtain the reduced representation of the initial gene expression matrix by applying nine different DR techniques. Each DR method generates input feature matrices of size $N \times k$, where N is the number of cell lines and constant, and k is the number of input features. We investigated $k \in \{1, 2, 3, \ldots, 25, 50, 100, 200, 300, 400, 500\}$. In the following, we call one triple consisting of an ML algorithm, DR method, and the number of input features k *setting.*
For each of the 179 drugs, we divide the available cell lines into a training (80%) and test (20 %) set by randomly drawing without replacement. The training set is then subject to a 5-fold cross-validation (CV) split on which we determine the best-performing hyperparameter combination of an ML model in terms of mean-squared error (MSE). For the best hyperparameter combination of a model, we train a final model on the complete training data set and evaluate its performance on the test set. In Table 7.2, we provide the model-specific hyperparameters. This procedure has to be performed for each setting separately, holding the training and test sets as well as the CV splits constant such that for one drug, all settings can be directly compared. Note that we also apply our DR methods only to samples in the respective training sets (including the CV).

7.1.3 Used dimensionality reduction techniques

In Chapter 4, we stated that DR techniques can be divided into feature selection (FS) and feature extraction (FE) approaches. In total, we analyzed six FS approaches (random, literature-based, variance, correlation, enrichment, minimum-redundancy-maximum-relevance) and three FE approaches (principal component analysis, pathway activity, autoencoder) that we present in the following.

Feature selection techniques

Generally, there exists a plethora of possibilities for choosing a subset of most informative features from the set of all available features. In Chapter 4, we introduced the three main groups of FS algorithms that can be distinguished: filter, wrapper and embedded methods. In our analysis, we solely focus on filter methods since these are most commonly applied for drug sensitivity prediction (cf. Table 7.1). In contrast to embedded methods, they are model-agnostic, i.e., independent of a specific ML model. Furthermore, they are also significantly less computationally expensive compared to wrapper methods. However, note that most ML models we investigated, i.e., random forest, boosting trees and the elastic net, perform embedded feature selection.

Randomized feature selection: An effortless and straightforward technique to reduce the dimensionality is the random selection of features. We generated the randomized feature sets by drawing k features without replacement from the set of all available features. To stabilize the prediction error estimation for each particular k, we draw ten random feature sets of size k and average the performance of the corresponding models.

Literature-based feature selection: Literature-based FS is extremely popular in anti-cancer drug sensitivity prediction. One possibility to perform this type of FS is to consider cancer driver genes [27, 21]. To this end, we decided to employ the cancer driver gene list from IntOGen (Release 2020-02-01) [244]. We removed genes with warnings (e.g., known artefacts) from the downloaded list and intersected the new gene list with the genes available in our input feature matrix from GDSC, resulting in the final list of 476 remaining genes. Next, we sorted the final gene list descendingly from most robust evidence of being a cancer driver (tier 1) to weakest evidence (tier 3). To break ties within a tier, genes were sorted descendingly according to the number of data sets in which they have been reported as cancer drivers. We then generated literature-based feature lists of size k by selecting the first k features of this sorted list.

Variance-based feature selection: For this FS, we calculated the variance of our input features (i.e., the variance of the expression per gene) and sorted the features in descending order from highest to lowest variance. We generated the list of size k by selecting the first k features for which the variance of expression values was largest.

Correlation-based feature selection: For this FS, we determined the Pearson correlation coefficient (PCC) between all input features and the response (cf. Chapter 4 for the definition of the PCC). We sorted the complete gene list decreasingly from the highest absolute PCC value to the lowest absolute PCC value and generated lists of size k by choosing the first k features of the sorted list.

Enrichment-based feature selection: Enrichment analyses are a family of approaches extremely popular in bioinformatics to uncover deregulated biological processes [210]. For example, Kolmogorov-Smirnov tests [245, 246] are employed to determine whether a category, i.e., a set of entities such as the genes of a specific pathway, is predominantly concentrated at the top or bottom of a sorted list, e.g., a list of differentially expressed genes. We investigated whether we can employ Kolmogorov-Smirnov tests to identify genes whose deregulation is linked to sensitivity or resistance to a particular drug. To this end, we proceed as follows: First, we generate a sorted list by increasingly ordering the available cell lines for a particular drug by their logarithmized IC50 values. Then, we define a category as the set of cell lines with a particular deregulation, i.e., up- or downregulation of a gene. We call a gene upregulated (downregulated) in a cell line if its expression z-score is greater (smaller) than the 95% (5%) percentile of the standard normal distribution. Note that we defined those categories using the complete training set of a particular

drug. Lastly, we can conduct Kolmogorov-Smirnow tests to check whether a certain category, i.e., the up- or downregulation of a particular gene, is overrepresented at the top or bottom of the sorted list of cell lines. Each test results in an enrichment direction indicating whether a putative enrichment occurs at the top (enriched) or the bottom (depleted) of the list and an associated p-value giving the significance of the test. We adjusted the p-values using the Benjamini-Hochberg procedure [247] separately for the two groups of up- and downregulated genes.
To obtain a final list from which we can select the k first elements, we proceed as follows: First, we assemble four lists from these results by combining each enrichment direction (enriched/depleted) with each deregulation state (upregulated/downregulated). We can sort each list from lowest (most significant) to highest p-value (least significant) and assign each gene a rank based on this ordering. We keep a gene solely in the list with its smallest rank (highest significance). To obtain a final sorted gene list, we merge the four lists starting with the smallest rank and break ties using the calculated p-values. We obtain lists of size k by choosing the first k features from this final list.

Minimum-redundancy-maximum relevance feature selection: To reduce the number of input features for SAURON-RF, we decided to use the greedy minimum-redundancy-maximum-relevance (MRMR) algorithm proposed by Kwak and Choi [30] since MRMR algorithms enjoy relative popularity, especially when applied to gene expression data, because of their simplicity and efficiency [248, 249, 250]. We already gave a thorough description of this algorithm in Chapter 4 and briefly presented it in Chapter 6. Thus, we will just revisit the main aspects relevant to this chapter.
In contrast to the two previously discussed FS approaches that aim to maximize the feature-response association (maximum relevance), MRMR-based FS methods simultaneously aim to minimize the feature-feature association (minimum redundancy). More specifically, the MRMR algorithm by Kwak and Choi [30] greedily adds features to the initially empty set of already selected features based on the evaluation of the mutual information between the set of all potential features and the response, as well as the set of all already selected features and the set of all potential features. Since the mutual information is defined between two discrete random variables, we had to discretize our continuous feature variables (gene ex-

pression values) and the continuous response variable (logarithmized IC50 values). To this end, we performed an equal-width binning with six bins. Given a desired number of selected features k, the output of the MRMR algorithm by Kwak and Choi is a list of size k decreasingly ordered from highest to lowest relevancy.

Feature extraction techniques

While FS techniques lower the dimensionality by selecting a subset of most relevant features, FE techniques transform the original set of features into a potentially smaller set of new features by application of (non-)linear functions. Thus, in contrast to FS strategies, where features from the original feature space are explicitly discarded, the new features generated by an FE technique can represent (non-)linear combinations of all original features. While this can allow for compact encoding of all relevant information from the original feature space in a few newly generated features, interpreting the new features and ML models trained on them can be difficult.

Principal component analysis: Principal component analysis is a technique that searches for a linear transformation of the original features such that most of the variation can be explained using fewer of the transformed features, called principal components. Each principal component is a normalized linear combination of the original features. Moreover, all principal components are orthogonal to each other. Thus, the first principal component is the direction in which the data varies most. It holds that all other principal components represent the direction explaining most of the variance while being orthogonal to all previously determined principal components [126]. Thereby, PCA generates a list of principal components ordered by the variance they explain.
We computed the principal components and obtained a list of size k by choosing the first k principal components.

Pathway activity score learning: Karagiannaki et al. [251] developed their Pathway Activity Score Learning (PASL) approach as a remedy for the limited interpretability of PCA. Roughly speaking, they follow the idea of calculating principal components on subsets of the feature matrix. Here, each subset corresponds to a predefined feature set, i.e., pathway and the obtained principal components to the pathway activity. In the corresponding publication, Karagiannaki et al. present several options to perform PASL. We restrict ourselves to describing the intuitive yet computationally expensive algorithm for calculating the optimal solution regarding explained variance. This optimal solution can be determined by repetition of the following steps until a stopping criterion is met: First, they calculate the first principal component of each pathway-associated sub-matrix and identify the principal component explaining most of the variance. This principal component is added to the solution. Then, they remove the contribution of this principal component from the complete input feature matrix and re-start the procedure. Like PCA features, PASL features are sorted from explaining most variance to explaining least variance. We employed the PASL default parameters and pathway data sets, to obtain feature lists of size k.

Autoencoder: An autoencoder is a type of neural network (cf. Chapter 4 for an introduction to artificial neural networks) that learns a lower-dimensional representation of an initial input matrix by minimizing the reconstruction error of this input matrix. It consists of two parts: an encoder part that compresses the input matrix to a lower dimension (e.g., k) and a decoder part that reconstructs the input matrix using this lower-dimensional representation. While the autoencoder is trained as a whole, the low-dimensional representation generated by the encoder part is used as input for ML models.
To generate a k-dimensional representation using autoencoders as DR technique, we must train one autoencoder per k for each drug and training set separately.

7.1.4 Used machine learning techniques

In Chapter 4, we already provided detailed descriptions of the four ML algorithms (elastic net, random forests, boosting trees, and neural networks) that we benchmark in this chapter. Therefore, we limit ourselves to briefly summarizing the key aspects here.

Elastic Net: The elastic net is a regression model that estimates the coefficients of a linear model with the ordinary least squares minimizer [177, 178]. Thereby, the response is modeled as a linear combination of the input features. Often, the coefficients of such linear models are penalized to reduce their variance and eliminate the least significant ones. The elastic net penalty, as proposed by Zou and Hastie, employs a combination of the L1 and L2 norm for this purpose [178] (cf. Chapter 4). This penalty allows some feature coefficients to become exactly zero, i.e., the elastic net performs embedded feature selection.

Random forests: Random forests are a decision tree-based ensemble method able to perform regression and classification [181]. Each tree of the ensemble is trained on a bootstrapped subset of the original training data set, and the predictions of all trees are combined into one prediction by averaging (regression) or majority vote (classification). To decide which feature to choose in order to perform a split within one node of a particular tree, only a randomly drawn subset of the features is considered. The best-performing feature with the corresponding splitting point is selected based on some performance measure, e.g., error reduction. This proceeding can also be regarded as a form of embedded feature selection.

Boosting trees: Like random forests, boosting trees is a decision-tree-based ensemble method. However, instead of parallelized training of decision trees on bootstrapped datasets, they employ sequential training. Here, each tree is fit to correct for the error of the predecessors. Depending on regularization parameters such as the depth of the single trees and the total number of trees, boosting trees also perform embedded feature selection.

Neural networks: Neural networks are an extremely versatile and flexible classification and regression approach loosely modeled after signal transmission mechanisms of neurons in the brain [119, 190]. They are represented as graphs consisting of multiple layers of nodes (artificial neurons) connected through weighted directed edges. In a neural network, nodes receive information via incoming edges, transform the information via (non-linear) activation functions, and forward the transformed information to their neighbours. While various network architectures exist, we solely consider fully connected feedforward neural networks, where each node is connected to all nodes of the consecutive layer.

7.2 Implementation and data deposition

We trained elastic nets, random forests and boosting trees with R. In particular, we employ glmnet v.4.1.3 [218] for elastic net, ranger v.0.13.1 [252] for random forests, and gbm v.2.1.8 [217] for boosting trees. For the training and tuning of neural networks, we rely on the python Keras API v.2.3.1[220] of Tensorflow v.1.13.1 [219]. In Table 7.2, we provide an overview of the tuned hyperparameters of the corresponding implementations. More detailed information can be found in Appendix Table D.41.

Most of the investigated DR methods (random, literature-based, variance, correlation, PCA (stats v.3.6.3 [253]), PASL [251]) were implemented with R. We applied PASL with the default settings given in the publication by Karagiannaki et al. [251]. The enrichment was conducted with the C++ implementation of the GeneTrail tool suite [210]. Moreover, we implemented the minimum-redundancy-maximum-relevance algorithm by Kwak and Choi [30] in C++. For the autoencoder, we use the python Keras API v.2.3.1[220] of Tensorflow v.1.13.1 [219]. Note that we did not perform hyperparameter tuning for the autoencoder because of an extremely high runtime (4.5 minutes on average for a single model). In Appendix Table and D.42, we provide information on the network architecture.

We deposited all corresponding code on our publicly available GitHub repository: `https://github.com/unisb-bioinf/ML_DR_Benchmarking_Drug_Sensitivity_Prediction`.

Table 7.2: Hyperparameter information of benchmarking study. In this table, we provide an overview of the employed R and python packages including the tuned hyperparameters of all four ML algorithms. If a particular hyperparameter is not mentioned in this table, we simply use the default settings of the respective package. In Appendix Tables D.41 and D.42, we supply further information on the network architecture and hyperparameters of the trained neural networks.

Model	Package	Parameter	Value(s)	#Combinations
Elastic net	glmnet, v. 4.1.3 (R) [218]	alpha	$[0, 1]$ in steps of 0.1	$11 \cdot 20 = 220$
		lambda	10^v, v: 20 equally spaced values $\in [-2, 2]$	
Random forest	ranger, v. 0.13.1 (R) [252]	mtry	$[1, 25]$ in steps of 2 and	up to 22
			$[40, 200]$ in steps of 20	
Boosting trees	gbm, v. 2.1.8 (R) [217]	n.trees	1-20	$20 \cdot 5 = 100$
		interaction.depth	1-5	
Neural network	Tensorflow, v. 1.13.1	# Hidden layers	1, 2, 3	$3 \cdot 2 \cdot 2 = 12$
	Keras API, v. 2.3.1	Activation function	tanh, ELU (none in output layer)	
	(Python) [219, 220]	Dropout	10%, 30%	

7.3 Results

In the ensuing sections, we present the results of our comprehensive benchmarking study. Here, we compare the different techniques with respect to runtime, statistical performance, and interpretability.

7.3.1 Runtime

We trained ML models for boosting trees, random forests, and elastic nets using an Intel Xeon Gold 6248 CPU with a 2.5GHz clock rate using 24 cores. For neural network training, we initially employed an Nvidia Tesla V100-SXM2 GPU since GPUs are optimized for such computations. However, the comparison to calculations on an Intel Xeon E5-2698 v4 using 24 cores revealed a higher runtime on the GPU. This observation may be attributed to the overhead from transferring calculations to the GPU that outweighs the computational speedup for small networks. Thus, we switched from GPU to CPU.

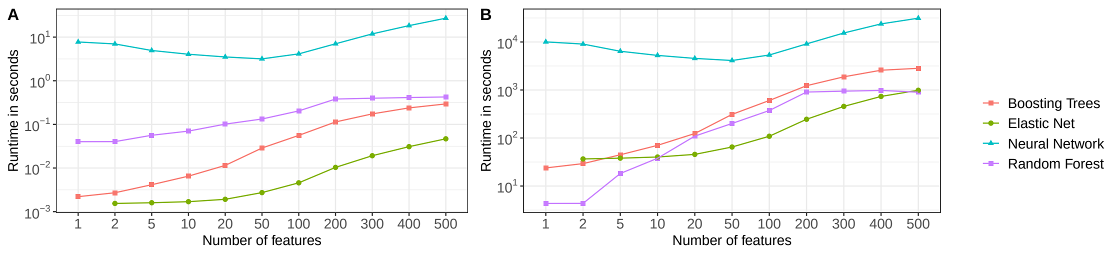


Figure 7.1: Runtime comparison of ML algorithms. Figure 7.1A visualizes the average runtime for one single model for each k. Figure 7.1B shows the accumulated runtime needed to tune (CV + final fitting + all hyperparameter combinations) one ML approach for each k.

In Figure 7.1, we present the observed runtimes. In Figure 7.1A, we plot the average runtime needed to train a single model for each k. Figure 7.1B depicts the accumulated runtime needed to tune (CV + final fitting + all hyperparameter combinations) one ML approach for each k. In Figure 7.1A, we find that the elastic net exhibits the lowest average runtime. Moreover, Figure 7.1B shows that despite being tested for most hyperparameter combinations, it remains the fastest approach regarding accumulated runtime for $k > 10$. On the contrary, neural networks are not only the method with the highest average runtime but also the approach with the highest accumulated runtime despite being tuned on the fewest hyperparameters.

7.3.2 Statistical performance

The statistical performance evaluation is subdivided into three parts: First, we show the average results across all drugs. Then, we assess the different settings (triples consisting of ML method, DR technique, and the number of input features k) in more detail. In particular, we focus on evaluating the complexity in terms of the parameter k. Finally, we explore options for improving predictive performance for the sensitive cell lines.

Average test performance In Figure 7.2, we depict the test error in terms of the mean-squared error (MSE) averaged across all drugs. More specifically, Figure 7.2A shows the performance averaged across all drugs and DR techniques. Thus, we obtain the best-performing ML model for each investigated k. To generate Figure 7.2B, we averaged across all drugs and ML methods, yielding one best-performing

DR technique per k.
Overall, Figure 7.2 clearly illustrates that errors decrease with increasing k. The most significant drop in error occurs between $k = 1$ and $k = 10$, followed by a drop for $k \geq 25$, which is particularly pronounced for the elastic net. The comparison of ML methods (Figure 7.2A) further reveals that neural networks consistently perform the worst, while random forests, boosting trees, and the elastic net achieve considerably similar performance. By comparing the DR techniques (Figure 7.2B), we observe that the autoencoder, the literature-based FS, and the random FS cannot compete with the remaining six DR techniques until $k = 25$. Then, their error significantly decreases. For the literature-based and random FS, the error approaches the level of the other six DR techniques, with the random FS being the least performing one out of these eight. However, the error of the autoencoder remains at an even higher level.
Note that neural network training was extremely time-intensive (cf. Section 7.3.1), which is why we trained the neural networks only for the 50 drugs with the highest number of available cell lines. To ensure that our evaluations were not biased, we also plotted the corresponding figure for the subset of 50 drugs employed for neural network training (cf. Appendix Figure D.37). This figure shows the same trends.

Analysis of different settings While the previous section provided a rough overview of the average performance of different ML methods and DR techniques, this section is dedicated to a more fine-grained analysis. To this end, recall that we refer to one triple consisting of an ML algorithm, DR method, and the number of input features k as setting. In this section, we determine and analyze the best-performing settings. In particular, we also quantify the improvement over a simple baseline method.
To generate the plots in Figure 7.3, we first identified the best-performing hyperparameter combination of each setting (based on the 5-fold CV error) for each investigated drug. Among these best-performing models for each setting and drug, we then identify the best-performing combination of DR technique and ML method for each drug and k based on test error. In Figure 7.3, we depict the best-performing settings for each k and in Figure 7.4 we show the best-performing k only. From Figure 7.3A, we can conclude that random forest and elastic net models are most successful with

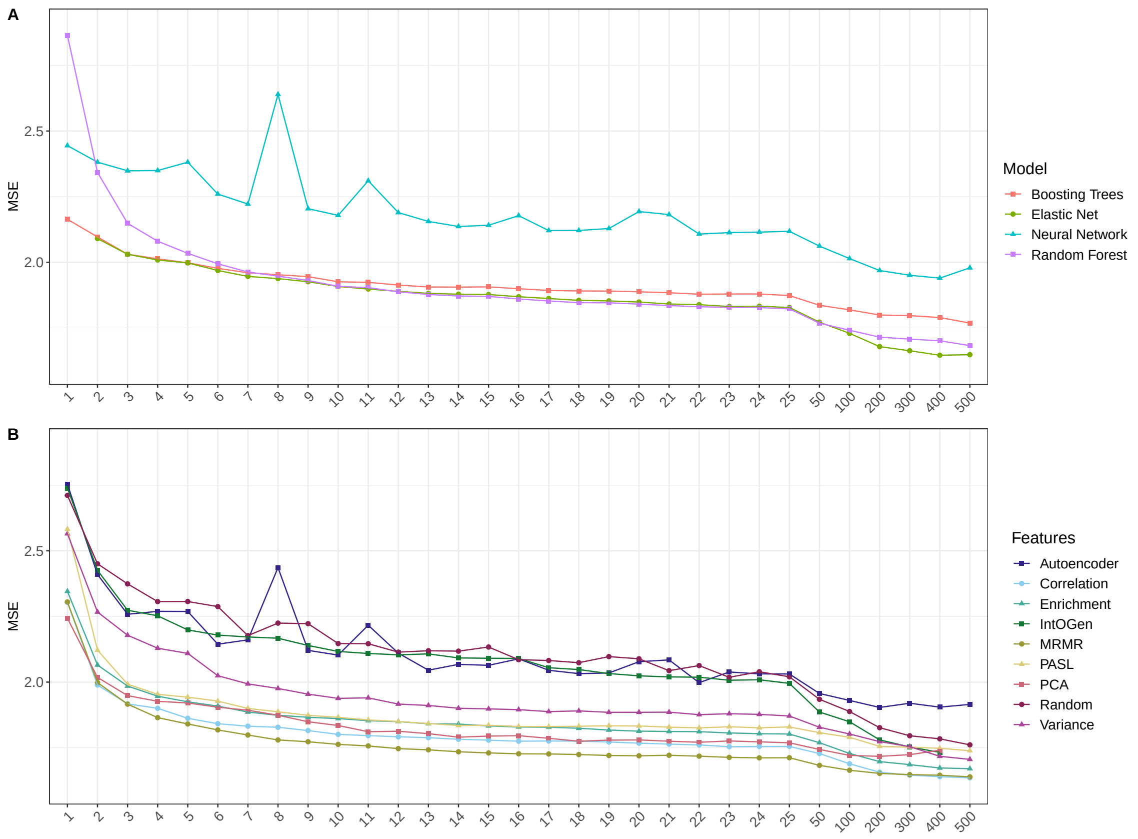


Figure 7.2: Average test performance. In this figure, the average test set mean-squared error (MSE) is depicted. In Figure A, we show the mean-squared error averaged across all drugs and DR techniques, yielding one best-performing ML model for each investigated k. To generate Figure B, we averaged across all drugs and ML methods, resulting in one best performing DR technique per k. For boosting trees, elastic net, and random forests, the average was calculated using the results for all 179 drugs. The high runtime of the neural networks did not allow us to fit models for all 179 drugs. Instead, we only trained models for 50 drugs. Consequently, the average is also only calculated across these 50 drugs. In Appendix Figure D.37, we depict the results of all methods for the subset of 50 drugs. This figure shows the same trends. Note that for some settings, it was not possible to train models: The elastic net cannot be trained for $k = 1$ since the used R package *glmnet* solely supports $k \geq 2$. The IntOGen cancer driver list encompasses 476 features. Thus, no results for $k = 500$ can be generated. The number of principal components of the *stats* R package is limited by the number of available input samples, and since most of the CV training folds contain less than 500 samples, no results for $k = 500$ exist.

increasing dominance of elastic net models for larger input feature sizes. With an increasing size of the feature sets, the redundancy of the features increases while their relevancy decreases. Potentially, this circumstance is more easily ignored by

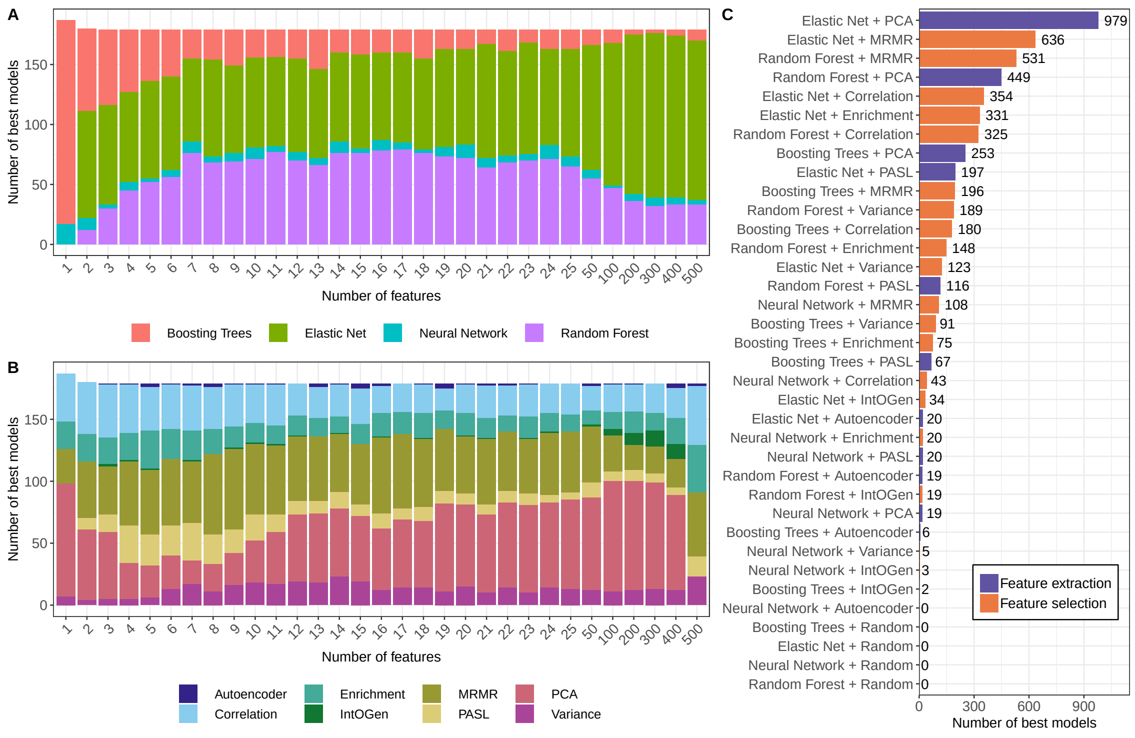


Figure 7.3: Best-performing settings for each drug. The plots in this figure depict the best-performing setting for each investigated drug and each k .

elastic net models than random forests since random forests randomly select features in each node, i.e., they might have to include less informative features in some splits. From the investigated DR techniques (cf. Figure 7.3B), PCA and MRMR are the most dominant ones, with an increasing performance advantage of PCA for larger k. In agreement with these results, Figure 7.3C reveals the combination of elastic net and PCA as most successful combination.

In the previously discussed Figures 7.3A to C, we presented the best-performing settings for each k. In Figures 7.4A to C, we depict the same results limited to only the k that achieved the best performance. We can see that for almost the complete set of drugs, a feature set size ≥ 50 is advantageous, while the mode of the distribution is reached at $k = 300$. The best-performing combination is once again PCA in front of the elastic net. However, except for the random FS, almost all DR techniques scored best for some drugs.

Up to this point, we compared different ML methods combined with various DR techniques and identified the best-performing settings using the MSE as an error measure. However, the raw MSE value cannot tell us how good the models actually

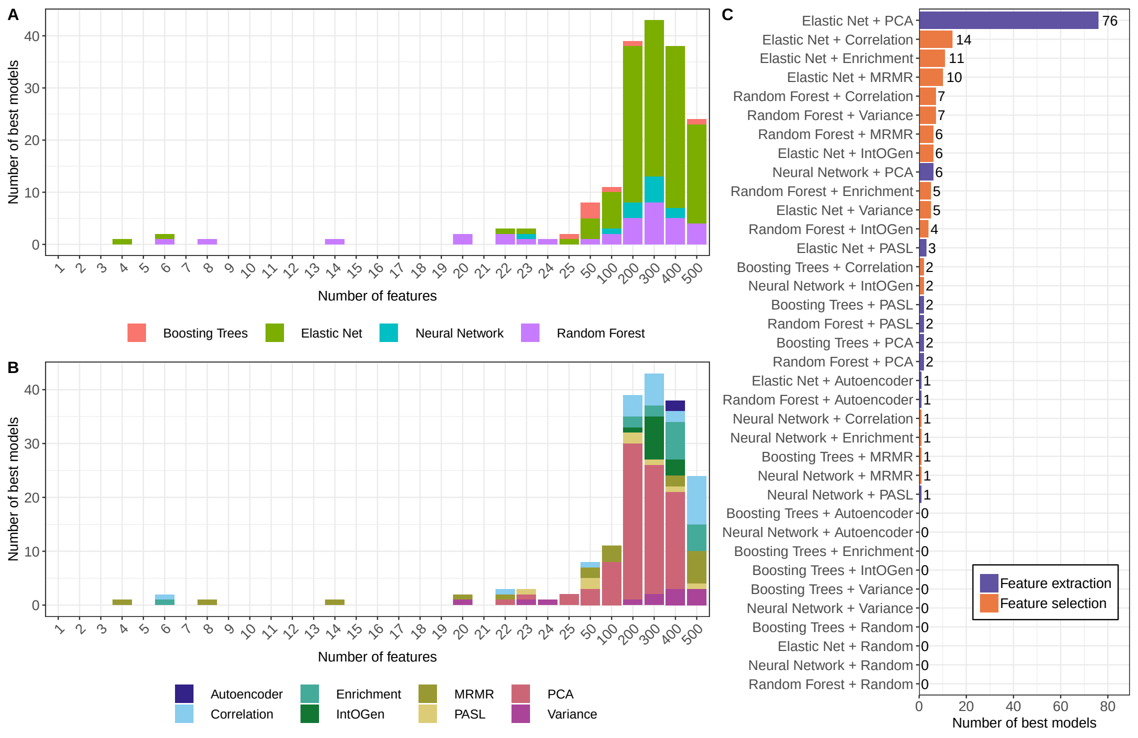


Figure 7.4: Best-performing settings for each drug. The plots in this figure depict the best-performing setting for each investigated drug the best k .

are. To this end, we can employ a baseline error, such as a simple dummy model that always predicts the mean response of the training samples. In Figure 7.5A, we plot the ratio between the test MSE of the best-performing ML/DR combination per k of a drug and the corresponding dummy model. All models are superior to the baseline, and for 80% of models, the improvement equals at least 20%, i.e., $\frac{MSE}{Baseline} \leq 0.8$. For 18% of models, the improvement is even 40% at least.
Figures 7.5B to D show analogous results for comparisons between models of different complexity in terms of k (B), Pearson correlation and PCA (C), and Pearson correlation and literature-based FS with IntOGen (D). In 7.5B, we note that for the majority of models (63%), the MSE increase by using fewer features ($k < 300$) is rather small ($< 10\%$), indicating that model complexity can be reduced significantly without major performance loss. Similarly, the comparison between PCA (best-performing FS) and Pearson correlation points towards the same direction: For 52% of models, the increase in error by using Pearson correlation instead of PCA is rather small (<10%). The Pearson correlation coefficient even outperformed PCA for 37% of models. From Figure 7.5D, we can learn that a simple literature-based

FS is not advisable. The Pearson correlation coefficient outperforms the IntOGen FS for 93% of models.

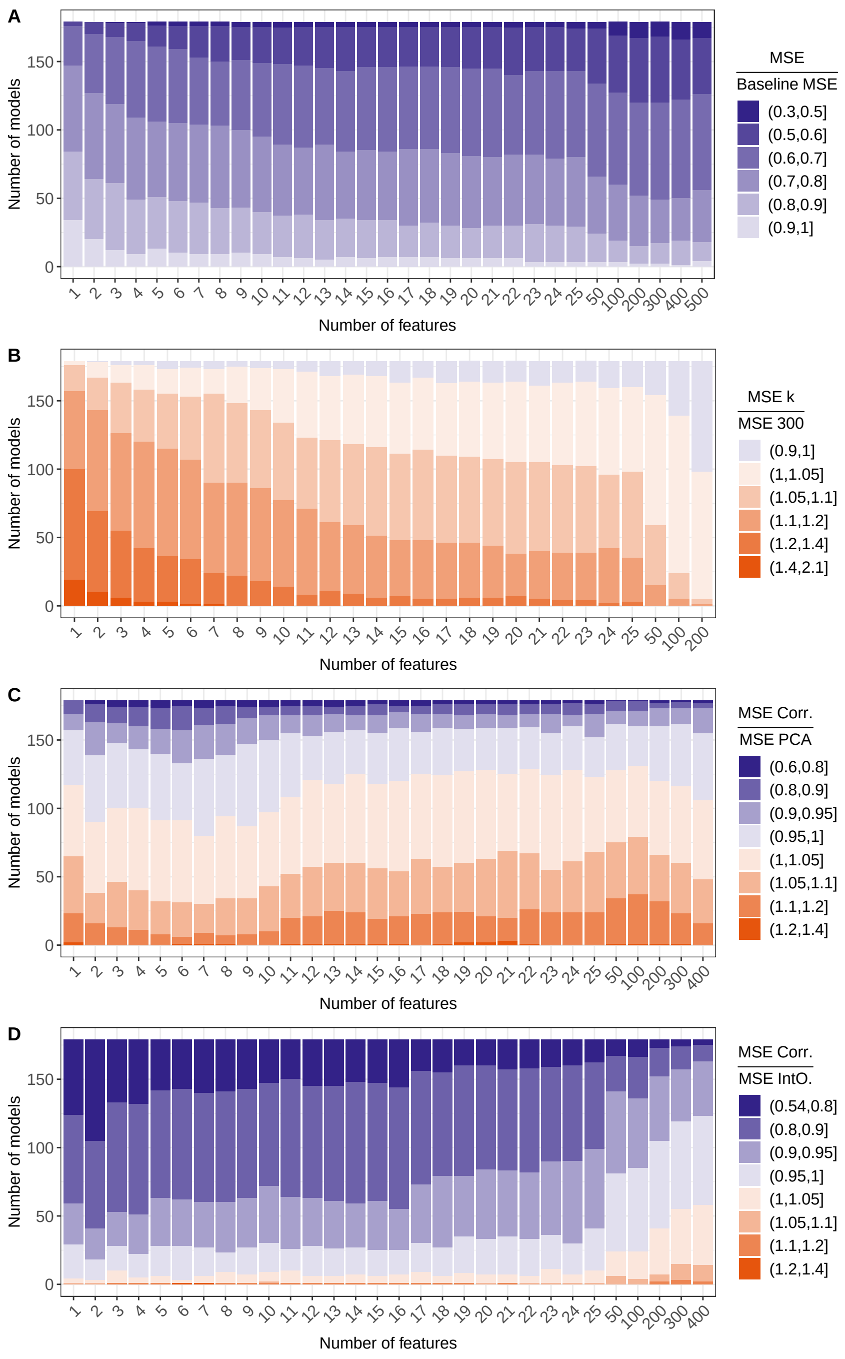


Figure 7.5: Error ratio. This figure depicts performance comparisons between different models. For comparing the models we employ the ratios between their respective test MSEs. I

Addressing regression imbalance Throughout Chapters 5 and 6, we illustrated that classifiers and regressors trained on the GDSC database suffer from performance issues because of the under-representation of sensitive cell lines, i.e., they exhibit a low sensitivity (classification) and a high MSE (regression) for those cell lines. Moreover, we have shown how to counteract these issues, known as class and regression imbalance, by introducing sample-specific weights or pursuing upsampling strategies.

Another straightforward implementation of a countermeasure against this class or regression imbalance is using error measures emphasizing the importance of the underrepresented class. In Figure 7.6, we depict the effects of one such approach on the test performance of the best-performing model: On the left, we employed the conventional CV MSE to select the hyperparameters during the CV and the best-performing setting per drug . On the right, we used a weighted MSE instead, i.e., we separately calculated the MSE of the sensitive and resistant cell lines and then averaged these two. Note that we binarized the drug response data using the described approach by Knijnenburg et al. [27] (cf. Chapter 5). We can observe that the MSE of the sensitive cell lines decreases considerably at the cost of a slight increase in MSE for the resistant cell lines, which is what we already noted in Chapters 5 and 6.

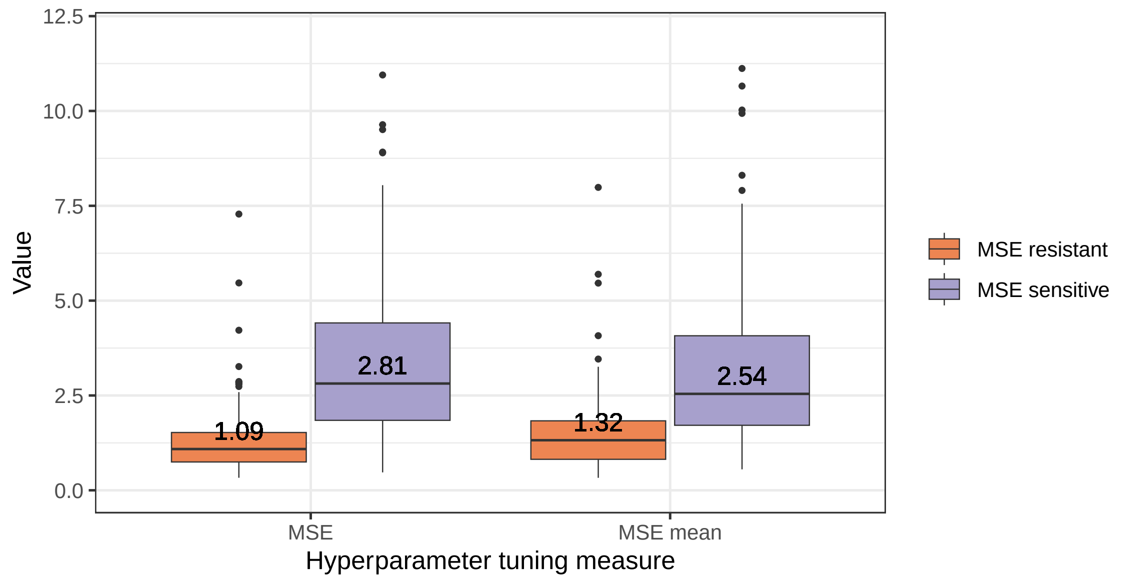


Figure 7.6: Effect of different error measures during hyperparameter tuning. This figure depicts the test MSE of the best-performing model for sensitive and resistant cell lines using the conventional MSE (left) for hyperparameter tuning and best-performing setting selection versus a weighted MSE instead (right). For the weighted version, we separately calculated the MSE of the sensitive and resistant cell lines, and then averaged these two.

7.3.3 Model and feature interpretability

In Chapter 4, we discussed that interpretability, i.e., the amount by which humans can understand an ML model, is a means to generate trust in ML. Especially for ML approaches in medicine, and therefore also drug sensitivity prediction methods, the quantification of interpretability is indispensable since these approaches should serve as decision support for human-centred affairs. In drug sensitivity prediction, these models are also used to gain insights into biomarkers of disease mechanisms and treatment success or failure, fostering basic medicine and pharmacy research alike. Thus, rendering these models amenable to human interpretation increases their usefulness.

In Chapter 4, we propose a taxonomy of interpretability based on definitions by Lipton [150] and Imrie et al. [146] (cf. Chapter 4, Figure 4.3). In that chapter, we also provide a detailed assessment of the existing drug sensitivity prediction models concerning interpretability. In this chapter, we limit ourselves to discussing some considerations that may arise before opting for a particular model.

The ML models and DR techniques of this benchmarking exhibit different levels of interpretability. If the original feature space already corresponds to understandable entities, e.g., genes, FS techniques are typically more interpretable than FE techniques since FE involves transformations of the original feature space, potentially obscuring meaning. However, FE techniques such as PASL try to increase interpretability in terms of transparency by introducing components corresponding to real-world entities, e.g., pathways. Indeed, this may help to elucidate high-level biological processes that are otherwise difficult to detect. From the four employed ML algorithms, neural networks are the least inherently interpretable ones (cf. Table 7.3 for a rough overview): they involve - even in simple network structures - various non-linear transformations of the input data. As a remedy, approaches for adding post-hoc explanations (i.e., explainability) are developed (cf. Chapter 4), giving rise to the branch of explainable artificial intelligence (XAI).
Usually, interpretability is not considered in isolation when opting for an ML model. For example, the bias-variance tradeoff (cf. Chapter 4) also plays a crucial role. Here, neural networks are a favorable approach if large amounts of data are available because they essentially incur no bias. The results of this chapter and the previous chapters do, however, not indicate that neural networks are particularly recommendable for drug sensitivity prediction with the presented data.

Table 7.3: Interpretability of ML methods. We assess the interpretability of the four ML methods using the terms transparency and explainability from the taxonomy of interpretability we proposed in Chapter 4. Briefly, transparency describes the inherent interpretability of an ML model and can roughly be approximated by its complexity. Explainability denotes any form of interpretability that is generated post hoc using explanations or justifications that were not part of the model. In this table, we solely list the explainability methods that are readily available in the used R/Python packages (cf. Table 7.2).

Model	Transparency	Explainability
Elastic net	+ easily interpretable feature coefficients	• feature importance: absolute value of coefficients; sign of coefficient denotes impact direction
Random forest	+ easily interpretable decision splits – typically large number of trees	• feature importance: error improvement obtained from splits using certain feature; error increase when feature is randomly perturbed • samples similar to given input: training samples reaching same leaf nodes
Boosting trees	+ easily interpretable decision splits – typically large number of trees – trees affect predictions to varying degree	• feature importance: error improvement obtained from splits using certain feature; error increase when feature is randomly perturbed
Neural network	– typically thousands of model parameters – complex, multi-layered computations	

7.4 Discussion

In this chapter, we presented the results of a large-scale benchmarking study. In total, we fitted over 16,000,000 models using four ML methods (elastic net, boosting trees, random forests, neural networks) combined with nine DR techniques (random, literature-based, variance, correlation, enrichment, MRMR, PCA, PASL, autoencoder). We identified the elastic net, a regularized linear model, as the top-performing ML method throughout all conducted analyses, i.e., in terms of statistical performance, runtime, and interpretability. Random forests were only slightly less performant. Despite being the most complex tested model, neural networks were not competitive with the other methods. For the investigated DR techniques, we observed the following: elastic nets should be combined with the FE technique PCA to achieve the lowest error. However, the MRMR FS was on par with PCA for most analyses. Moreover, the features obtained by MRMR are easier to interpret than PCA-derived ones. The least-performing DR techniques were the IntOGen

driver list, the random FS, and the autoencoder.
Overall, our results imply that relatively simple ML methods suffice to achieve reasonable predictions of anti-cancer drug sensitivity using cell line data. This finding also seems to align well with a recent study by Kapoor and Narayanan [254]. They identify data leakage, i.e., the inappropriate handling of data before, during, or after the model training process, as a root cause of overoptimistic claims about the performance of complex ML methods in comparison to simple ones [254]. However, we recognize several limitations of our benchmarking study. Firstly, we solely focused on gene expression data since it is known to be the most informative omics data type [17]. Including other omics data as features could induce a different result since more complex ML methods and DR techniques may more accurately capture intricate biological processes between omics types. Generally, one can argue that we benchmarked only relatively basic approaches for both ML methods and DR techniques. Moreover, the results from Chapters 5 and 6 also show that advanced modelling can help improve upon basic models, especially for the underrepresented class of sensitive cell lines. In this chapter, however, we show that by employing a different error measure during the tuning of basic methods, we can also improve the performance for these cell lines.
In the last chapter of this thesis, when we conclude the complete work presented therein, we will deal with these issues once more. We believe that the results of this study may serve as a guide for developing future models.

8 Reliable anti-cancer drug sensitivity prediction and prioritization

In the previous three chapters, we already addressed various unresolved issues for anti-cancer drug response prediction using supervised ML classification and regression techniques. Our newly developed methods, and most related works from the literature, predict a single class or single continuous drug response value for one specific cell line, i.e., they return point predictions. By comparing these point predictions to the known actual values, the overall model performance during training, validation, and testing can be assessed using conventional error measures such as MSE for regression or sensitivity and specificity for classification. While a thorough evaluation on a test set can already hint towards the reliability of a model, we have neither a guarantee nor an estimation for the model uncertainty for a new sample, i.e., we do not know if a prediction for a new sample is likely going to be close to its actual but unknown value. In real-world healthcare applications, the described scenario represents the standard use case. Thus, a reliability estimate or guarantee is required for a translation of computational methods into actual medical decision-making.

From the related methods, only Fang et al. already integrate some form of reliability estimation for their predictions (see Table 8.1). They estimated prediction intervals with a quantile regression random forest [31]. Intuitively, the length of the interval corresponds to the reliability of the prediction. While their approach highlights the significance of reliability estimation, it is limited to RF regression. Moreover, their approach does not give a confidence level or guarantee. In the drug discovery and toxicity prediction domains, conformal prediction (CP) recently gained popularity [202, 255, 256, 257] (see Chapter 4 for a detailed description of CP). In this chapter, we introduce the corresponding concepts into the drug sensitivity prediction domain. Therefore, we developed a CP-based pipeline to reliably predict anti-cancer drug responses for classification, regression and combined classification and regression methods such as SAURON-RF. Moreover, we then proceed to shift our focus from reliable drug sensitivity prediction to drug prioritization. Here, drug prioritization is defined as the task of identifying and subsequently sorting the list of effective drugs for each sample. To allow for this shift, a drug sensitivity measure that is comparable across drugs is necessary. To this end, we propose a novel drug sensitivity measure, the CMax viability, which is based on clinically relevant drug concentrations as described in [258]. As mentioned above, we define drug prioritization as the task of identifying and subsequently sorting the list of recommendable drugs. As such, it is similar in spirit to drug recommendation [157]. He et al. define drug recommendation as the task of correctly ranking the k most efficient drugs. Indeed, we can formulate drug prioritization as drug recommendation with k being an already known, cell line-specific value given by the number of drugs that were effective for a cell line. However, the implementations of drug recommendation by He et al. [157], and Liu et al. [131] have some major drawbacks compared to our approach. Firstly, they cannot evaluate whether they only identified effective drugs. Secondly, they do not predict sensitivities directly. Thirdly, they provide no certainty estimation.

In the following, we briefly describe a quantile regression algorithm for SAURON-RF needed to render it amenable to CP. Moreover, we provide a multi-class extension of SAURON-RF to investigate whether a more fine-grained discretization of response values could increase performance. Then, we present our novel drug sensitivity measure, the CMax viability, which enables a straightforward prioritization of drugs because of its across-drug comparability. Finally, we depict how to apply CP to reliably predict anti-cancer drug responses and perform drug prioritization based on these predictions.

Table 8.1: Comparison between different tools for drug sensitivity prediction with respect to certainty estimation and prioritization. In this table, we compare the existing drug sensitivity prediction approaches with respect to the used methodology and drug sensitivity measure. Moreover, we assess whether the methods provide certainty estimates and perform drug prioritization.
* Rahman et al. [155] use the jackknife-after-the-bootstrap approach but did not employ it to deliver reliable predictions.
** He et al. [157] and Liu et al. [131] perform drug recommendation, which is similar but not identical to prioritization.

Name and author	Methodology	Drug sensitivity measure	Reliability	Prioritization
Menden et al., 2013 [152]	neural network	IC50	✗	✗
Zhang et al., 2015 [153]	dual-layer integrated drug-cell line similarity network	activity area, IC50	✗	✗
LOBICO by Knijnenburg et al., 2016 [27]	integer linear program delivering Boolean rules	binarized IC50	✗	✗
Stanfield et al., 2017 [166]	cell line and drug proximity networks	binarized IC50	✗	✗
SRMF by Wang et al., 2017 [154]	similarity regularized matrix factorization	IC50, activity area	✗	✗
HARF by Rahman et al., 2017 [155]	random forest augmented with cancer types	AUC	(✗)*	✗
HNMDRP by Zhang et al., 2018 [167]	similarity networks	IC50	✗	✗
Matlock et al., 2018 [156]	model stacking	AUC	✗	✗
KRL by He et al., 2018 [157]	kernelized rank learning	normalized IC50	✗	(✓)**
RWEN by Basu et al., 2018 [158]	response-weighted elastic net	AUC	✗	✗
CDRscan by Chang et al., 2018 [159]	convolutional neural networks	IC50	✗	✗
QRF by Fang et al., 2018 [31]	quantile regression random forest	activity area	✓	✗
NCFGER by Liu et al., 2018 [160]	neighbor-based collaborative filtering with global effect removal	IC50	✗	✗
DeepDR by Chiu et al., 2019 [161]	neural networks	IC50	✗	✗

Continued on next page

Table 8.1 – continued from previous page

Name and author	Methodology	Drug sens. meas.	Reliability	Prio.
Deep-Resp-Forest by Su et al., 2019 [168]	deep cascaded forest	binarized activity area, binarized IC50	✗	✗
Dr.VAE by Rampášek et al., 2019 [129]	semi-supervised generative modeling based on variational autoencoders	binarized area above the dose-response curve	✗	✗
netBITE by Oskooei et al., 2019 [162]	biased tree ensemble	IC50	✗	✗
Deng et al., 2020 [163]	neural network	normalized Actarea	✗	✗
PathDSP by Tang et al., 2021 [164]	neural network	IC50	✗	✗
MERIDA by Lenhof et al., 2021 [21]	integer linear program delivering Boolean rules	binarized IC50	✗	✗
GraphDRP by Nguyen et al., 2022 [165]	neural network	normalized IC50	✗	✗
PPORank by Liu et al., 2022 [131]	deep reinforcement learning	normalized IC50	✗	(✓)**
SAURON-RF by Lenhof et al., 2022 [29]	simultaneous regression and classification random forest	IC50 and binarized IC50 simultaneously	✗	✗
reliable SAURON-RF by Lenhof and Eckhart et al., 2023 [33]	simultaneous regression and classification random forest using CP	IC50 and binarized IC50 simultaneously, CMax viability and binarized CMax viability simultaneously	✓	✓

Authors' contributions
This chapter is based on the preprint *Reliable Anti-Cancer Drug Sensitivity Prediction and Prioritization* [33] in terms of content and text. Therein, the reliable SAURON-RF drug sensitivity prediction and prioritization pipeline using CP is presented. This pipeline represents the follow-up work to the SAURON-RF paper. I conceived the idea for the study and in particular for employing CP in the context of drug sensitivity prediction. Lisa-Marie Rolli implemented an initial version of the CP pipeline in her bachelor's thesis, which I supervised. The CP framework was then extended in her work as a student assistant at our chair. Lea Eckhart developed the novel drug sensitivity measure, which I then discretized for usage with SAURON-RF. I also implemented the novel extensions of SAURON-RF, i.e., the multi-class extension as well as the quantile regression functionality. Moreover, I drafted the manuscript. The computational experiments were jointly performed by me, Lea Eckhart, and Lisa-Marie Rolli. Hans-Peter Lenhof and Andrea Volkamer supervised the study. All authors discussed the results and commented on the manuscript.

8.1 Conformal drug sensitivity prediction and prioritization

A prerequisite for translating ML models into healthcare decision support systems is the creation of trust in their predictions. We developed and implemented a conformal prediction (CP) pipeline to address this demand for drug sensitivity prediction and prioritisation. Since we demonstrated (cf. Chapter 6) that joint classification and regression methods outperform regression and classification alone for drug sensitivity prediction, the framework is able to handle classification, regression, and joint classification and regression methods.

8.1.1 SAURON-RF extensions

SAURON-RF represents a possibility of performing classification and regression simultaneously. To that end, it pursues the strategy to augment the canonical regression random forest algorithm with class information for the training samples (cf. Chapter 6 for a detailed description). In summary, SAURON-RF employs the canonical regression random forest algorithm for model fitting with a continuous response (e.g., IC50 values). However, a binary response vector, e.g., partitioning into sensitive and resistant samples, is also used as input to calculate sample-specific weights and to weight the regression predictions of the trees. Here, we present two extensions to SAURON-RF: Firstly, we adopt the quantile regression algorithm described by Meinshausen [32] to SAURON-RF. By doing so, we enable the estimation of reliabilities for our predictions and, in particular, the implementation of a combined regression and classification CP framework. Secondly, we enable the processing of more than two classes to allow for a more fine-grained analysis of sensitivity levels.

Quantile regression for SAURON-RF

The goal of supervised learning algorithms is to express the relationship between a predictor variable, e.g., in our case, the P-dimensional random variable X, and the real-valued response variable Y, such that the resulting model approximates Y with minimal error. To this end, standard regression algorithms typically employ a squared-error loss function with which the conditional mean $E(Y|X = x)$ is estimated [32]. Random forests also approximate the conditional mean [32]. However, there exist cases in which not only the conditional mean but the complete conditional distribution $F(y|X = x)$ is of interest, e.g., reliability estimation or outlier detection [32, 259]. In our application case, we are indeed interested in estimating the dispersion of response values to assess the reliability of our prediction and to obtain a drug response value for a specific cell line that is unlikely to be exceeded. With quantile regression, such questions can be addressed [260]. For this purpose, Meinshausen proposed quantile regression forests, a generalisation to random forests able to estimate the conditional distribution function $F(y|X = x)$. We present an adapted version of this quantile regression algorithm for SAURON-RF in the following.

Let the conditional distribution function $F(y|X = x)$ be defined by the probability that Y is at most y for X equal to x, i.e.,

$$F(y|X = x) = \Pr(Y \leq y|X = x). \tag{8.1}$$

The α-quantile for $X = x$ is then defined as the minimum y for which the conditional distribution function is at least α:

$$Q_\alpha(x) = \inf\{y : F(y|X = x) \geq \alpha\} \quad . \tag{8.2}$$

Hence, we need an estimate of the conditional distribution function to perform quantile regression. Meinshausen shows that this is indeed possible with random forests by interpreting them as proposed by Lin and Jeon, who view them as an adaptive neighbourhood classification or regression algorithm [261]. In particular, Meinshausen employs the fact that the final prediction of an ordinary random forest is an estimate of the conditional mean and that it can be viewed as a weighted sum of the response values of the training observations. To this end, let $\mathbf{y} \in \mathbb{R}^N$ be the response vector containing response values of N samples. Then, the final prediction of the ordinary RF can be expressed as

$$E(Y|X = x) = \hat{f}(x) = \sum_{i=1}^{N} w_i(x) \cdot y_i \tag{8.3}$$

with $w_i(x)$ representing a forest-wide weight for each training sample $i \in \{1, \ldots, N\}$ (cf. Meinshausen [32] for definition in usual random forests). In contrast, we expressed the final prediction of SAURON-RF as a weighted average of the tree predictions $\hat{f}_b(x), \forall b \in \{1, \ldots, B\}$ (cf. Equation 6.2 in Chapter 6):

$$\hat{f}(x) = \sum_{b=1}^{B} w_b(x) \cdot \hat{f}_b(x) \quad . \tag{8.4}$$

Here, $w_b(x)$ is a tree-specific weight. The equivalence of Equation 8.4 and 8.3 can however also be established. To this end, let $\delta(v)$ be the set of bootstrap samples that belong to the node v and let w_j^v be the node-specific sample weight for the bootstrap sample j as defined in Chapter 6 Equation 6.5. Moreover, let w_{i*}^v be the node-specific sample weight of the training samples, which can be derived from the weights of the bootstrap samples of the trees by calculating

$$w_{i^*}^v = \sum_{j \in \delta(v): j=i} w_j^v, \forall i \in \{1, \dots, N\}. \tag{8.5}$$

We now start with Equation 6.2 and transform it to 8.3

$$\begin{aligned}
\sum_{b=1}^{B} w_b(x) \cdot \hat{f}_b(x) = & \sum_{b=1}^{B} w_b(x) \cdot \sum_{n \in \delta(\mu_b)} w_n^{\mu_b} \cdot y_n \\
\overset{\text{introduce } I_{\delta(\mu_b)}(i) = \begin{cases} 1, \text{ if sample } i \text{ is in leaf node } \mu_b \\ 0, \text{ otherwise} \end{cases}}{=} & \sum_{b=1}^{B} w_b(x) \cdot \sum_{i=1}^{N} I_{\delta(\mu_b)}(i) \cdot w_{i^*}^{\mu_b} \cdot y_i \\
= & \sum_{b=1}^{B} \sum_{i=1}^{N} w_b(x) \cdot I_{\delta(\mu_b)}(i) \cdot w_{i^*}^{\mu_b} \cdot y_i \\
= & \sum_{i=1}^{N} \sum_{b=1}^{B} w_b(x) \cdot I_{\delta(\mu_b)}(i) \cdot w_{i^*}^{\mu_b} \cdot y_i \\
\overset{\text{define } w_i(x) = \sum_{b=1}^{B} w_b(x) \cdot I_{\delta(\mu_b)}(i) \cdot w_{i^*}^{\mu_b}}{=} & \sum_{i=1}^{N} w_i(x) \cdot y_i \quad .
\end{aligned}$$

Given this equivalence, we can - in analogy to Meinshausen - estimate the conditional distribution function by

$$\hat{F}(y|X = x) = \sum_{i=1}^{N} w_i(x) \cdot I_{y_i \leq y} \tag{8.6}$$

with $I_{y_i \leq y}$ being 1 iff $y_i \leq y$ and 0 otherwise. Finally, the quantile regression forest algorithm for SAURON-RF reads as follows

1. Train the SAURON-RF regression random forest as explained in Chapter 4 Section 4.3.4 and Chapter 6 Section 6.1.2.

2. For a new sample x, trace a route from root to leaf for each tree $b \in \{1, \ldots, B\}$, which results in the set of reached leaf nodes $L = \{\mu_1, \ldots, \mu_B\}$.

3. For each $\mu_b \in L$, calculate the node-specific sample weights (cf. Equation 8.5) of all training samples $x_i, i \in \{1, \ldots, N\}$.

4. Then, average these weights across L to obtain a forest-wide weight of each training sample $i \in \{1, \cdots, N\}$, i.e.,

$$w_i(x) = \sum_{b=1}^{B} w_b(x) \cdot I_{\delta(\mu_b)}(i) \cdot w_{i^*}^{\mu_b}, \forall i \in \{1, \ldots, N\} \tag{8.7}$$

5. Now, an estimate of the distribution function $\hat{F}(y|X = x)$ can be determined for all $y \in \mathbb{R}$ by using Equation 8.6.

6. By plugging $\hat{F}(y|X = x)$ into Equation 8.2, calculate the estimate of the conditional quantile $\hat{Q}_\alpha(x)$, i.e., return the minimal response value y for which the estimate of the conditional distribution function $\hat{F}(y|X = x)$ is at least α.

Multi-class extension

For the initial version of SAURON-RF, we solely considered a binary division into sensitive and resistant cell lines, i.e., we gave definitions for the sample-weight functions of the binary case (see Chapter 6). However, especially for drug sensitivity prediction, allowing for a more fine-grained class division can be advantageous to more accurately reflect the biological variance and uncertainty of drug response. Thus, we provide straightforward extensions for the Equations 6.3 and 6.4.
Let $C = \{c_1, \ldots, c_k\}$ be a set of k classes. Furthermore, suppose that N_{c_j} with $j \in \{1, \ldots, k\}$ is the number of samples of class c_j. W.l.o.g., let c_k be the class containing the relative majority (mode) of samples. The *simple* sample weights can be determined by the formula

$$w_i^* = \begin{cases} 1, & \text{if sample } i \text{ belongs to } c_k \\ \frac{N_{c_k}}{N_{c_j}}, & \text{if sample } i \in c_j, \forall j \in \{1, \ldots, k-1\} \end{cases} \tag{8.8}$$

To define the *linear* and *quadratic* weight function for the multi-class setting, we additionally assume that the classes are ordered in ascending order of the thresholds that divide the corresponding class pairs. To this end, let $t_{j,r} \in \{t_{1,2}, \dots, t_{k-1,k}\}$ be the threshold that divides the samples from class j and r. Let $\mathbf{d}$ and $\mathbf{y}$ be the discrete and continuous drug response vectors respectively. The weight function in Equation 6.4 remains unaltered for samples belonging to class c_1 and c_k since these classes have only one neighbouring threshold. For all other samples, the distances from the two thresholds are averaged. In total, the following formula provides the sample weights

$$w_i^* = \begin{cases} \frac{|y_i - t_{1,2}|^g}{k \cdot \sum_{\forall n \in \{1,\dots,N\}: d_n = d_i} |y_n - t_{1,2}|^g}, & \text{if sample } i \text{ belongs to } c_1 \\ \frac{|y_i - t_{k-1,k}|^g}{k \cdot \sum_{\forall n \in \{1,\dots,N\}: d_n = d_i} |y_n - t_{k-1,k}|^g}, & \text{if sample } i \text{ belongs to } c_k \\ \frac{|y_i - t_{j-1,j}|^g + |y_i - t_{j,j+1}|^g}{k \cdot \sum_{\forall n \in \{1,\dots,N\}: d_n = d_i} |y_n - t_{j-1,j}|^g + |y_n - t_{j,j+1}|^g}, & \text{otherwise} \end{cases} \tag{8.9}$$

with $g \in \{1, 2\}$.

8.1.2 Definition of novel drug sensitivity measure

In Chapter 3, we thoroughly described a variety of drug sensitivity measures calculated from dose-response curves resulting from cell line viability assays. There, we already pointed out that established measures such as the IC50 and AUC values, as provided by the GDSC, are only comparable across cell lines but not across drugs. To address this demand, we propose a novel drug sensitivity measure called CMax viability. The across-drug comparability of the novel measure will then allow for a direct comparison of efficiency between different drugs applied to one particular cancer sample. We define the CMax viability of a drug as the viability of a cell line at the CMax concentration of that drug. Here, the CMax concentration is the peak plasma concentration of a drug after administering the highest clinically recommended dose [258]. The CMax viability can take values in the range $[0, 1]$, where 0 corresponds to no viability of cancer cells after treatment, and 1 indicates 100% viability. To derive the viability of a cell line at the CMax concentration, we determine the intersection point between the dose-response curve of the cell line

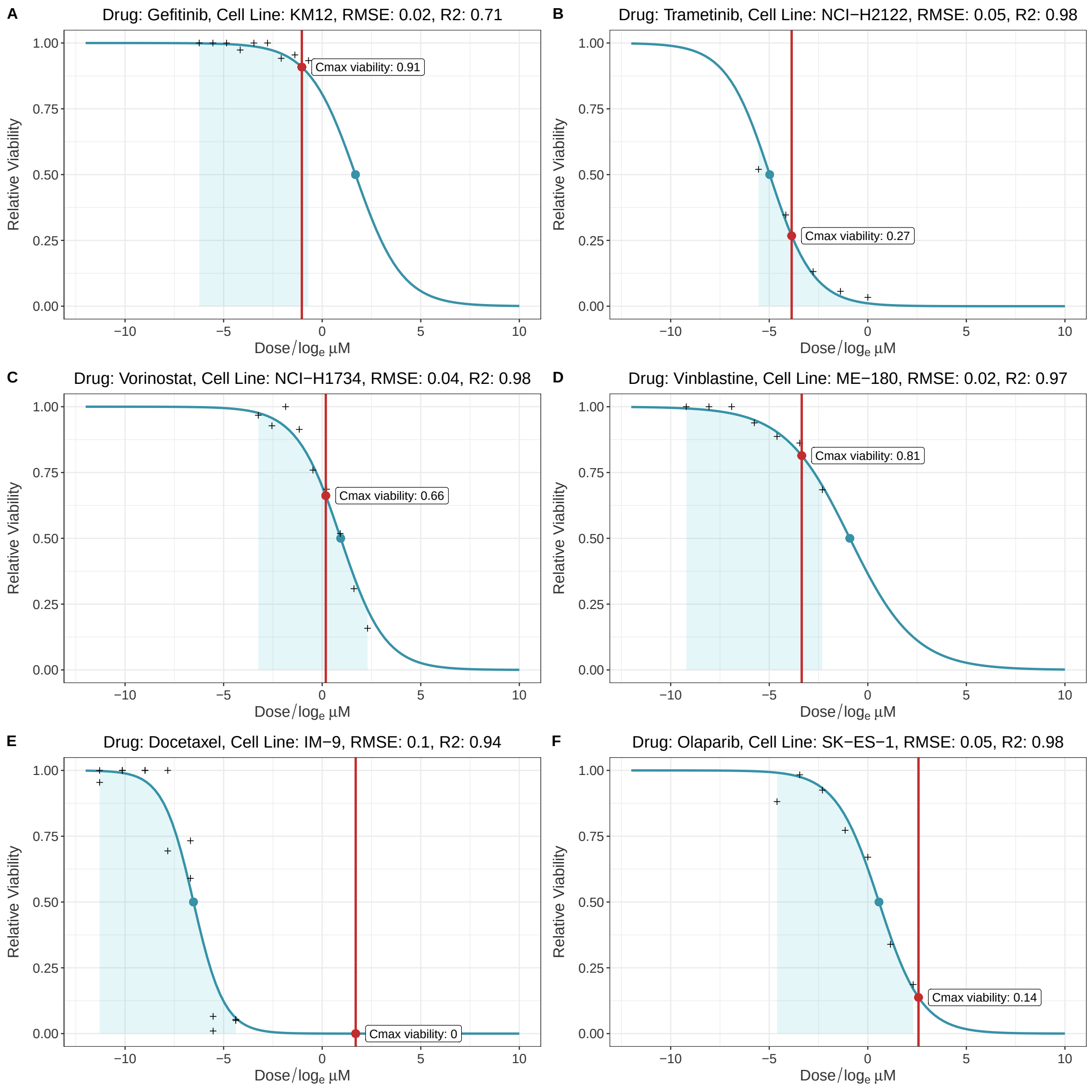


Figure 8.1: Exemplary dose-response curves for six drug and cell line combinations. This figure depicts six exemplary dose-response curves fitted with the multilevel mixed effects model by Vis et al. [99]. Here, the black crosses represent the actual dose-response measurements provided by the GDSC database. The blue-shaded area highlights the tested concentration range and the blue circle marks the IC50 concentration (cf. Chapter 3, Section 3.1.2 for a definition). The intersection point between the red vertical line (passing through the CMax concentration) and the dose-response curve marks the CMax viability (cf. Section 8.1.2 for the definition). Used abbreviations in plots: RMSE (Root Mean Squared Error), R2 (R-Squared)

calculated from raw viability data and the line parallel to the viability axis passing through the CMax concentration (see Figure 8.1 for examples).

8.1.3 Conformal prediction pipeline

In Chapter 4, we discuss ML and various CP variants in detail. In the following, we briefly summarise and describe the key aspects of inductive conformal prediction as we employ it to perform reliable drug sensitivity prediction and prioritization.
CP represents a mathematical rigorous certainty estimation framework applicable to all regression and classification ML methods provided that the latter supply a notion of (un)certainty. Given a user-specified maximal error rate $\alpha \in [0, 1]$, CP returns prediction sets (classification) or intervals (regression) that contain the true response with a certainty, also known as coverage, of almost exactly $1 - \alpha$. This is called marginal coverage property (see Chapter 4, Section 4.4.1). We designed and implemented a flexible CP framework in Python that can be used for regression, classification and joint regression and classification methods. In the following, we outline the functionality of our framework. The corresponding graphical overview of the framework applied to SAURON-RF is given in Figure 8.2.
Input of CP: In ML, we usually assume that our samples are drawn i.i.d. to guarantee the claimed properties of our methods. For the CP guarantee to hold, we only need to assume the exchangeability of the data, i.e., that the underlying joint probability distribution is invariant to finite permutations [197]. The training and testing of supervised ML methods typically requires at least two disjoint data sets: a training data set for parameter selection and a test set for the final evaluation of the trained model. CP demands a third disjoint data set, the calibration data set, used to derive a distribution on the (un)certainty of the trained model. Accordingly, the trained model needs to provide a notion of uncertainty (or certainty). In the case of SAURON-RF, we employ

$$1 - \frac{\#\text{trees that voted for predicted class}}{\#\text{trees}}$$

as a notion of uncertainty for classification. For regression, we quantify the dispersion of response values with quantile regression. In addition to this notion of uncertainty, the user has to specify a maximal allowed error rate α, which allows for a flexible certainty control. If a model cannot generate single-class prediction sets or narrow intervals given a strict error rate, increasing α, i.e., lowering the desired certainty, might still help to identify the most reliable trends.

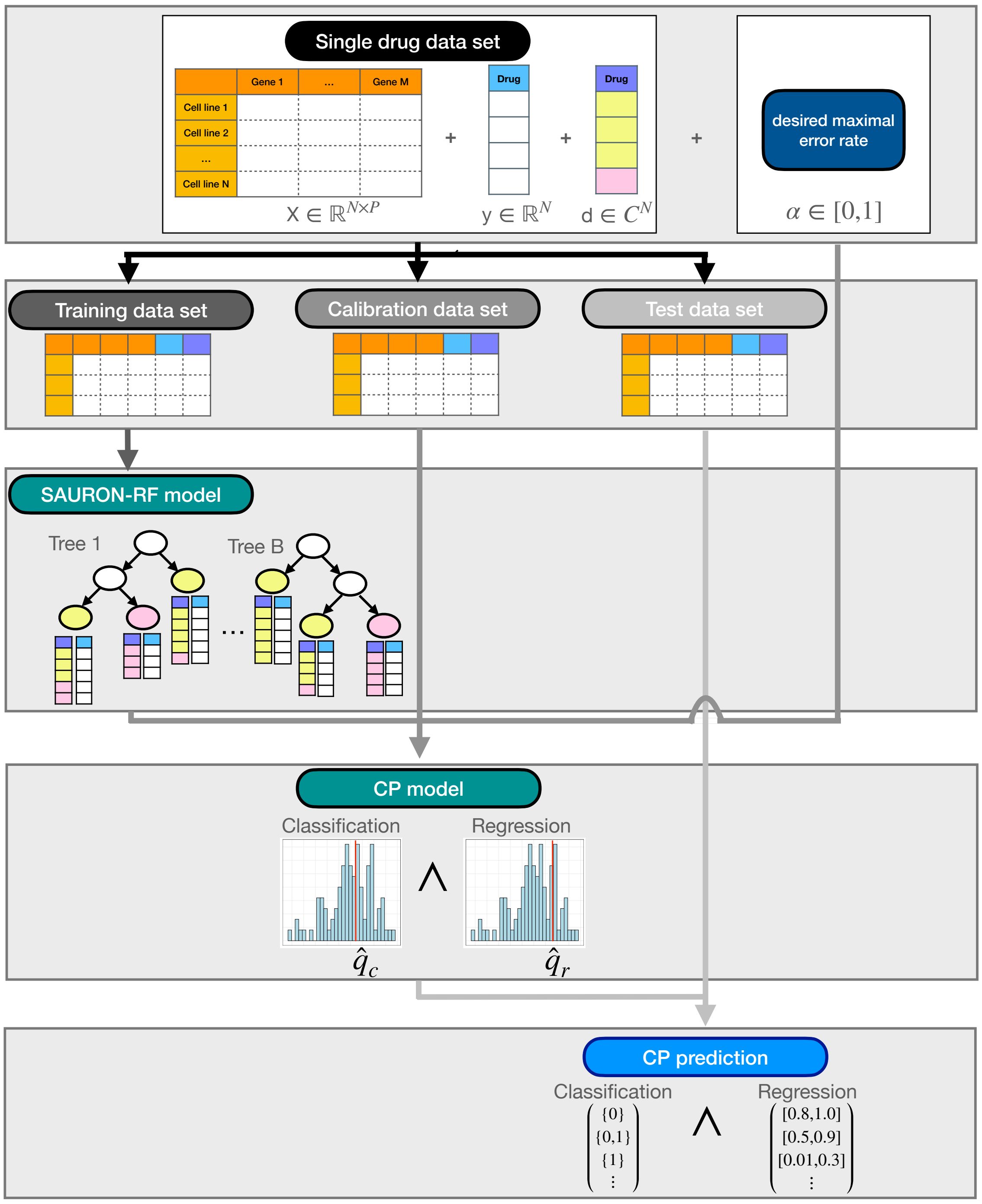


Figure 8.2: CP pipeline. This figure depicts how CP can help to perform reliable (simultaneous) regression and classification. At first, the given drug data set has to be split into three disjoint data sets: a training, a calibration, and a test set. The ML method, e.g., SAURON-RF, is then trained on the training data set. Afterwards, the resulting model is applied to the calibration data to derive a distribution of (un)certainty of the predictions. Together with the user-specified maximal allowed error rate α, this distribution is used to define a threshold that when appropriately applied to the test data set guarantees a certainty of $1-\alpha$ of the test set predictions.

Score functions: CP integrates a given notion of uncertainty in a score function, also called non-conformity score in the corresponding literature. By applying this score function to the calibration data set, a score distribution can be generated. For classification, we implemented three different score functions: True-class (TC), Summation (Sum), and Mondrian (Mon). For regression, we implemented a score function called Quantile (Qu). In Chapter 4, Section 4.4.2 and Section 4.4.3 we give their definitions. Given a score distribution and the maximal allowed error rate α, CP derives a threshold $\hat{q}$ that can be used to generate valid intervals or sets. More specifically, $\hat{q}$ is a modified $(1 - \alpha)$ quantile of the score distribution if the score function quantifies uncertainty (see Chapter 4, Section 4.4.1 for details). Note that it is possible to implement all of the mentioned score functions as uncertainty measures, i.e., high values correspond to high uncertainty, and low values correspond to high certainty.

Output of CP: After training the ML model on the training data set, and employing its notion of uncertainty in a score function to derive a score distribution on the calibration data set, the CP output for the test set can be generated. The trained ML model has to be applied to the test set, and the score function must also be evaluated. By combining $\hat{q}$ with the derived score per test set sample, the point prediction of the ML model can be exchanged with a valid prediction set (classification) or interval (regression). More specifically, CP returns prediction sets (classification) or intervals (regression) that fulfil the marginal coverage property. Some scores guarantee special versions of this property. The Mondrian score, for example, provides this coverage for every ground truth class, which is especially desirable when there is a considerable class imbalance present, as is the case for drug sensitivity prediction in cancer [29]. Our CP pipeline cannot only return prediction sets or intervals but also both simultaneously, making it amenable to joint classification and regression methods such as SAURON-RF. Moreover, when combining this capability with our novel drug sensitivity measure that is comparable across drugs, we can leverage the full potential of SAURON-RF and ultimately perform drug prioritization: we can first reliably identify effective drugs (classification) and then rank them by their predicted efficiency (regression) using the (upper limit of the) CP interval. We depict this application scheme in Figure 8.3.

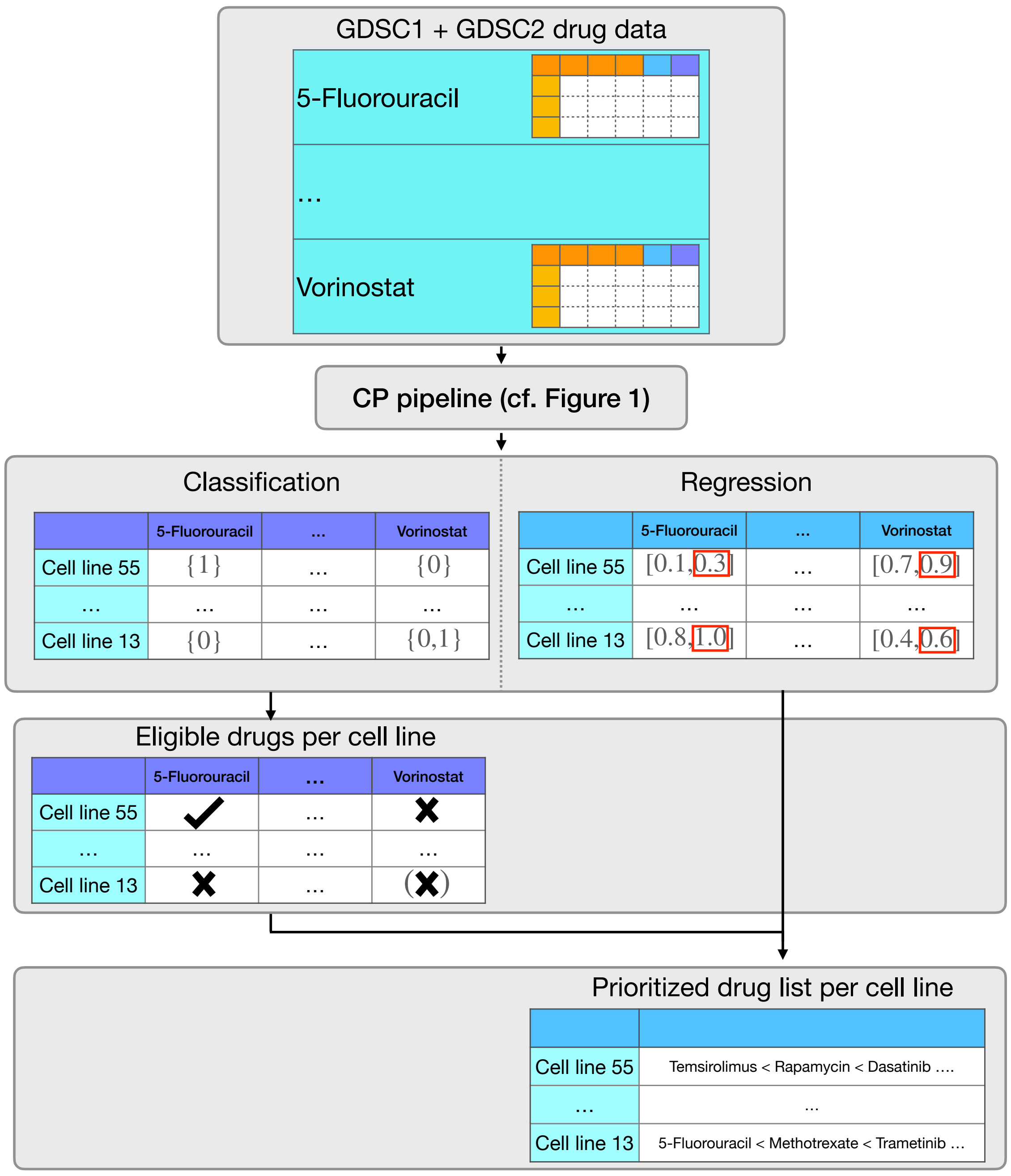


Figure 8.3: Drug prioritization pipeline. This figure shows a drug prioritization pipeline originating from combining the CMax viability with SAURON-RF and CP. The output of the CP pipeline (cf. Figure 1) deployed with our SAURON-RF method are sets for the classification task and intervals for the regression task. Here, sets that contain only one element indicate that we can be confident about the initial point prediction (single class) of the trained model. Thus, we can identify effective drugs by filtering for sets solely comprising the class corresponding to drug sensitivity (1: sensitive). Due to the across-drug comparability of the CMax viability, we can rank these drugs by their predicted efficiency using, for example, the upper limit of the regression interval that represents a value not being surpassed with high probability.

8.2 Data preparation

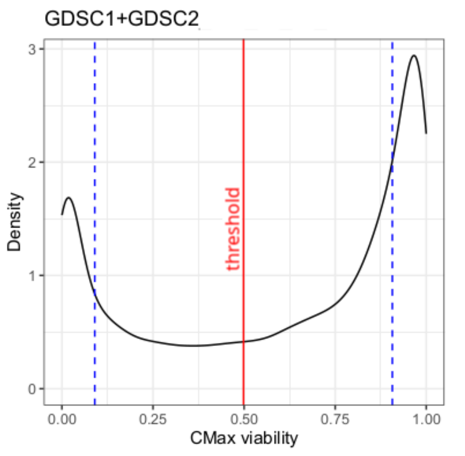
For all analyses in this chapter, we employ release 8.3 (June 2020) of the GDSC cancer cell line panel [19], which is the same version deployed for the analyses of Chapter 6. In particular, we again used the pre-processed gene expression values and the pre-computed logarithmized IC50 drug responses. In addition, we downloaded the raw viability data and obtained a list of CMax concentrations from [258] to calculate the CMax viability measure.

8.2.1 Drug response processing

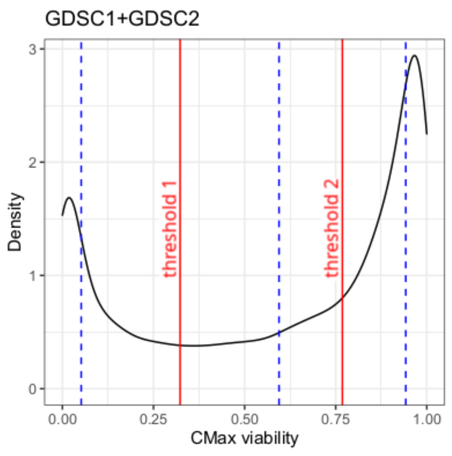
In our experiments, we use two different drug sensitivity measures, i.e., the logarithmized IC50 value and the CMax viability, separately to fit our models. To achieve a fair performance comparison between the two measures, we restrict our analyses to drugs with availability for both. Thus, we considered 107 drugs from GDSC1 (60) and GDSC2 (47) in total. As a method that simultaneously performs classification and regression, SAURON-RF requires a continuous and discrete drug response vector as input. Therefore, we also derive discretized drug response vectors for both sensitivity measures.

IC50 value processing: As a continuous measure of drug sensitivity, we employ the logarithmized IC50 values provided by the GDSC. The corresponding binarized drug response was obtained by applying the custom R-script as described in Chapter 5. For each drug, we thereby derive one binarization threshold that divides the cell lines into sensitive and resistant ones, finally resulting in one binary drug response vector.

CMax viability processing: To calculate the viability at the CMax concentration, we first determined the dose-response curves for all cell line-drug combinations simultaneously with the multilevel mixed effects model by Vis et al. [99] using the raw drug sensitivity data from the GDSC. For each drug-cell line combination, we then identify the viability at which the corresponding dose-response curve passes through the line parallel to the viability (Y) axis through the CMax concentration



(a) One threshold



(b) Two thresholds

Figure 8.4: Threshold determination. This figure depicts the density of the CMax viability values and the derived thresholds (red lines) as well as the medoids (blue dashed lines) resulting from applying PAM.

of the drug. We call this CMax viability (see Section 8.1.2). Since SAURON-RF demands a discrete and a continuous drug response vector as input, we also discretize the CMax viabilities. In contrast to the IC50 data, we do not derive specific thresholds for each drug. Instead, we leverage the across-drug comparability of the viabilities to determine one threshold (binarization) or even several thresholds (discretizations such as threefold division) applicable to all drugs. To this end, we employ the partitioning around medoids (PAM) clustering algorithm (see Chapter 4, Section 4.3.2), which has already been used in drug sensitivity prediction to discretize GI50 values [17]. Using PAM on the complete set of available CMax viabilities across all drugs, we identify either two clusters or three clusters of cell lines. The mid-points between the clusters are discretization thresholds (cf. Figure 8.4). In the case of two classes, we then interpret the cell lines of the two clusters to be sensitive (1) and resistant (0). If we derive two thresholds, we interpret the clusters as sensitive, ambiguous, and resistant cell lines. Consequently, when we apply the discretization threshold(s) to the continuous CMax viabilities of a particular drug, we obtain a binary (two classes) or ternary (three classes) response vector. In our SAURON-RF analyses, we combine the continuous response vector of one drug either with the binary or the ternary response vector of that drug.

8.2.2 Model training

In all previous chapters, we were concerned with the prediction of continuous or discrete drug response values and an evaluation of these predictions from a drug-centric perspective. Specifically, this means that we assessed the model performance on a per-drug basis. For each drug, we predicted the sensitivities of the cell lines and then determined the model performance, e.g. we calculated the Pearson correlation coefficient between our predictions for one drug and the actual values of that drug. Eventually, we are interested in assessing model performance from a cell-line-centric perspective, i.e., for each cell line, we would like to identify and subsequently prioritize all suitable drugs. Given the across-drug comparability of our novel drug sensitivity measure, the CMax viability, we can now finally conduct such analyses. In the ensuing sections, we present results for two major settings: drug-centric analysis and cell line-centric analysis, which have been performed as described below.

Drug-centric analysis: To achieve a fair performance comparison between the IC50 values and the CMax viabilities, we only considered drugs where both values were available, which resulted in 107 (60 from GDSC1, 47 from GDSC2) potentially analyzable drugs. Here, one drug data set consists of the following triple: the gene expression matrix, the continuous response of a particular drug, and the discretized response of that drug. We partitioned each data set into a training (70%), calibration (15 %), and test (15%) set. The training set was further subdivided to serve as input for a 5-fold cross-validation (CV). Within each CV step, the fold usually employed as the test set is partitioned into a disjoint calibration and test set. If the discretized CMax viabilities for one drug contained only one class or consisted of an insufficient number of samples per available class, we discarded this drug for the CMax viability and the corresponding IC50 analyses (see Figure 8.5 for the GDSC2 binary case and Appendix Figures E.38 - E.40 for all other cases). In total, we could thus analyze 41 drugs for the binarized drug responses of GDSC1, 32 drugs for the binarized drug responses of GDSC2, 37 drugs for the ternary drug responses of GDSC1, and 28 drugs for the ternary drug responses of GDSC2. For each data set, the final model is trained on the complete training data, and the CP pipeline is applied accordingly afterwards. Here, we only report the results for the newer GDSC2 data set, which is based on an improved drug sensitivity assay. The results for the GDSC1 data set can be found in the Appendix Figures E.54 - E.64.

Cell line-centric analysis: When we shift the focus of the performance evaluation from a drug to an investigated cancer sample, we have to adjust the generation of our training, calibration, and test set. In particular, if we prioritize drugs for one particular cell line, this cell line must be previously unseen by each drug-specific model in the training process in order to derive an unbiased estimate of the drug prioritization capability. Since this has to hold for all cell lines in the test set, all drugs must share the cell lines in the test set. To ensure this for our analyses, we first determined the common cell lines between all investigated drugs for each database (GDSC1/GDSC), resulting in 243 common cell lines for the GDSC1 drugs and 609 common cell lines for the GDSC2 drugs. For each database, we then randomly sampled a test and calibration set from the common cell lines such that their sizes were approximately equal to the average sizes from our drug-centric analyses. The calibration and test sets each contained 121 cell lines for GDSC1 and 152 cell lines for GDSC2. The remaining cell lines are added to the training sets of the drugs. For this analysis, we decided to employ only drugs with at least 6% of samples in each class since models for drugs with a comparably low imbalance perform better than models trained on drugs with high class imbalance. Beyond that, the CP certainty guarantee also becomes more accurate the more calibration samples are available, see [198, 197], which is particularly important for the minority class. In total, we analyzed 25 drugs for GDSC1 and 25 drugs for GDSC2. Again, we report only the results for the GDSC2 data set here. The respective results for the GDSC1 data set can be found in Appendix Tables E.65 - E.75.

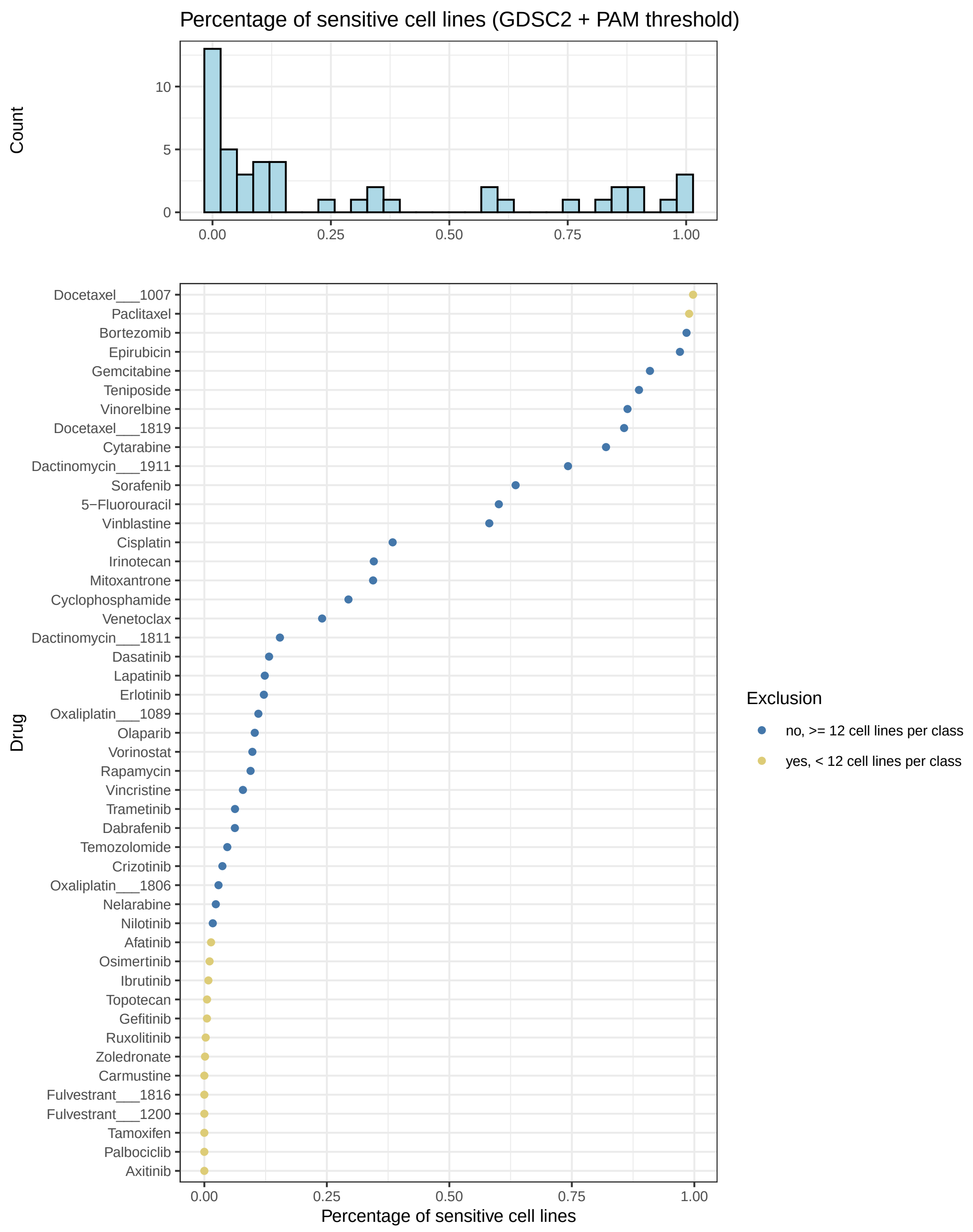


Figure 8.5: Percentage of sensitive cell lines of binarized CMax viability in GDSC2. The upper row of this figure shows a histogram for the percentage of sensitive cell lines across all available drugs. The lower row depicts the corresponding percentage for each drug. Moreover, the color of the points indicates which drugs had to be excluded for the analysis and why.

8.3 Implementation and data deposition

Using the previous implementation of SAURON-RF as basis, we implemented the extensions of SAURON-RF using Python 3 and the sklearn RandomForestRegressor package [182]. Similar to its predecessor, extended SAURON-RF including the CP framework can be called from the console with a single configuration file in JSON-format as input. The CMax viability values were determined using R. In particular, the gdscIC50 R package was employed for the fitting of the dose response curves [99]. The CP framework was implemented in Python 3. The respective code and data has been deposited in the publicly available GitHub: `https://github.com/unisb-bioinf/Conformal-Drug-Sensitivity-Prediction.git`.

8.4 Results

In the ensuing sections, we present the results of the drug-centric and the cell line-centric analysis. We start with an evaluation of the CP pipeline applied to the established logarithmized IC50 value as the drug sensitivity measure, i.e., with standard drug sensitivity prediction from a drug-centric perspective. Subsequently, we show the results of a drug-centric assessment for our novel drug sensitivity measure, the CMax viability. Here, we did not only assess a binary division of cell lines but also the proposed ternary division. Finally, we leverage the full potential of our pipeline and perform drug prioritization, i.e., we evaluate the capabilities of SAURON-RF together with the CMax viability and CP to reliably detect effective drugs and afterwards rank them by predicted efficiency.

8.4.1 Drug-centric analysis - drug sensitivity prediction

At first, we applied SAURON-RF without CP to the IC50 data. In Figures 8.6 and 8.7, we show the respective classification and regression performance on the test set. With an average sensitivity of 56%, specificity of 87%, Matthew's correlation coefficient (MCC) of 0.35, and mean-squared error (MSE) of 2.5 across all drugs, the performance is similar to what we and others observed previously [29, 21].

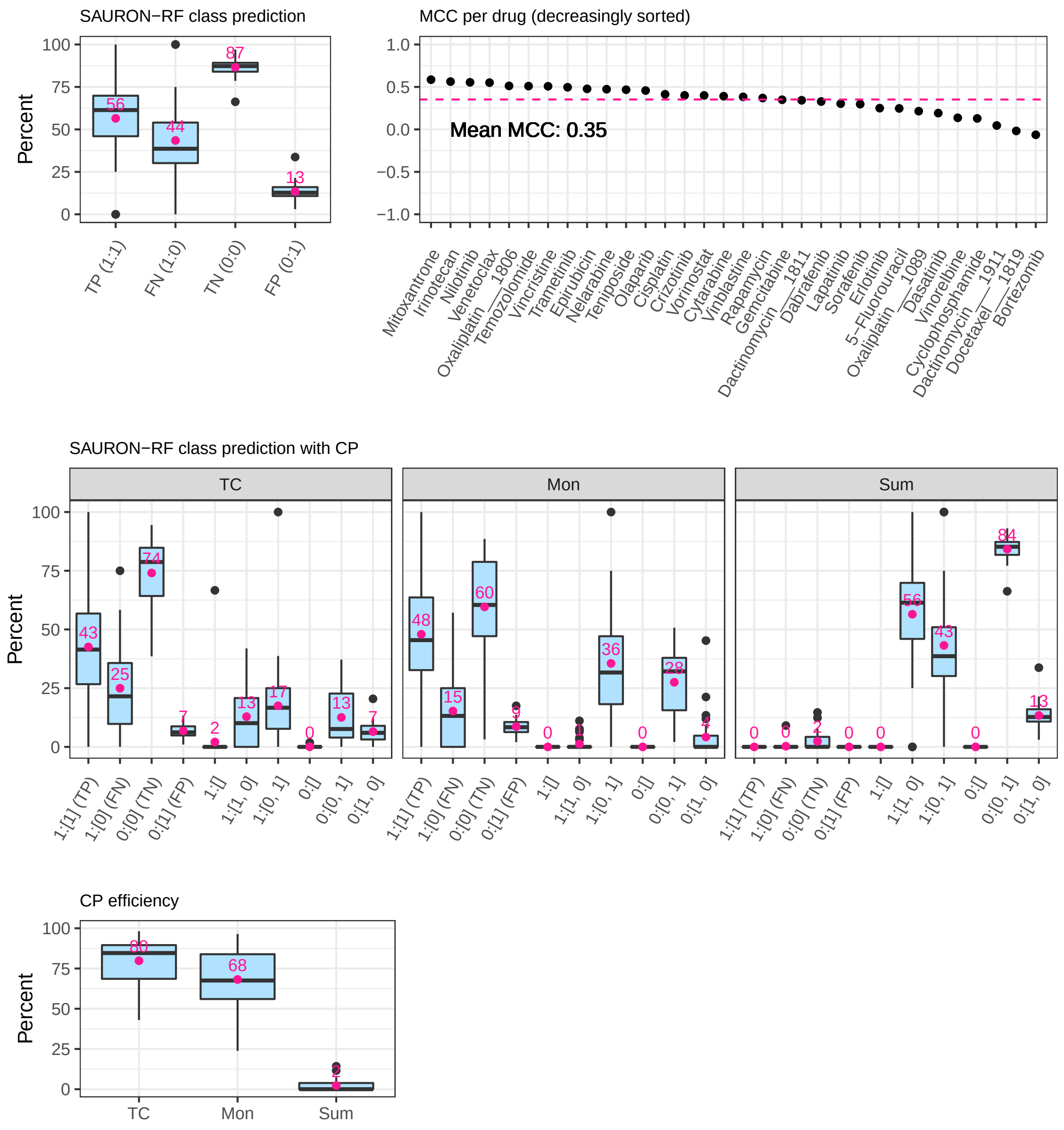


Figure 8.6: Classification test set performance GDSC2. The upper row of this figure depicts the classification performance of SAURON-RF across the different drugs from GDSC2. The middle row shows the effects of CP on the performance in terms of true positive/negative predictions. In the lower row, the CP efficiency is presented.

To achieve certainty, we employed our CP pipeline with a fixed allowed error rate of $\alpha = 10\%$. We notice that the certainty guarantee for classification and regression is indeed fulfilled for each of the three investigated classification scores and the

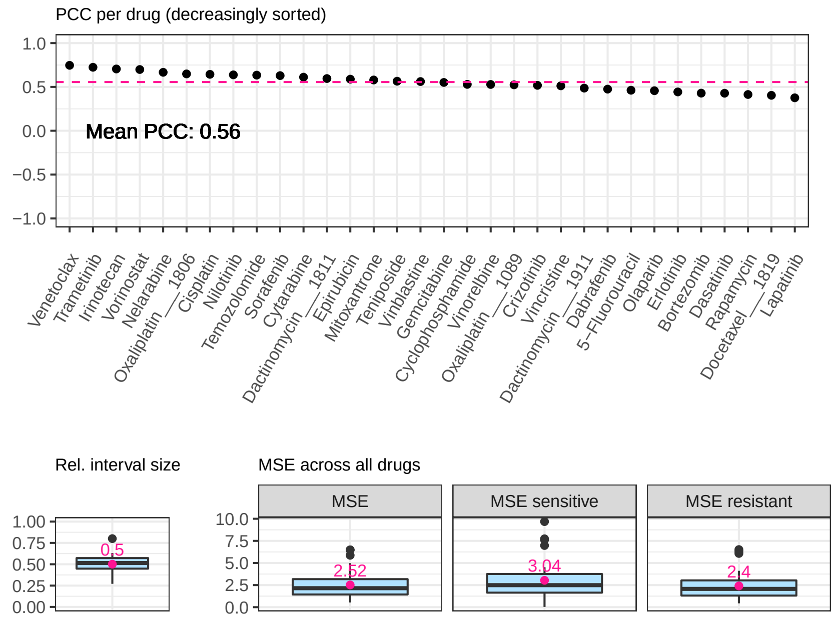


Figure 8.7: Regression test set performance GDSC2. The upper row of this figure depicts the Pearson correlation coefficient between the actual continuous response values and the predicted continuous response values for all drugs. The lower row shows the mean-squared error (MSE) and the interval width of the CP Quantile regression score relative to the spanned training ranges of the drugs.

regression score, i.e., our sets (classification) and our intervals (regression) contain the actual response with a probability of almost exactly $1 - \alpha = 90\%$ on average across all drugs (see Appendix Figure E.41). Next, we analyzed whether this also holds for each class to investigate the effect of class imbalance on the validity (see Appendix Figure E.41). Indeed, we fulfil the marginal coverage property for the majority class (resistant cell lines) for all scores. For the sensitive cell lines (minority class), the Summation score delivers valid sets in all cases, while the True-Class score coverage fluctuates with a mean of approximately 73%. The Mondrian score, which is supposed to fulfil the coverage property for each actual class by definition, exhibits significantly fewer fluctuations than the True-Class score and reaches a coverage of 85% across all drugs. For the Quantile regression score, the coverage for the sensitive cell lines is 86%. Since the adherence to the CP certainty guarantee depends on the number of available data points [197], the sensitive cell line scarcity can cause these fluctuations.

In our current application scenario, a valid prediction set can either stem from a

single class prediction or the set with all classes. CP efficiency is typically employed to quantify the number of single-class predictions among all predictions. It is defined as the number of single-class predictions divided by the total number of samples. In Figure 8.6, we depict the per-drug CP efficiency for the classification scores. We note that the True-Class score with an average CP efficiency of 80% clearly outperforms the Mondrian and Summation scores. The low CP efficiency of the Summation score then directly explains its high coverage: the Summation score almost exclusively predicts two-class sets as output (low efficiency), which by definition must contain the actual class in a binary classification (high coverage). For regression, the CP efficiency is given by the width of the interval. Consequently, it is highly desirable that these intervals are narrow. In Figure 8.7, we can, however, see that on average, across all drugs, the intervals are relatively large (approximately 50% relative to the spanned training range), which indicates that the trained models need to be refined in that respect. We discuss improvement strategies in the Discussion section.
With CP for classification, we pursue the goal of retaining the true positive and true negative predictions while minimizing the errors, i.e., false positive and false negative predictions. With the fixed $\alpha = 10\%$, the false positive (FP) errors were, on average, reduced from 13% to 9% and the false negative (FN) errors from 44% to 15% for the Mondrian score (cf. Figure 8.6). However, the true positive (TP) and true negative (TN) predictions also decrease: from 56% to 48% for the TP and 87% to 60% for the TN. In general, the True-class score also effectively removes FN (from 44% to 25%) and FP (from 13% to 7%). Again, the true predictions are also reduced: from 56% to 43% for the TP and from 87% to 74% for the TN. In contrast, we note that the Summation score does not only almost completely remove the false predictions but also the true predictions, which is in accordance with our previous observations for efficiency. Thus, the True-class score and the Mondrian score clearly outperform the Summation score, while the Mondrian score seems to perform better for the TP and FN values and the True-class score for the TN and FP values.
Next, we applied SAURON-RF and the CP pipeline to the newly derived CMax viability data set. We find that the CMax viabilities could be predicted with similar sensitivity (64%), specificity (76%), and MCC (0.35) compared to the IC50 data. We again ascertain that CP with a fixed error rate of $\alpha = 10\%$ delivers the desired 90% certainty guarantee on average (cf. Appendix Figure E.42). Indeed, it approximately holds for all three classification scores and the regression score on average across all

drugs. For the CMax viabilities, class imbalance also represents an issue. Contrary to the IC50 data, for some drugs, the sensitive cell lines constitute the minority class, and for others, the resistant cell lines do. Still, we discover the same overall trends for the validity of the scores of the minority and majority classes (see Appendix Figures E.43 and E.44). Regarding CP efficiency and the reduction in FP and FN predictions, we could also identify similar tendencies compared to our IC50 analyses (see Appendix Figures E.45 and E.46). Notably, with an average relative interval size of 0.62, the predicted regression intervals are larger for the CMax analyses than for the IC50 analyses. Overall, the CMax viability could be predicted with similar performance as the established IC50 value.
In the previous paragraphs, we described the results for a division of the CMax viability and IC50 values into two classes. However, a more fine-grained division into, e.g. three classes (sensitive, ambiguous, resistant) may more accurately reflect the biological variance and uncertainty of the experimental drug response values and may thus be even more accurately learned and predicted by models. We first applied SAURON-RF without CP to the ternary CMax drug data sets. The results (see Appendix Figures E.47 - E.49) reflect all general tendencies we reported for the binary partition. Here, it is particularly noteworthy that confusions between the sensitive and resistant classes seem rather rare (9% on average for the sensitive samples and 6% on average for resistant samples), which aligns with the goal of improving certainty. Nevertheless, both classes displayed a high confusion with the ambiguous class (37% on average for the sensitive class and 39% on average for the resistant class), and the average PCC (0.49) and MCC (0.3) are slightly lower than those for the binary partition. We also evaluated the validity and efficiency of the CP pipeline. Briefly, the efficiency was considerably lower than for the two-class partition. Thus, we decided to focus on the binary partition for the cell line-centric analysis.

8.4.2 Cell line-centric analysis - drug prioritization

Due to the shared test set and the across-drug-comparability of the CMax viability, we can now assess the performance from a cell line-centric perspective, i.e., for each cell line, we can identify effective drugs (classification) and then prioritize them (regression). We call a drug effective if its CP class set prediction for a particular cell line consists solely of the single class indicating sensitivity (1). We then subsequently rank all drugs that fulfil this property for a particular cell line using the upper limit of the CP interval. Figures 8.8 and 8.9 exemplary depict the results for such a prioritization task for two particular cell lines (see Appendix Figures E.50 - E.53 for further cell lines). Notably, for both examples, the SAURON-RF point predictions not only efficiently distinguish between effective and non-effective drugs (MCC 0.66 and 0.6) but also sort them exceptionally well (PCC 0.9 and 0.89). For the first example, there still exist FN predictions but no FP predictions, which we would like to remove. Both the True-class and the Mondrian score expectedly accomplish the task of removing the FN predictions well at the cost of a few TP predictions. Also, in accordance with our previous drug-centric analyses, the Summation score removes all single-class predictions. For the second example, we have several FP and one FN prediction. Removing FP predictions is particularly important since this equals the avoidance of ineffective drug treatments. The True-class score outperforms the Mondrian and the Summation score in removing these false predictions. In particular, the True-class score eliminated 3 out of 4 FP predictions, while the Mondrian score removed only 2 out of 4, suggesting that the True-class score might be more suitable to avoid ineffective drug treatments. In total, the True-class score seems to slightly outperform the Mondrian score, while both are superior to the Summation score. The CP regression intervals are again spanning a wide range of values. Nevertheless, they are ascending alongside the actual values, which indicates that they can be employed for sorting the drugs. A Spearman correlation coefficient (SCC) of 0.87 (0.8) between the upper limit of the CP interval and the true values confirms this impression. In the lower rows of Figures 8.8 and 8.9, we also depicted the potential prioritizations obtained by sorting the sets of effective drugs after CP deployment. For the Summation score, no prioritization is possible since no drug was predicted to be effective after CP. However, the rankings introduced by the CP upper limit of the interval are reasonably similar to the actual rankings for the restricted sets of drugs from the Mondrian (SCC 0.6 and 0.59) and True-class (SCC 0.62 and 0.55) scores.

Finally, we analyzed whether these observations hold for all test cell lines (cf. Figure 8.10). With an average MCC of 0.53, sensitivity of 71%, specificity of 81%, and PCC of 0.81, SAURON-RF performs well in both the classification and the regression task. The Mondrian and the True-class score effectively remove the false

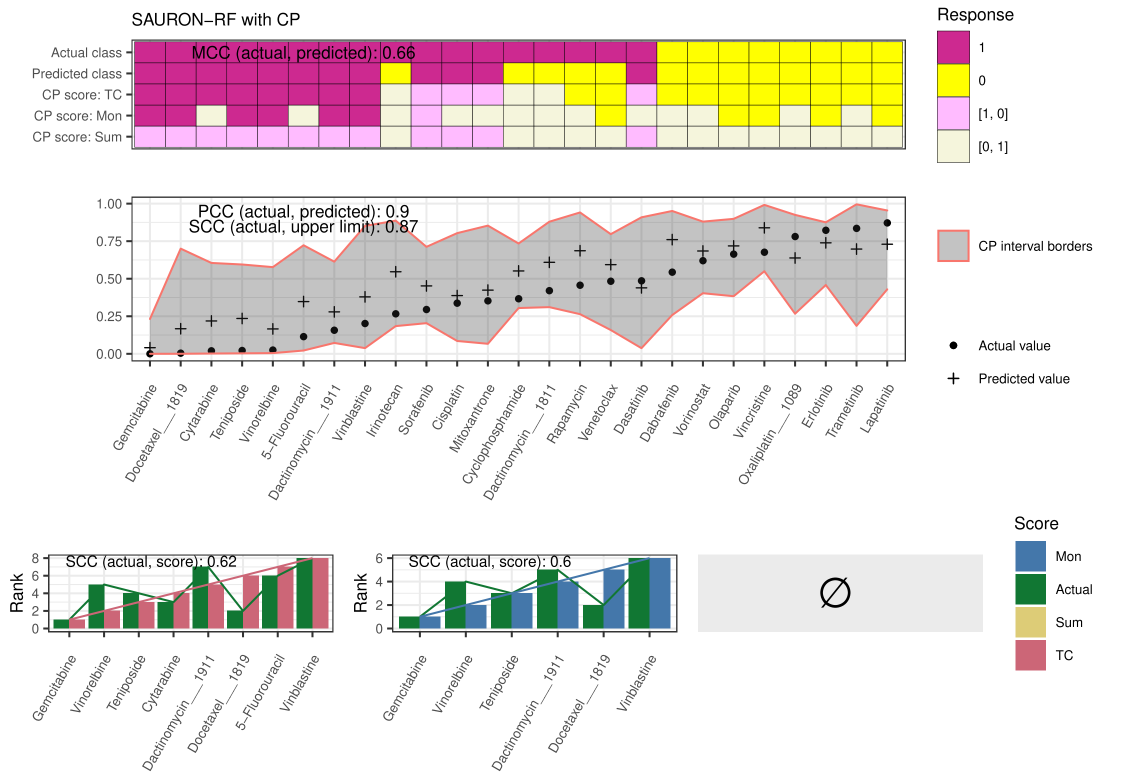


Figure 8.8: Prioritization example GDSC2. This figure exemplifies the performance of our prioritization pipeline (cf. Figure 8.3) when applied to one particular cell line (COSMIC ID 1240154) from the test set of the GDSC2 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

predictions: 48 % less FN for Mondrian compared to 53% less FN for True-class as well as 42% less FP for Mondrian and 52% less FP for True-class. However, both scores reduce not only the false predictions but also true predictions: 45% less TP for Mondrian compared to 26% less for True-class as well as 39% less TN for Mondrian compared to 17% for True-class. Indeed, the True-class score not only reduces the false predictions to a greater extent but also preserves more correct predictions, i.e., it clearly outperforms the Mondrian (and the Summation) score in this analysis. For the regression part of the pipeline, we note that the average SCC between the SAURON-RF predictions and the actual values (0.82) is slightly higher than the average SCC between the upper limit of the CP interval and the

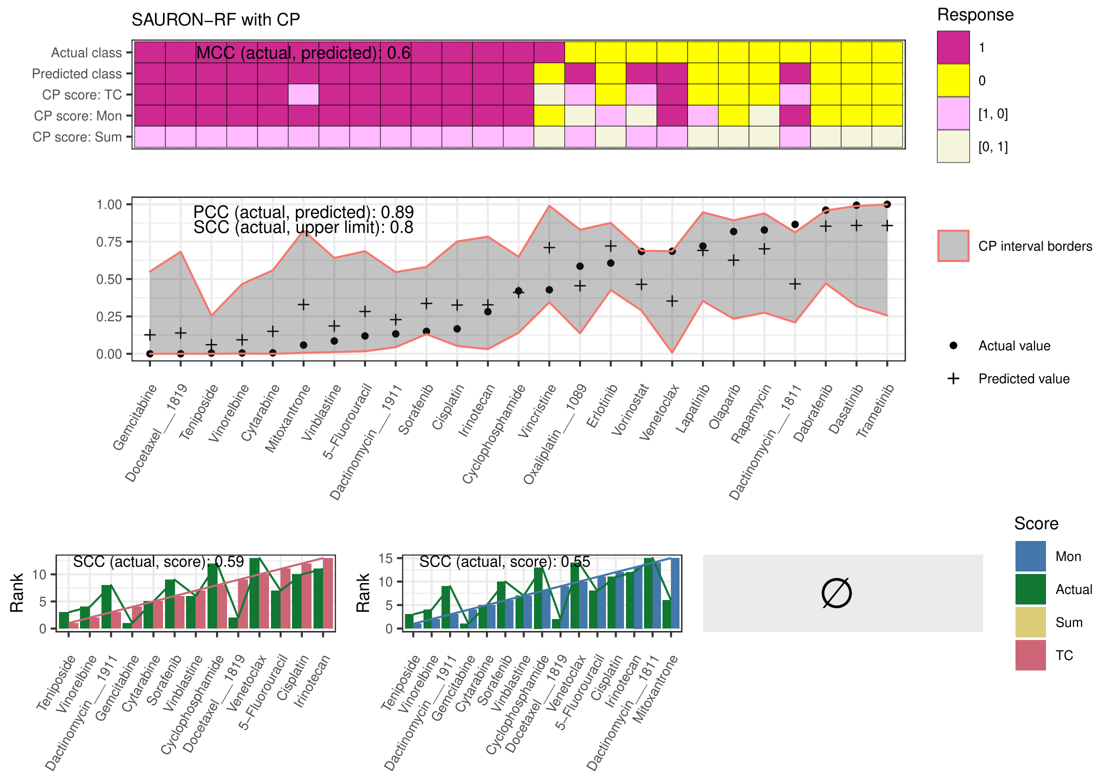


Figure 8.9: Second prioritization example GDSC2. This figure exemplifies the performance of our prioritization pipeline (cf. Figure 8.3) when applied to one particular cell line (COSMIC ID 688031) from the test set of the GDSC2 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

actual values (0.75). The goal of the prioritization task is to obtain a complete list of potentially effective drugs sorted by their efficiency. We already noticed that the True-Class score retains more TP predictions than the Mondrian score, i.e., it yields more complete lists of effective drugs. Furthermore, the predicted effective drug list from the True-class score has a higher median precision (92%) than the list predicted by using the Mondrian score (83%). Both are superior to SAURON-RF only (76%). Despite the fact that the TP predictions are also reduced by performing CP, the actual most efficient drug belongs to this list 75% of the time for the True-class score and 56% for the Mondrian score. Moreover, the first drug in our predicted effective drug list still has a median rank of three in the original drug list for both the True-class and the Mondrian score and is a TP prediction in 85% (TC) and 79% (Mon) of cases. The CMax viability difference between this drug and the actual first drug is below 0.1 for 62% of cell lines for the True-Class score and 56% of cell lines for the Mondrian score. In relation to the CMax viability range ($[0, 1]$), this value indicates reasonable proximity of the actual first drug and the drug that we predict to occupy rank one. Overall, we find that the True-Class score is most convincing concerning correctness and completeness.

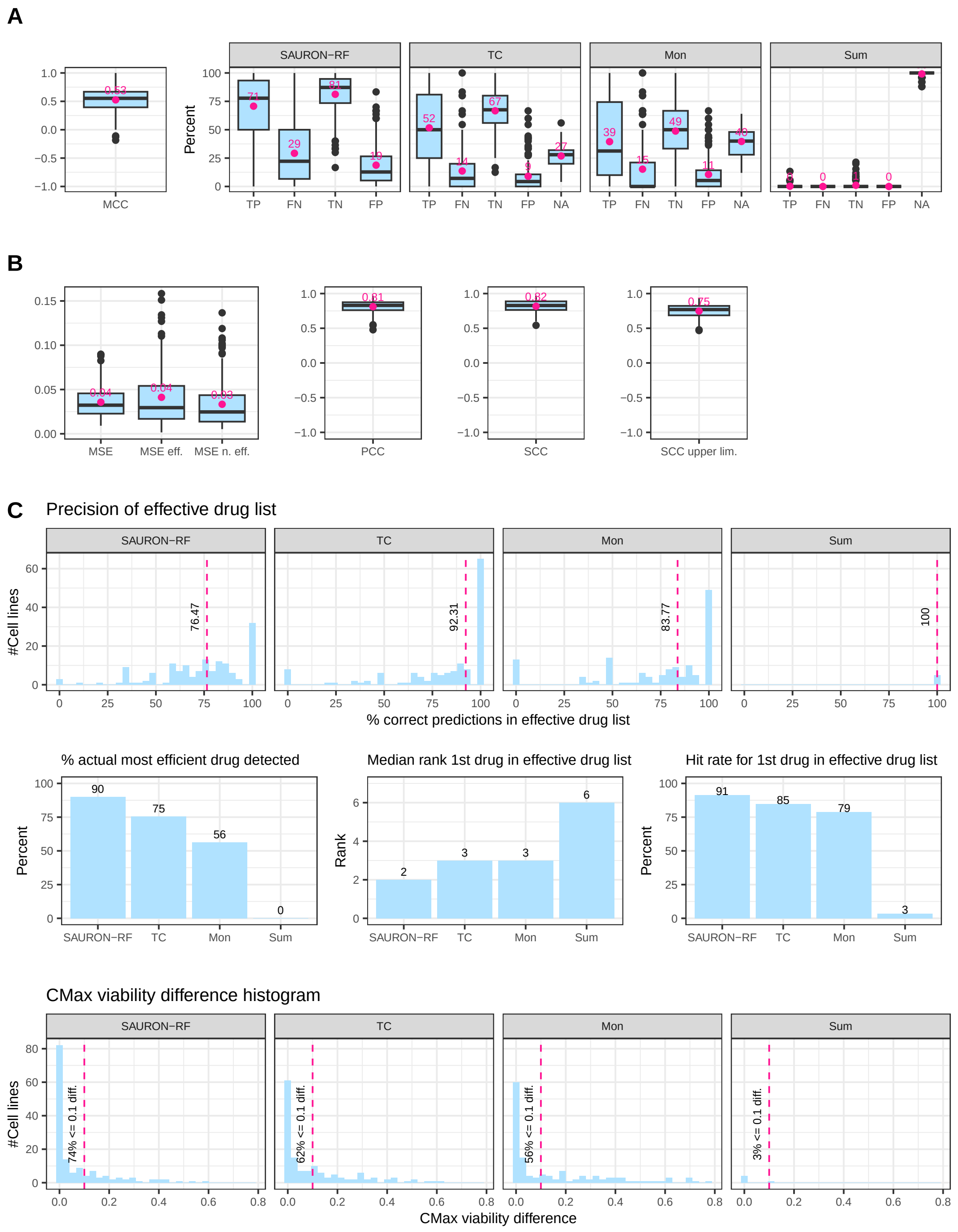


Figure 8.10: Prioritization results across all test cell lines of GDSC2. In A, we show the classification performance of SAURON-RF with and without CP. B depicts the regression performance in terms of MSE, PCC and SCC. Here, the MSE is given for the effective drugs, the ineffective drugs, and all drugs. We provide the SCC using SAURON-RF only (SCC) and SAURON-RF with the upper limit of the CP interval (SCC upper lim.). In C, the upper row depicts the precision of SAURON-RF only (SAURON-RF) and SAURON-RF with CP (TC + upper limit, Mon + upper limit, Sum + upper limit). In the middle row, we show the percentage of cell lines for which the most efficient drug was detected, the median rank of the first drug in our predicted effective drug list and the percentage of cell lines for which this prediction was a TP. The CMax viability difference between our first drug and the actual first drug is shown in the lower row.

8.5 Discussion

With the reliable SAURON-RF drug sensitivity prediction and prioritization methodology, we aimed to address two crucial challenges in the area of anti-cancer drug treatment optimization with ML systems: We were interested in (1) reliably predicting anti-cancer drug responses (2) and prioritizing drugs for a given cancer sample based on these reliable predictions.
To tackle the first challenge, we implemented a conformal prediction pipeline providing user-specified certainty levels. Our pipeline can handle not only regression or classification methods but also joint classification and regression methods. Our results demonstrate that CP can substantially improve predictions. In particular, CP does not only provide guarantees for predictions, but it successfully diminishes false predictions, i.e., FP and FN, while retaining TP and TN.
To address the second challenge, we developed a novel drug sensitivity measure called CMax viability that is comparable across drugs. Since the CMax viability is based on clinically relevant drug concentrations, it may also help to translate findings into clinical application. By deploying the CP pipeline with our joint regression and classification method SAURON-RF and the CMax viability, we could finally fulfil the prioritization task: We could first use the classification part of our model combined with CP to successfully identify drugs that are very likely effective. In particular, by applying CP, we could eliminate 52% of the remaining 19% ineffective drugs falsely predicted to be effective by SAURON-RF. In total, we thus achieved a median overall 92% precision of our prioritized drug lists, which 75% of the time also contained the actual most efficient drug. Finally, we could also predict the continuous drug sensitivity and, through the extension of SAURON-RF with quantile regression, build intervals that contain the correct response with a high probability. Our results indicate, that the first drug of our predicted list has a similar CMax viability value as the actual most efficient drug. Thus, the presented CP drug sensitivity prediction and prioritization pipeline can serve as a valuable asset in medical decision support systems by delivering a sorted list of recommended drugs serving as reinforcement of already established therapies or as a suggestion of alternative treatment options.
Nevertheless, we recognize several starting points for improvement. We currently train our models on cell line-based monotherapy responses because of the relatively high abundance of the corresponding data, which is beneficial for training ML mod-

els. However, since monotherapy can promote drug resistance [262], integrating data from drug combination screens would be highly desirable to increase the value of our tool for actual medical decision-making. Similarly, incorporating data from more complex model systems such as patient-derived xenografts or organoids may be advantageous because they are assumed to more accurately represent tumour characteristics [84]. Apart from that, we were focusing solely on the gene expression data as input features. While gene expression is assumed to be the most informative data type [17], the interpretability of models can benefit from the integration of additional data types such as mutation and copy number variation data, and, in particular, a priori knowledge, e.g., in the form of known biomarkers [21], biological pathways and gene interaction networks [19, 162], or drug-based features [152, 153]. If those features complement the information from the gene expression data, the performance in terms of certainty might also be increased. Besides, we opted for a particular type of conformal prediction in this work and implemented three different classification scores and one regression score. Since we noted that the regression intervals are rather wide and the prioritization of the effective drug list might be negatively affected by this, it might be beneficial to investigate different regression scores. In addition, there exists a plethora of CP-based techniques [263], some of which may even further improve classification and regression results.

9 Summary, discussion, and conclusion

Don't adventures ever have an end? I suppose not. Someone else always has to carry on the story.

(Bilbo Baggins, LotR)

The development of (trustworthy) machine learning (ML) methods is of greater importance than ever before. Already today, there is hardly any research field or company that does not rely on some ML-aided analysis of data. ML-based systems are anticipated to continue to tremendously augment and change our lives, including especially sensitive areas such as human healthcare.
In this thesis, we pursued the goal of improving ML-based tools that are designed to be incorporated into medical decision support systems for anti-cancer drug treatment:

- To this end, we started with describing the fundamental concepts of ML in Chapter 4, including a description of the four major ML branches: supervised, unsupervised, semi-supervised, and reinforcement learning. Moreover, we discussed properties that render ML methods trustworthy, particularly reliability and interpretability. Throughout this chapter, we repeatedly contextualized the existing body of drug sensitivity prediction approaches. Almost all approaches fall within the supervised ML realm, which is also the basis of the ML methods developed and presented in this thesis. While reliability has hardly been addressed so far for drug sensitivity prediction, the concept of interpretability has been used rather intuitively using different connotations without a clear definition. Therefore, we unified its prevalent connotations and proposed an easily extensible taxonomy of interpretability, which may serve as reference, facilitating future research.

- In Chapter 5, we devised a classifier focusing on model interpretability: MERIDA delivers Boolean logic-based rules as output. Unlike all of its competitors, MERIDA explicitly includes a priori pharmacogenomic knowledge in terms of known sensitivity- and resistance biomarkers, rendering the model even more interpretable. By crafting features based on pharmacogenomic knowledge, we not only increase the interpretability of the models but simultaneously reduce the dimensionality of the feature space. Apart from that, MERIDA addresses the class imbalance issue present in the GDSC drug data set.

- In Chapter 6, we show that imbalance is not only an issue for classification but also for regression, i.e., the high specificity of most anti-cancer drugs induces a skewed distribution of drug response values in favour of the more drug-resistant cell lines, negatively affecting the regression performance for the sensitive cell lines. We exemplified that all commonly applied ML regression algorithms (neural networks, boosting trees, elastic net, random forest) suffer from this imbalance. To diminish the negative influence of the regression imbalance on our predictions, we designed SAURON-RF. It is a joint regression and classification method based on canonical regression random forests augmented with sample-specific and tree-specific weights derived from the class distribution. It outperforms mere classification, mere regression, and sequential execution of classification followed by regression.

- While MERIDA relies on literature-driven feature selection and creation, SAURON-RF employs a heuristic based on the maximum-relevance-minimum-redundancy principle, i.e., a selection of features statistically associated with the response. In Chapter 7, we presented the results of a comprehensive benchmarking study on DR techniques and ML methods. More specifically, we trained four different supervised ML methods (elastic net, random forest, boosting trees, neural networks) combined with nine different DR approaches (random, literature-based, variance, correlation, enrichment, minimum-redundancy-maximum-relevance, principal component analysis, pathway activity, autoencoder) resulting in more than 16,000,000 investigated models. Overall, this analysis revealed that relatively simple ML methods outperform more complex ones: the elastic net outperformed all other methods, closely followed by random forests. Neural networks were not competitive, neither as an ML method nor as a DR technique.

- In Chapter 4, we already pointed out that reliability, e.g., in the form of certainty guarantees, has hardly been integrated into current approaches for drug sensitivity prediction. In Chapter 8, we addressed this demand by designing and implementing a framework for reliable drug response classification and regression based on conformal prediction (CP). This framework is, in principle, applicable to any supervised ML method. To render SAURON-RF eligible, we extended it with a quantile regression algorithm adapted from Meinshausen et al. [32]. We demonstrated that CP not only delivers the desired certainty guarantees but also successfully diminishes false predictions while retaining correct ones - ultimately improving model performance. The CP certainty guarantees enable SAURON-RF to abstain from casting a prediction for a new instance if it is uncertain, potentially facilitating integration into decision-support tools.

- Throughout the entire work up to Chapter 8, we employed the IC50 value as a drug response measure. It is comparable across cell lines but not across drugs. Given a particular sample, it can consequently not be employed to compile a list of recommendable drugs sorted by efficiency, i.e., a straightforward drug prioritization for one sample is prevented. As a remedy, we propose a novel measure with across-drug comparability, the CMax viability. Thereby, we can finally leverage the full potential of SAURON-RF: we can first identify effective drugs using classification and then rank these by their efficiency with regression. Thus, combined with the CP framework our tool can assist decision-making by providing a list of prioritized drugs that can reinforce medical advice or suggest alternative treatment options at a user-specified certainty guarantee.

In Chapters 5 to 8, we already discussed specific weaknesses of our novel approaches in dedicated discussion sections. In this chapter, we want to comprehensively review our results in their entirety, focusing on recurring issues and their putative remedies.

9.1 Input data

We decided to train our ML methods on the Genomics of Drug Sensitivity in Cancer (GDSC) database, which is a publicly available pharmacogenomic cancer cell line panel encompassing approximately 1000 cell lines screened with several hundreds of compounds. The database also provides multi-omics profiles of the cell lines, including (epi)genomic and transcriptomic data. The joint analysis of the multi-omics profiles and the drug response data enables the identification of associations between tumour biology and therapeutic response. Compared to other model systems (cf. Chapter 3), cancer cell lines have enjoyed immense popularity for anti-cancer drug screening, resulting in the most comprehensive, publicly available resource to study anti-cancer drug response. The comparatively high number of different samples (cell lines) played a critical role in our choice since ML models - like any statistical evaluation - benefit from a high abundance of the investigated data.

However, model systems reflect reality only to some degree. For example, cancer cell lines are not genetically identical to the tumours of origin, and they can neither represent the 3D structure nor the cell diversity of a tumour and its environment. Consequently, our results, including our trained models, cannot be translated directly into the clinic. A relatively straightforward option to improve our models in that respect is considering data from more realistic model systems, such as xenograft models or 3D cell cultures. Since the available amount of data is comparatively low (cf. Chapter 3), we might aim at developing methods for combining data from heterogeneous sources instead of training models only on one model system. To this end, transfer learning [264] or meta-learning [265] techniques may become leveraged. Possibly, such techniques also represent an opportunity to incorporate existing tumour and clinical data.

Another noteworthy limitation of our data choice is that the GDSC database solely contains monotherapy responses. Yet, monotherapy promotes drug resistance [262], which is why drug combinations are administered instead [10, 266, 267, 268]. Since the number of putative combinations - even when resorting to the investigation of pairwise combinations - is enormous, the available amount of experimental drug synergy data is significantly more limited than monotherapy data (cf. Chapter 3). Thus, training supervised ML models will likely prove even harder for synergy data. We could employ standard semi-supervised learning as a potential remedy to this issue. Since we have the sensitivity data at our disposal, a more appropriate solution might involve the combination of these two data types in a semi-supervised fashion. For example, we might perform multi-task learning with sensitivity and synergy data as response values since we expect that the relatedness of the tasks can be exploited. A more explicit utilization of the commonalities between the tasks

might involve the derivation of constraints from the sensitivity data for the missing synergy data.
A further point for discussion is our focus on cell line-derived input features, neglecting the possibility of incorporating chemical information of drugs. Notably, there already exist works exploiting this information (e.g., [152] and [153]). While we believe that incorporating drug-based features can be beneficial for modelling, we are unaware of a comprehensive benchmarking between methods with and without such knowledge.
In the following years, technological advances in experimental techniques will undoubtedly change and improve the data situation, opening up new possibilities for data analysis.

9.2 Machine learning and algorithmic choices

Throughout this thesis, we investigated various supervised ML approaches to predict the sensitivity of cancer cell lines to anti-cancer compounds. While we started our analyses with an inherently interpretable Boolean-logic-based optimization approach, we also studied classical ML techniques, i.e., elastic net, boosting trees, random forests, and neural networks. One prominent of our findings is that neural networks, which were the most complex tested approach, are not superior to simpler methods (cf. Chapter 6 and Chapter 7). On the contrary, in our benchmarking, they were the least-performing ML method with respect to runtime, statistical performance, and inherent interpretability. Due to the low or even absent model uncertainty of neural networks, we would initially expect that they are the most suitable approach to model complex biological processes. However, their low model uncertainty is coupled with a high estimation uncertainty, i.e., extensive data sources are required to estimate their parameters. Because of the limited data availability, it thus seems logical that they are not the most convincing approach. Our results for MERIDA and elastic nets support this interpretation. Both methods should incur a high model uncertainty (high bias) and were competitive with or superior to methods of lower model uncertainty. Nevertheless, there exist various works on neural networks for drug sensitivity prediction claiming their superiority (e.g., CDRscan by Chang et al. [159], DeepDR by Chiu et al. [161], and the model by Deng et al. [163]). Thus, the question arises how we can harmonize our results with theirs. Clearly, we focused on certain aspects during our benchmarking. For example, we only used gene expression data. The inclusion of additional omics types may, however, reveal that neural networks could more accurately capture biological processes between omics types. Generally, we trained rather shallow neural networks, and one can extend this criticism to the benchmarking study as a whole: we trained relatively basic approaches. That being said, we would also like to note that we refrained from training more complex neural networks because of their extremely high runtime. Moreover, the mentioned works also have their respective shortcomings. Chang et al. and Deng et al. did not perform dimensionality reduction in front of their basic competitor methods. Moreover, it is unclear whether Chang et al. performed hyperparameter tuning for the basic competitors or the neural network. Chiu et al. employed a dimensionality reduction method for the basic competitors, however, they used the inferior dimensionality reduction method from their neural network comparison and did not optimize the number of input features. Interestingly, in a recent study, Li et al. benchmarked some neural network methods including the one by Deng et al. [269]. They found that RFs mostly perform similarly or even better. A recent study on ML-based science reveals another putative reason for reports on extremely well-performing complex ML methods [254]. In this study, Kapoor and Narayanan outline that data leakage, which roughly corresponds to inappropriate handling of data before, during, or after the model training process, causes overoptimistic claims about the performance of ML methods. To clarify the question of which method is most suitable for the currently available drug response data, a large-scale comparison study between existing advanced (neural network) methods and basic competitors would be needed.

All our novel approaches are based on discriminative supervised ML methods since they are straightforward modelling techniques for drug response prediction. In Chapter 4, we, however, outlined the diversity of the ML landscape and indicated which areas are hardly researched, including generative supervised ML methods, semi-supervised learning and reinforcement learning. Accordingly, it would be of interest to advance the methodological development of these areas and benchmark already existing methods from these areas against discriminative supervised models.

Regardless of which specific ML method proves to be particularly useful for drug sensitivity prediction, its development will encompass strategies to achieve trustworthiness, e.g., in terms of interpretability, reliability, reproducibility, security, safety, privacy, and fairness. In our work, we solely focused on reliability and interpretability. To implement reliability, our conformal prediction framework, and conformal prediction more generally, is a particularly attractive starting point, as conformal prediction is not only flexible and versatile but also easy to use. In our work, we tested only some conformal prediction techniques. Yet, there exists a plethora of different ones [263], some of which may serve our goals better. Additionally, we can and should pursue other strategies on the path to reliability, e.g., performing out-of-distriubtion estimation [270] or using probabilistic graphical models [271]. Regarding interpretability, we were mainly concerned with generating inherently interpretable (transparent) models. Interestingly, our results allow us to conclude that relatively simple, transparent models suffice for drug sensitivity prediction given the current data situation. The apparent biological complexity suggests that with increasing data availability, they might not be able to compete with more complex, less inherently interpretable models anymore. Consequently, efficient methods for generating post-hoc interpretability (explainability) are needed. Here, inherently interpretable models may be re-used to generate partial explanations for observed phenomena in less inherently interpretable models. Note that, even for methods traditionally regarded as more inherently interpretable as deep neural networks, e.g., random forests, the derivation of post-hoc explanations can help to better understand the underlying biology.

9.3 Perspectives

In the previous paragraphs, we already outlined solutions to specific problems arising for drug sensitivity prediction in cancer. However, similar problems occur regularly and all over (ML-based) science. Depending on the scientific discipline, the importance of the problems may, of course, vary. From our previous considerations, we derive the following main long-term challenges in the development of methods for drug sensitivity prediction, but also of methods for medical decision support more generally:

- How can we combine complex (heterogeneous) data sources most efficiently? Biological systems are extremely complex, and data gathered about them tends to add complexity, e.g., because of biases introduced by the experimental technique or a missing temporal resolution of the measurements. Thus, the available data must be combined such that conclusions can be drawn despite or thanks to the complexity.

- How can we develop benchmarking standards? Our world seems to change faster every day, and information is - though also more accessible - more obscure and scattered than ever before. In addition, negative results are often not reported because they are not valued sufficiently. As a consequence, the scientific literature represents an even more distorted image of science in a field than is already the case due to subjective perceptions of scientists. These circumstances represent an obstacle to comprehensive benchmarking and should be considered when designing benchmarking standards.
- How can we implement trustworthy (ML) systems? From a pure computational perspective, we can invent and implement techniques to ensure whatever we as humans consider to be an aspect of trustworthiness. However, one factor we should not overlook is the human itself as part of the interaction with the (ML) system. Not only do we need human-computer interaction studies for the design of the user interfaces, but also a wide public awareness and understanding of such systems.

Mastering such challenges requires a vibrant and diverse scientific community and a political and societal environment enabling the needed transformations. While personalized medicine has been a dream for centuries, we believe that the tasks ahead can be accomplished in a relatively short time and (ML-based) personalized medicine is a forthcoming reality now.

Bibliography

[1] Edward Abrahams and Mike Silver. The history of personalized medicine. *Integrative neuroscience and personalized medicine*, pages 3–16, 2010.

[2] Cyrille Delpierre and Thomas Lefèvre. Precision and personalized medicine: What their current definition says and silences about the model of health they promote. implication for the development of personalized health. *Frontiers in Sociology*, 8:1112159, 2023.

[3] Jaeyun Sung, Yuliang Wang, Sriram Chandrasekaran, Daniela M Witten, and Nathan D Price. Molecular signatures from omics data: from chaos to consensus. *Biotechnology journal*, 7(8):946–957, 2012.

[4] Edwin Baldwin, Jiali Han, Wenting Luo, Jin Zhou, Lingling An, Jian Liu, Hao Helen Zhang, and Haiquan Li. On fusion methods for knowledge discovery from multi-omics datasets. *Computational and structural biotechnology journal*, 18:509–517, 2020.

[5] World Health Organization (WHO). Cancer. `https://www.who.int/en/news-room/fact-sheets/detail/cancer`, 2022. [Online; accessed 21-July-2023].

[6] Hyuna Sung, Jacques Ferlay, Rebecca L Siegel, Mathieu Laversanne, Isabelle Soerjomataram, Ahmedin Jemal, and Freddie Bray. Global cancer statistics 2020: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: a cancer journal for clinicians*, 71(3):209–249, 2021.

[7] German Federal Statistical Office (Destatis). Todesursachen nach krankheitsarten 2022 in %. `https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Gesundheit/Todesursachen/_inhalt.html#sprg229156`, 2024. [Online; accessed 8-January-2024].

[8] Jacob K Pedersen, Gerda Engholm, Axel Skytthe, Kaare Christensen, and Academy of Geriatric Cancer Research (AgeCare). Cancer and aging: epidemiology and methodological challenges. *Acta Oncologica*, 55(sup1):7–12, 2016.

[9] Genevieve Housman, Shannon Byler, Sarah Heerboth, Karolina Lapinska, Mckenna Longacre, Nicole Snyder, and Sibaji Sarkar. Drug resistance in cancer: an overview. *Cancers*, 6(3):1769–1792, 2014.

[10] Neil Vasan, José Baselga, and David M Hyman. A view on drug resistance in cancer. *Nature*, 575(7782):299–309, 2019.

[11] Raihan Rafique, SM Riazul Islam, and Julhash U Kazi. Machine learning in the prediction of cancer therapy. *Computational and Structural Biotechnology Journal*, 19:4003–4017, 2021. doi: https://doi.org/10.1016/j.csbj.2021.07.003.

[12] George Adam, Ladislav Rampášek, Zhaleh Safikhani, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. Machine learning approaches to drug response prediction: challenges and recent progress. *NPJ precision oncology*, 4(1):1–10, 2020. doi: https://doi.org/10.1038/s41698-020-0122-1.

[13] Robert H Shoemaker. The nci60 human tumour cell line anticancer drug screen. *Nature Reviews Cancer*, 6(10):813–823, 2006.

[14] Mathew J Garnett, Elena J Edelman, Sonja J Heidorn, Chris D Greenman, Anahita Dastur, King Wai Lau, Patricia Greninger, I Richard Thompson, Xi Luo, Jorge Soares, et al. Systematic identification of genomic markers of drug sensitivity in cancer cells. *Nature*, 483(7391):570–575, 2012. doi: https://doi.org/10.1038/nature11005.

[15] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, Joseph Lehár, Gregory V Kryukov, Dmitriy Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391): 603–607, 2012. doi: https://doi.org/10.1038/nature11003.

[16] Brinton Seashore-Ludlow, Matthew G Rees, Jaime H Cheah, Murat Cokol, Edmund V Price, Matthew E Coletti, Victor Jones, Nicole E Bodycombe, Christian K Soule, Joshua Gould, et al. Harnessing connectivity in a large-scale small-molecule sensitivity dataset. *Cancer discovery*, 5(11):1210–1223, 2015. doi: https://doi.org/10.1158/2159-8290.CD-15-0235.

[17] James C Costello, Laura M Heiser, Elisabeth Georgii, Mehmet Gönen, Michael P Menden, Nicholas J Wang, Mukesh Bansal, Muhammad Ammad-Ud-Din, Petteri Hintsanen, Suleiman A Khan, et al. A community effort to assess and improve drug sensitivity prediction algorithms. *Nature biotechnology*, 32(12):1202–1212, 2014. doi: https://doi.org/10.1038/nbt.2877.

[18] Wanjuan Yang, Jorge Soares, Patricia Greninger, Elena J Edelman, Howard Lightfoot, Simon Forbes, Nidhi Bindal, Dave Beare, James A Smith, I Richard Thompson, et al. Genomics of drug sensitivity in cancer (gdsc): a resource for therapeutic biomarker discovery in cancer cells. *Nucleic acids research*, 41 (D1):D955–D961, 2012.

[19] Francesco Iorio, Theo A Knijnenburg, Daniel J Vis, Graham R Bignell, Michael P Menden, Michael Schubert, Nanne Aben, Emanuel Gonçalves, Syd Barthorpe, Howard Lightfoot, et al. A landscape of pharmacogenomic interactions in cancer. *Cell*, 166(3):740–754, 2016. doi: https://doi.org/10.1016/j.cell.2016.06.017.

[20] Cancer Cell Line Encyclopedia Consortium, Genomics of Drug Sensitivity in Cancer Consortium, et al. Pharmacogenomic agreement between two cancer cell line data sets. *Nature*, 528(7580):84–87, 2015.

[21] Kerstin Lenhof, Nico Gerstner, Tim Kehl, Lea Eckhart, Lara Schneider, and Hans-Peter Lenhof. Merida: a novel boolean logic-based integer linear program for personalized cancer therapy. *Bioinformatics*, 37(21):3881–3888, 2021. doi: https://doi.org/10.1093/bioinformatics/btab546.

[22] Abel Gonzalez-Perez, Christian Perez-Llamas, Jordi Deu-Pons, David Tamborero, Michael P Schroeder, Alba Jene-Sanz, Alberto Santos, and Nuria Lopez-Bigas. Intogen-mutations identifies cancer drivers across tumor types. *Nature methods*, 10(11):1081–1082, 2013.

[23] John G Tate, Sally Bamford, Harry C Jubb, Zbyslaw Sondka, David M Beare, Nidhi Bindal, Harry Boutselakis, Charlotte G Cole, Celestino Creatore, Elisabeth Dawson, et al. Cosmic: the catalogue of somatic mutations in cancer. *Nucleic acids research*, 47(D1):D941–D947, 2019.

[24] Malachi Griffith, Nicholas C Spies, Kilannin Krysiak, Joshua F McMichael, Adam C Coffman, Arpad M Danos, Benjamin J Ainscough, Cody A Ramirez, Damian T Rieke, Lynzey Kujan, et al. Civic is a community knowledgebase for expert crowdsourcing the clinical interpretation of variants in cancer. *Nature genetics*, 49(2):170–174, 2017.

[25] Debyani Chakravarty, Jianjiong Gao, Sarah Phillips, Ritika Kundra, Hongxin Zhang, Jiaojiao Wang, Julia E Rudolph, Rona Yaeger, Tara Soumerai, Moriah H Nissan, et al. Oncokb: a precision oncology knowledge base. *JCO precision oncology*, 1:1–16, 2017.

[26] David Tamborero, Carlota Rubio-Perez, Jordi Deu-Pons, Michael P Schroeder, Ana Vivancos, Ana Rovira, Ignasi Tusquets, Joan Albanell, Jordi Rodon, Josep Tabernero, et al. Cancer genome interpreter annotates the biological and clinical relevance of tumor alterations. *Genome medicine*, 10:1–8, 2018.

[27] Theo A Knijnenburg, Gunnar W Klau, Francesco Iorio, Mathew J Garnett, Ultan McDermott, Ilya Shmulevich, and Lodewyk FA Wessels. Logic models to predict continuous outputs based on binary inputs with an application to personalized cancer therapy. *Scientific reports*, 6(1):1–14, 2016. doi: https://doi.org/10.1038/srep36812.

[28] Rita P Ribeiro and Nuno Moniz. Imbalanced regression and extreme value prediction. *Machine Learning*, 109(9):1803–1835, 2020. doi: https://doi.org/10.1007/s10994-020-05900-9.

[29] Kerstin Lenhof, Lea Eckhart, Nico Gerstner, Tim Kehl, and Hans-Peter Lenhof. Simultaneous regression and classification for drug sensitivity prediction using an advanced random forest method. *Scientific Reports*, 12(1): 13458, 2022.

[30] Nojun Kwak and Chong-Ho Choi. Input feature selection for classification problems. *IEEE transactions on neural networks*, 13(1):143–159, 2002. doi: 10.1109/72.977291.

[31] Yun Fang, Peirong Xu, Jialiang Yang, and Yufang Qin. A quantile regression forest based method to predict drug response and assess prediction reliability. *PLoS One*, 13(10):e0205155, 2018.

[32] Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of machine learning research*, 7(6), 2006.

[33] Kerstin Lenhof, Lea Eckhart, Lisa-Marie Rolli, Andrea Volkamer, and Hans-Peter Lenhof. Reliable anti-cancer drug sensitivity prediction and prioritization. *ResearchSquare Preprint*, 11 2023. doi: https://doi.org/10.21203/rs.3.rs-3542373/v2.

[34] Melina Claussnitzer, Judy H Cho, Rory Collins, Nancy J Cox, Emmanouil T Dermitzakis, Matthew E Hurles, Sekar Kathiresan, Eimear E Kenny, Cecilia M Lindgren, Daniel G MacArthur, et al. A brief history of human disease genetics. *Nature*, 577(7789):179–189, 2020.

[35] Johns Hopkins University. Omim gene map statistics. `https://www.omim.org/statistics/geneMap`, 2024. [Online; accessed 5-January-2024].

[36] Tucker L Apgar and Charles R Sanders. Compendium of causative genes and their encoded proteins for common monogenic disorders. *Protein Science*, 31 (1):75–91, 2022.

[37] Ryan R Brinkman, Marie-Pierre Dubé, Guy A Rouleau, Andrew C Orr, and Mark E Samuels. Human monogenic disorders—a source of novel drug targets. *Nature Reviews Genetics*, 7(4):249–260, 2006.

[38] Freddie Bray, Mathieu Laversanne, Elisabete Weiderpass, and Isabelle Soerjomataram. The ever-increasing importance of cancer as a leading cause of premature death worldwide. *Cancer*, 127(16):3029–3030, 2021.

[39] National Cancer Institute. Age and cancer risk. `https://www.cancer.gov/about-cancer/causes-prevention/risk/age`, 2024. [Online; accessed 7-January-2024].

[40] German Federal Statistical Office (Destatis). Todesursachen die 10 häufigsten todesfälle durch krebs. `https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Gesundheit/Todesursachen/Tabellen/sterbefaelle-krebs-insgesamt.html?view=main[Print]`, 2024. [Online; accessed 8-January-2024].

[41] Steven A Frank. Genetic predisposition to cancer—insights from population genetics. *Nature reviews genetics*, 5(10):764–772, 2004.

[42] Wellcome Sanger Institute. Census overview. `https://cancer.sanger.ac.uk/census`, 2024. [Online; accessed 7-January-2024].

[43] Robert A Weinberg and Robert A Weinberg. *The biology of cancer.* WW Norton & Company, 2006.

[44] Douglas Hanahan and Robert A Weinberg. The hallmarks of cancer. *cell*, 100 (1):57–70, 2000.

[45] Douglas Hanahan and Robert A Weinberg. Hallmarks of cancer: the next generation. *cell*, 144(5):646–674, 2011.

[46] Douglas Hanahan. Hallmarks of cancer: new dimensions. *Cancer discovery*, 12(1):31–46, 2022.

[47] Ron Sender, Shai Fuchs, and Ron Milo. Revised estimates for the number of human and bacteria cells in the body. *PLoS biology*, 14(8):e1002533, 2016.

[48] Ian A Hatton, Eric D Galbraith, Nono SC Merleau, Teemu P Miettinen, Benjamin McDonald Smith, and Jeffery A Shander. The human cell count and size distribution. *Proceedings of the National Academy of Sciences*, 120(39): e2303077120, 2023.

[49] Wojciech Zakrzewski, Maciej Dobrzyński, Maria Szymonowicz, and Zbigniew Rybak. Stem cells: past, present, and future. *Stem cell research & therapy*, 10(1):1–22, 2019.

[50] Mark B Gerstein, Can Bruce, Joel S Rozowsky, Deyou Zheng, Jiang Du, Jan O Korbel, Olof Emanuelsson, Zhengdong D Zhang, Sherman Weissman, and Michael Snyder. What is a gene, post-encode? history and updated definition. *Genome research*, 17(6):669–681, 2007.

[51] Sara Martire and Laura A Banaszynski. The roles of histone variants in fine-tuning chromatin organization and function. *Nature reviews Molecular cell biology*, 21(9):522–541, 2020.

[52] Dmitry V Fyodorov, Bing-Rui Zhou, Arthur I Skoultchi, and Yawen Bai. Emerging roles of linker histones in regulating chromatin structure and function. *Nature reviews Molecular cell biology*, 19(3):192–206, 2018.

[53] Giacomo Cavalli and Edith Heard. Advances in epigenetics link genetics to the environment and disease. *Nature*, 571(7766):489–499, 2019.

[54] Olga Kelemen, Paolo Convertini, Zhaiyi Zhang, Yuan Wen, Manli Shen, Marina Falaleeva, and Stefan Stamm. Function of alternative splicing. *Gene*, 514 (1):1–30, 2013.

[55] DL Gonzalez, S Giannerini, and R Rosa. On the origin of degeneracy in the genetic code. *Interface Focus*, 9(6):20190038, 2019.

[56] Shahin Ramazi and Javad Zahiri. Post-translational modifications in proteins: resources, tools and prediction methods. *Database*, 2021:baab012, 2021.

[57] Ryan E Mills, W Stephen Pittard, Julienne M Mullaney, Umar Farooq, Todd H Creasy, Anup A Mahurkar, David M Kemeza, Daniel S Strassler, Chris P Ponting, Caleb Webber, et al. Natural genetic variation caused by small insertions and deletions in the human genome. *Genome research*, 21(6):830–839, 2011.

[58] Barkur S Shastry. Snps: impact on gene function and phenotype. *Single nucleotide polymorphisms: Methods and protocols*, pages 3–22, 2009.

[59] Jochen Graw. *Genetik 5.Auflage*, volume 5. Springer, 2010.

[60] Sigal Shachar, Omer Ziv, Sharon Avkin, Sheera Adar, John Wittschieben, Thomas Reißner, Stephen Chaney, Errol C Friedberg, Zhigang Wang, Thomas Carell, et al. Two-polymerase mechanisms dictate error-free and error-prone translesion dna synthesis in mammals. *The EMBO journal*, 28(4):383–393, 2009.

[61] Gene Levinson and George A Gutman. Slipped-strand mispairing: a major mechanism for dna sequence evolution. *Molecular biology and evolution*, 4(3): 203–221, 1987.

[62] Aline V Probst, Elaine Dunleavy, and Geneviève Almouzni. Epigenetic inheritance during the cell cycle. *Nature reviews Molecular cell biology*, 10(3): 192–206, 2009.

[63] Christina Pagiatakis, Elettra Musolino, Rosalba Gornati, Giovanni Bernardini, and Roberto Papait. Epigenetics of aging and disease: a brief overview. *Aging clinical and experimental research*, 33:737–745, 2021.

[64] Satish Kalari and Gerd P Pfeifer. Identification of driver and passenger dna methylation in cancer by epigenomic analysis. *Advances in genetics*, 70:277–308, 2010.

[65] Carla Sawan and Zdenko Herceg. Histone modifications and cancer. *Advances in genetics*, 70:57–85, 2010.

[66] Nazneen Rahman. Realizing the promise of cancer predisposition genes. *Nature*, 505(7483):302–308, 2014.

[67] Michael R Stratton, Peter J Campbell, and P Andrew Futreal. The cancer genome. *Nature*, 458(7239):719–724, 2009.

[68] Bert Vogelstein, Nickolas Papadopoulos, Victor E Velculescu, Shibin Zhou, Luis A Diaz Jr, and Kenneth W Kinzler. Cancer genome landscapes. *science*, 339(6127):1546–1558, 2013.

[69] Heidi Chial. Proto-oncogenes to oncogenes to cancer. `https://www.nature.com/scitable/topicpage/proto-oncogenes-to-oncogenes-to-cancer-883/`, 2008. [Online; accessed 17-January-2024].

[70] Jenny Fernando and Rob Jones. The principles of cancer treatment by chemotherapy. *Surgery (Oxford)*, 33(3):131–135, 2015.

[71] Haojie Jin, Liqin Wang, and René Bernards. Rational combinations of targeted cancer therapies: background, advances and challenges. *Nature Reviews Drug Discovery*, 22(3):213–234, 2023.

[72] Jiri Polivka Jr and Filip Janku. Molecular targets for cancer therapy in the pi3k/akt/mtor pathway. *Pharmacology & therapeutics*, 142(2):164–175, 2014.

[73] Helena Pópulo, José Manuel Lopes, and Paula Soares. The mtor signalling pathway in human cancer. *International journal of molecular sciences*, 13(2): 1886–1918, 2012.

[74] Pixu Liu, Hailing Cheng, Thomas M Roberts, and Jean J Zhao. Targeting the phosphoinositide 3-kinase pathway in cancer. *Nature reviews Drug discovery*, 8(8):627–644, 2009.

[75] Tian Xu, Dejuan Sun, Yi Chen, and Liang Ouyang. Targeting mtor for fighting diseases: A revisited review of mtor inhibitors. *European journal of medicinal chemistry*, 199:112391, 2020.

[76] Maartje W Rohaan, Sofie Wilgenhof, and John BAG Haanen. Adoptive cellular therapies: the current landscape. *Virchows Archiv*, 474:449–461, 2019.

[77] Özcan Met, Kasper Mølgaard Jensen, Christopher Aled Chamberlain, Marco Donia, and Inge Marie Svane. Principles of adoptive t cell therapy in cancer. In *Seminars in immunopathology*, volume 41, pages 49–58. Springer, 2019.

[78] Karim Vermaelen. Vaccine strategies to improve anti-cancer cellular immune responses. *Frontiers in immunology*, 10:8, 2019.

[79] Jihoon Kim, Bon-Kyoung Koo, and Juergen A Knoblich. Human organoids: model systems for human biology and medicine. *Nature Reviews Molecular Cell Biology*, 21(10):571–584, 2020.

[80] Virginia Brancato, Joaquim Miguel Oliveira, Vitor Manuel Correlo, Rui Luis Reis, and Subhas C Kundu. Could 3d models of cancer enhance drug screening? *Biomaterials*, 232:119744, 2020.

[81] Yunxin Lai, Xinru Wei, Shouheng Lin, Le Qin, Lin Cheng, and Peng Li. Current status and perspectives of patient-derived xenograft models in cancer research. *Journal of hematology & oncology*, 10:1–14, 2017.

[82] Mélanie AG Barbosa, Cristina PR Xavier, Rúben F Pereira, Vilma Petrikaitė, and M Helena Vasconcelos. 3d cell culture models as recapitulators of the tumor microenvironment for the screening of anti-cancer drugs. *Cancers*, 14 (1):190, 2021.

[83] G Emerens Wensink, Sjoerd G Elias, Jasper Mullenders, Miriam Koopman, Sylvia F Boj, Onno W Kranenburg, and Jeanine ML Roodhart. Patient-derived organoids as a predictive biomarker for treatment response in cancer patients. *NPJ precision oncology*, 5(1):30, 2021.

[84] Hui Gao, Joshua M Korn, Stéphane Ferretti, John E Monahan, Youzhen Wang, Mallika Singh, Chao Zhang, Christian Schnell, Guizhi Yang, Yun Zhang, et al. High-throughput screening using patient-derived tumor xenografts to predict clinical trial drug response. *Nature medicine*, 21(11): 1318–1325, 2015.

[85] Yu-Han Huang and Christopher R Vakoc. A biomarker harvest from one thousand cancer cell lines. *Cell*, 166(3):536–537, 2016.

[86] U.S. Department of Health and National Institutes of Health (NIH) Human Services. Nci-60 screening methodology. `https://dtp.cancer.gov/discovery_development/nci-60/methodology.htm`, 2021. [Online; accessed 5-December-2023].

[87] Wellcome Sanger Institute & Massachusetts General Hospital Cancer Centre. Genomics of drug sensitivity in cancer. `https://www.cancerrxgene.org`, 2023. [Online; accessed 5-December-2023].

[88] Amrita Basu, Nicole E Bodycombe, Jaime H Cheah, Edmund V Price, Ke Liu, Giannina I Schaefer, Richard Y Ebright, Michelle L Stewart, Daisuke Ito, Stephanie Wang, et al. An interactive resource to identify cancer genetic and lineage dependencies targeted by small molecules. *Cell*, 154(5):1151–1161, 2013.

[89] Matthew G Rees, Brinton Seashore-Ludlow, Jaime H Cheah, Drew J Adams, Edmund V Price, Shubhroz Gill, Sarah Javaid, Matthew E Coletti, Victor L Jones, Nicole E Bodycombe, et al. Correlating chemical sensitivity and basal gene expression reveals mechanism of action. *Nature chemical biology*, 12(2): 109–116, 2016.

[90] Steven M Corsello, Rohith T Nagari, Ryan D Spangler, Jordan Rossen, Mustafa Kocak, Jordan G Bryan, Ranad Humeidi, David Peck, Xiaoyun Wu, Andrew A Tang, et al. Discovering the anticancer potential of non-oncology drugs by systematic viability profiling. *Nature cancer*, 1(2):235–248, 2020.

[91] Broad Institute (depmap). Discovering the anti-cancer potential of non-oncology drugs by systematic viability profiling. `https://depmap.org/repurposing/`. [Online; accessed 5-December-2023].

[92] Susan L Holbeck, Richard Camalier, James A Crowell, Jeevan Prasaad Govindharajulu, Melinda Hollingshead, Lawrence W Anderson, Eric Polley, Larry Rubinstein, Apurva Srivastava, Deborah Wilsker, et al. The national cancer institute almanac: a comprehensive screening resource for the detection of anticancer drug pairs with enhanced therapeutic activity. *Cancer research*, 77 (13):3564–3576, 2017.

[93] Jennifer O'Neil, Yair Benita, Igor Feldman, Melissa Chenard, Brian Roberts, Yaping Liu, Jing Li, Astrid Kral, Serguei Lejnine, Andrey Loboda, et al. An unbiased oncology compound screen to identify novel combination strategies. *Molecular cancer therapeutics*, 15(6):1155–1162, 2016.

[94] Michael P Menden, Dennis Wang, Mike J Mason, Bence Szalai, Krishna C Bulusu, Yuanfang Guan, Thomas Yu, Jaewoo Kang, Minji Jeon, Russ Wolfinger, et al. Community assessment to advance computational prediction of cancer drug combinations in a pharmacogenomic screen. *Nature communications*, 10 (1):2674, 2019.

[95] Anna Astashkina, Brenda Mann, and David W Grainger. A critical evaluation of in vitro cell culture models for high-throughput drug screening and toxicity. *Pharmacology & therapeutics*, 134(1):82–106, 2012.

[96] Martin J Stoddart. *Mammalian cell viability: methods and protocols*, volume 740. Springer, 2011.

[97] Terry L Riss, Richard A Moravec, Andrew L Niles, Sarah Duellman, Hélène A Benink, Tracy J Worzella, and Lisa Minor. Cell viability assays. *Assay Guidance Manual [Internet]*, 2016.

[98] Wellcome Sanger Institute & Massachusetts General Hospital Cancer Centre. Genomics of drug sensitivity in cancer, help and documentation. `https://www.cancerrxgene.org/help#t_curve`, 2023. [Online; accessed 7-December-2023].

[99] Daniel J Vis, Lorenzo Bombardelli, Howard Lightfoot, Francesco Iorio, Mathew J Garnett, and Lodewyk FA Wessels. Multilevel models improve precision and speed of ic50 estimates. *Pharmacogenomics*, 17(7):691–700, 2016.

[100] Nikita Pozdeyev, Minjae Yoo, Ryan Mackie, Rebecca E Schweppe, Aik Choon Tan, and Bryan R Haugen. Integrating heterogeneous drug sensitivity data from cancer pharmacogenomic studies. *Oncotarget*, 7(32):51619, 2016.

[101] Marc Hafner, Mario Niepel, Mirra Chung, and Peter K Sorger. Growth rate inhibition metrics correct for confounders in measuring sensitivity to cancer drugs. *Nature methods*, 13(6):521–527, 2016.

[102] Mary J Lindstrom and Douglas M Bates. Nonlinear mixed effects models for repeated measures data. *Biometrics*, pages 673–687, 1990.

[103] Elizabeth A Brooks, Sualyneth Galarza, Maria F Gencoglu, R Chase Cornelison, Jennifer M Munson, and Shelly R Peyton. Applicability of drug response metrics for cancer studies using biomaterials. *Philosophical Transactions of the Royal Society B*, 374(1779):20180226, 2019.

[104] Francesco Iorio, Theo A Knijnenburg, Daniel J Vis, Graham R Bignell, Michael P Menden, Michael Schubert, Nanne Aben, Emanuel Gonçalves, Syd Barthorpe, Howard Lightfoot, et al. A landscape of pharmacogenomic interactions in cancer supplement. `https://ars.els-cdn.com/content/image/1-s2.0-S0092867416307462-mmc1.pdf`, 2016. [Online; accessed 13-December-2023].

[105] Wellcome Sanger Institute & Massachusetts General Hospital Cancer Centre. Genomics of drug sensitivity in cancer, help and documentation. `https://www.cancerrxgene.org/help`, 2023. [Online; accessed 13-December-2023].

[106] David Jones, Keiran M Raine, Helen Davies, Patrick S Tarpey, Adam P Butler, Jon W Teague, Serena Nik-Zainal, and Peter J Campbell. cgpcavemanwrapper: simple execution of caveman in order to detect somatic single nucleotide variants in ngs data. *Current protocols in bioinformatics*, 56(1):15–10, 2016.

[107] Kai Ye, Marcel H Schulz, Quan Long, Rolf Apweiler, and Zemin Ning. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 25(21): 2865–2871, 2009.

[108] Wellcome Sanger Institute. Cosmic what is the cell lines project? `https://cancer.sanger.ac.uk/cell_lines/help/desc`, . [Online; accessed 15-December-2023].

[109] Francesco Iorio. Gdsc1000 resources. `https://www.cancerrxgene.org/gdsc1000/GDSC1000_WebResources/Home.html`. [Online; accessed 15-December-2023].

[110] Affymetrix. Data sheet genome-wide human snp array 6.0. `https://assets.thermofisher.com/TFS-Assets/LSG/brochures/genomewide_snp6_datasheet.pdf`, 2009. [Online; accessed 14-December-2023].

[111] Wellcome Sanger Institute. Cosmic cnv overview. `https://cancer.sanger.ac.uk/cosmic/help/cnv/overview`, . [Online; accessed 14-December-2023].

[112] Chris D Greenman, Graham Bignell, Adam Butler, Sarah Edkins, Jon Hinton, Dave Beare, Sajani Swamy, Thomas Santarius, Lina Chen, Sara Widaa, et al. Picnic: an algorithm to predict absolute allelic copy number variation with microarray cancer data. *Biostatistics*, 11(1):164–175, 2010.

[113] Peter Van Loo, Silje H Nordgard, Ole Christian Lingjærde, Hege G Russnes, Inga H Rye, Wei Sun, Victor J Weigman, Peter Marynen, Anders Zetterberg, Bjørn Naume, et al. Allele-specific copy number analysis of tumors. *Proceedings of the National Academy of Sciences*, 107(39):16910–16915, 2010.

[114] Rafael A Irizarry, Bridget Hobbs, Francois Collin, Yasmin D Beazer-Barclay, Kristen J Antonellis, Uwe Scherf, and Terence P Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2003.

[115] Manhong Dai, Pinglang Wang, Andrew D Boyd, Georgi Kostov, Brian Athey, Edward G Jones, William E Bunney, Richard M Myers, Terry P Speed, Huda Akil, et al. Evolving gene/transcript definitions significantly alter the interpretation of genechip data. *Nucleic acids research*, 33(20):e175–e175, 2005.

[116] Nils J Nilsson. *The quest for artificial intelligence.* Cambridge University Press, 2009.

[117] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433. URL `https://doi.org/10.1093/mind/LIX.236.433`.

[118] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

[119] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[120] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[121] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[122] Alexander L Fradkov. Early history of machine learning. *IFAC-PapersOnLine*, 53(2):1385–1390, 2020.

[123] Tom M Mitchell. Machine learning, 1997.

[124] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[125] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. 2006. *Cambridge, Massachusettes: The MIT Press View Article*, 2, 2006.

[126] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. *An introduction to statistical learning*, volume 112. Springer, 2013.

[127] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[128] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020.

[129] Ladislav Rampášek, Daniel Hidru, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. Dr. vae: improving drug response prediction via modeling of drug perturbation effects. *Bioinformatics*, 35(19):3743–3751, 2019.

[130] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[131] Mingyang Liu, Xiaotong Shen, and Wei Pan. Deep reinforcement learning for personalized treatment recommendation. *Statistics in medicine*, 41(20): 4034–4056, 2022.

[132] European Commission. Ethics guidelines for trustworthy ai. `https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai`, 2022. [Online; accessed 18-October-2023].

[133] Francis Galton. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

[134] Karl Pearson. Vii. note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58(347-352):240–242, 1895.

[135] Charles Spearman. The proof and measurement of association between two things. 1961.

[136] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[137] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020.

[138] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[139] Matjaž Kukar and Igor Kononenko. Reliable classifications with machine learning. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*, pages 219–231. Springer, 2002.

[140] Vu-Linh Nguyen, Sébastien Destercke, Marie-Hélène Masson, and Eyke Hüllermeier. Reliable multi-class classification based on pairwise epistemic and aleatoric uncertainty. In *27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 5089–5095, 2018.

[141] Giovanna Nicora, Miguel Rios, Ameen Abu-Hanna, and Riccardo Bellazzi. Evaluating pointwise reliability of machine learning prediction. *Journal of Biomedical Informatics*, 127:103996, 2022.

[142] Benjamin Kompa, Jasper Snoek, and Andrew L Beam. Second opinion needed: communicating uncertainty in medical machine learning. *NPJ Digital Medicine*, 4(1):4, 2021.

[143] Cornelia Gruber, Patrick Oliver Schenk, Malte Schierholz, Frauke Kreuter, and Göran Kauermann. Sources of uncertainty in machine learning–a statisticians' view. *arXiv preprint arXiv:2305.16703*, 2023.

[144] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110:457–506, 2021.

[145] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[146] Fergus Imrie, Robert Davis, and Mihaela van der Schaar. Multiple stakeholders drive diverse interpretability requirements for machine learning in healthcare. *Nature Machine Intelligence*, pages 1–6, 2023.

[147] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, pages 8–13, 2017.

[148] Maya Krishnan. Against interpretability: a critical examination of the interpretability problem in machine learning. *Philosophy & Technology*, 33(3): 487–502, 2020.

[149] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[150] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3): 31–57, 2018.

[151] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

[152] Michael P Menden, Francesco Iorio, Mathew Garnett, Ultan McDermott, Cyril H Benes, Pedro J Ballester, and Julio Saez-Rodriguez. Machine learning prediction of cancer cell sensitivity to drugs based on genomic and chemical properties. *PLoS one*, 8(4):e61318, 2013. doi: https://doi.org/10.1371/journal.pone.0061318.

[153] Naiqian Zhang, Haiyun Wang, Yun Fang, Jun Wang, Xiaoqi Zheng, and X Shirley Liu. Predicting anticancer drug responses using a dual-layer integrated cell line-drug network model. *PLoS computational biology*, 11(9): e1004498, 2015. doi: https://doi.org/10.1371/journal.pcbi.1004498.

[154] Lin Wang, Xiaozhong Li, Louxin Zhang, and Qiang Gao. Improved anticancer drug response prediction in cell lines using matrix factorization with similarity regularization. *BMC cancer*, 17(1):1–12, 2017.

[155] Raziur Rahman, Kevin Matlock, Souparno Ghosh, and Ranadip Pal. Heterogeneity aware random forest for drug sensitivity prediction. *Scientific reports*, 7(1):1–11, 2017. doi: https://doi.org/10.1038/s41598-017-11665-4.

[156] Kevin Matlock, Carlos De Niz, Raziur Rahman, Souparno Ghosh, and Ranadip Pal. Investigation of model stacking for drug sensitivity prediction.

BMC bioinformatics, 19(3):21–33, 2018. doi: https://doi.org/10.1186/s12859-018-2060-2.

[157] Xiao He, Lukas Folkman, and Karsten Borgwardt. Kernelized rank learning for personalized drug recommendation. *Bioinformatics*, 34(16):2808–2816, 2018.

[158] Amrita Basu, Ritwik Mitra, Han Liu, Stuart L Schreiber, and Paul A Clemons. Rwen: response-weighted elastic net for prediction of chemosensitivity of cancer cell lines. *Bioinformatics*, 34(19):3332–3339, 2018.

[159] Yoosup Chang, Hyejin Park, Hyun-Jin Yang, Seungju Lee, Kwee-Yum Lee, Tae Soon Kim, Jongsun Jung, and Jae-Min Shin. Cancer drug response profile scan (cdrscan): a deep learning model that predicts drug effectiveness from cancer genomic signature. *Scientific reports*, 8(1):8857, 2018.

[160] Hui Liu, Yan Zhao, Lin Zhang, and Xing Chen. Anti-cancer drug response prediction using neighbor-based collaborative filtering with global effect removal. *Molecular Therapy-Nucleic Acids*, 13:303–311, 2018. doi: https://doi.org/10.1016/j.omtn.2018.09.011.

[161] Yu-Chiao Chiu, Hung-I Harry Chen, Tinghe Zhang, Songyao Zhang, Aparna Gorthi, Li-Ju Wang, Yufei Huang, and Yidong Chen. Predicting drug response of tumors from integrated genomic profiles by deep neural networks. *BMC medical genomics*, 12(1):143–155, 2019. doi: https://doi.org/10.1186/s12920-018-0460-9.

[162] Ali Oskooei, Matteo Manica, Roland Mathis, and María Rodríguez Martínez. Network-based biased tree ensembles (netbite) for drug sensitivity prediction and drug sensitivity biomarker identification in cancer. *Scientific reports*, 9 (1):1–13, 2019. doi: https://doi.org/10.1038/s41598-019-52093-w.

[163] Lei Deng, Yideng Cai, Wenhao Zhang, Wenyi Yang, Bo Gao, and Hui Liu. Pathway-guided deep neural network toward interpretable and predictive modeling of drug sensitivity. *Journal of Chemical Information and Modeling*, 60 (10):4497–4505, 2020.

[164] Yi-Ching Tang and Assaf Gottlieb. Explainable drug sensitivity prediction through cancer pathway enrichment. *Scientific reports*, 11(1):3128, 2021.

[165] Tuan Nguyen, Giang TT Nguyen, Thin Nguyen, and Duc-Hau Le. Graph convolutional networks for drug response prediction. *IEEE/ACM transactions on computational biology and bioinformatics*, 19(1):146–154, 2021.

[166] Zachary Stanfield, Mustafa Coşkun, and Mehmet Koyutürk. Drug response prediction as a link prediction problem. *Scientific reports*, 7(1):40321, 2017.

[167] Fei Zhang, Minghui Wang, Jianing Xi, Jianghong Yang, and Ao Li. A novel heterogeneous network-based method for drug response prediction in cancer cell lines. *Scientific reports*, 8(1):3355, 2018.

[168] Ran Su, Xinyi Liu, Leyi Wei, and Quan Zou. Deep-resp-forest: a deep forest model to predict anti-cancer drug response. *Methods*, 166:91–102, 2019.

[169] Adnan Qayyum, Junaid Qadir, Muhammad Bilal, and Ala Al-Fuqaha. Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering*, 14:156–180, 2020.

[170] Richard Bellman and Robert Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

[171] Lei Chen. *Curse of Dimensionality*, pages 545–546. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_133. URL `https://doi.org/10.1007/978-0-387-39940-9_133`.

[172] Zena M Hira and Duncan F Gillies. A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics*, 2015, 2015.

[173] Weikuan Jia, Meili Sun, Jian Lian, and Sujuan Hou. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693, 2022.

[174] Yvan Saeys, Inaki Inza, and Pedro Larranaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[175] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[176] Leonard Kaufman. Partitioning around medoids (program pam). *Finding groups in data*, 344:68–125, 1990.

[177] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[178] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.

[179] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[180] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[181] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. doi: https://doi.org/10.1023/A:1010933404324.

[182] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[183] Suggests RColorBrewer and Maintainer Andy Liaw. Package 'randomforest'. *University of California, Berkeley: Berkeley, CA, USA*, 2018.

[184] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5: 197–227, 1990.

[185] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

[186] Bo Jiang, Si Chen, Beibei Wang, and Bin Luo. Mglnn: Semi-supervised learning via multiple graph cooperative learning neural networks. *Neural Networks*, 153:204–214, 2022.

[187] Russell Reed and Robert J MarksII. *Neural smithing: supervised learning in feedforward artificial neural networks.* Mit Press, 1999.

[188] Thomas Fösel, Petru Tighineanu, Talitha Weiss, and Florian Marquardt. Reinforcement learning with neural networks for quantum feedback. *Physical Review X*, 8(3):031084, 2018.

[189] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[190] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[191] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. An introduction to statistical learning: With applications in python. *(No Title)*, 2023.

[192] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[193] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[194] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[195] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[196] Ryan Tibshirani. Conformal prediction.

[197] Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[198] Vladimir Vovk. Conditional validity of inductive conformal predictors. In *Asian conference on machine learning*, pages 475–490. PMLR, 2012.

[199] Yaniv Romano, Matteo Sesia, and Emmanuel Candes. Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems*, 33:3581–3591, 2020.

[200] Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193*, 2020.

[201] Matteo Fontana, Gianluca Zeni, and Simone Vantini. Conformal prediction: a unified review of theory and new challenges. *Bernoulli*, 29(1):1–23, 2023.

[202] Ulf Norinder, Lars Carlsson, Scott Boyer, and Martin Eklund. Introducing conformal prediction in predictive modeling. a transparent and flexible alternative to applicability domain determination. *Journal of chemical information and modeling*, 54(6):1596–1603, 2014.

[203] Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

[204] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. Interpretability of machine learning-based prediction models in healthcare. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(5):e1379, 2020.

[205] Anil P Kamath, Narendra K Karmarkar, KG Ramakrishnan, and Mauricio GC Resende. A continuous approach to inductive inference. *Mathematical programming*, 57:215–238, 1992.

[206] Francisco Sanchez-Vega, Marco Mina, Joshua Armenia, Walid K Chatila, Augustin Luna, Konnor C La, Sofia Dimitriadoy, David L Liu, Havish S Kantheti, Sadegh Saghafinia, et al. Oncogenic signaling pathways in the cancer genome atlas. *Cell*, 173(2):321–337, 2018.

[207] Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis: Nonparametric discrimination: Small sample performance. 1952.

[208] Max Kuhn. Caret: classification and regression training. *Astrophysics Source Code Library*, pages ascl–1505, 2015.

[209] Eun-Young Lee, Jung Yeon Yu, A Rome Paek, So Hee Lee, Hyonchol Jang, Soo Young Cho, June Hyuk Kim, Hyun Guy Kang, Tak Yun, Sung Eun Oh, et al. Targeting tjp1 attenuates cell–cell aggregation and modulates chemosensitivity against doxorubicin in leiomyosarcoma. *Journal of Molecular Medicine*, 98:761–773, 2020.

[210] Nico Gerstner, Tim Kehl, Kerstin Lenhof, Anne Müller, Carolin Mayer, Lea Eckhart, Nadja Liddy Grammes, Caroline Diener, Martin Hart, Oliver Hahn, et al. Genetrail 3: advanced high-throughput enrichment analysis. *Nucleic Acids Research*, 48(W1):W515–W520, 2020.

[211] Karthic Swaminathan, Andrew Campbell, Vassilis Papalazarou, Farah Jaber-Hijazi, Colin Nixon, Ewan McGhee, Douglas Strathdee, Owen J Sansom, and Laura M Machesky. The rac1 target nckap1 plays a crucial role in the progression of braf; pten-driven melanoma in mice. *Journal of Investigative Dermatology*, 141(3):628–637, 2021.

[212] Tong Gan, Ashley T Stevens, Xiaopeng Xiong, Yang-An Wen, Trevor N Farmer, Austin T Li, Payton D Stevens, Sanam Golshani, Heidi L Weiss, B Mark Evers, et al. Inhibition of protein tyrosine phosphatase receptor type f suppresses wnt signaling in colorectal cancer. *Oncogene*, 39(44):6789–6801, 2020.

[213] Jian-Hua Mao, Il-Jin Kim, Di Wu, Joan Climent, Hio Chung Kang, Reyno DelRosario, and Allan Balmain. Fbxw7 targets mtor for degradation and cooperates with pten in tumor suppression. *Science*, 321(5895):1499–1502, 2008.

[214] Reuben J Shaw, Nabeel Bardeesy, Brendan D Manning, Lyle Lopez, Monica Kosmatka, Ronald A DePinho, and Lewis C Cantley. The lkb1 tumor suppressor negatively regulates mtor signaling. *Cancer cell*, 6(1):91–99, 2004.

[215] Hossein Sharifi-Noghabi, Soheil Jahangiri-Tazehkand, Petr Smirnov, Casey Hon, Anthony Mammoliti, Sisira Kadambat Nair, Arvind Singh Mer, Martin

Ester, and Benjamin Haibe-Kains. Drug sensitivity prediction from cell line-based pharmacogenomics data: guidelines for developing machine learning models. *Briefings in bioinformatics*, 22(6):bbab294, 2021.

[216] Jongwoo Song. Bias corrections for random forest in regression using residual rotation. *Journal of the Korean Statistical Society*, 44(2):321–326, 2015. doi: https://doi.org/10.1016/j.jkss.2015.01.003.

[217] B. Greenwell, B. Boehmke, J. Cunningham, et al. gbm: Generalized boosted regression models. `https://CRAN.R-project.org/package=gbm`, 2020. R package version 2.1.8.

[218] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39(5):1–13, 2011. doi: https://doi.org/10.18637/jss.v039.i05. R package version 4.1.1.

[219] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Python library tensorflow-gpu version 1.13.1.

[220] F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015. Python library version 2.3.1.

[221] Minh Thanh Vo, Anh H Vo, Trang Nguyen, Rohit Sharma, and Tuong Le. Dealing with the class imbalance problem in the detection of fake job descriptions. *Computers, Materials & Continua*, 68(1):521–535, 2021. doi: DOI:10.32604/cmc.2021.015645.

[222] Anjana Gosain and Saanchi Sardana. Handling class imbalance problem using oversampling techniques: A review. In *2017 international conference on advances in computing, communications and informatics (ICACCI)*, pages 79–85. IEEE, 2017. doi: 10.1109/ICACCI.2017.8125820.

[223] KPNV Satyasree and J Murthy. An exhaustive literature review on class imbalance problem. *Int J Emerg Trends Technol Comput Sci*, 2:109–118, 2013.

[224] Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2-3):427–436, 2008. doi: https://doi.org/10.1016/j.neunet.2007.12.031.

[225] Yueh-Min Huang, Chun-Min Hung, and Hewijin Christine Jiau. Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. *Nonlinear Analysis: Real World Applications*, 7(4): 720–747, 2006. doi: https://doi.org/10.1016/j.nonrwa.2005.04.006.

[226] Lorenza Pasqualini, Huajie Bu, Martin Puhr, Narisu Narisu, Johannes Rainer, Bettina Schlick, Georg Schäfer, Mihaela Angelova, Zlatko Trajanoski, Stefan T. Börno, Michal R. Schweiger, Christian Fuchsberger, and Helmut Klocker. miR-22 and miR-29a are members of the androgen receptor cistrome modulating LAMC1 and mcl-1 in prostate cancer. *Molecular Endocrinology*, 29(7):1037–1054, jul 2015. doi: 10.1210/me.2014-1358.

[227] Le Zhang, Cuixia Li, and Xiulan Su. Emerging impact of the long noncoding RNA MIR22hg on proliferation and apoptosis in multiple human cancers. *Journal of Experimental & Clinical Cancer Research*, 39(1), dec 2020. doi: 10.1186/s13046-020-01784-8.

[228] Steven M Chan, Daniel Thomas, M Ryan Corces-Zimmerman, Seethu Xavy, Suchita Rastogi, Wan-Jen Hong, Feifei Zhao, Bruno C Medeiros, David A Tyvoll, and Ravindra Majeti. Isocitrate dehydrogenase 1 and 2 mutations induce BCL-2 dependence in acute myeloid leukemia. *Nature Medicine*, 21(2): 178–184, jan 2015. doi: 10.1038/nm.3788.

[229] M. Rahmani, M. M. Aust, E. Hawkins, R. E. Parker, M. Ross, M. Kmieciak, L. B. Reshko, K. A. Rizzo, C. I. Dumur, A. Ferreira-Gonzalez, and S. Grant. Co-administration of the mTORC1/TORC2 inhibitor INK128 and the bcl-2/bcl-xL antagonist ABT-737 kills human myeloid leukemia cells through mcl-1 down-regulation and AKT inactivation. *Haematologica*, 100(12):1553–1563, oct 2015. doi: 10.3324/haematol.2015.130351.

[230] Matthew G Rees, Brinton Seashore-Ludlow, Jaime H Cheah, Drew J Adams,

Edmund V Price, Shubhroz Gill, Sarah Javaid, Matthew E Coletti, Victor L Jones, Nicole E Bodycombe, Christian K Soule, Benjamin Alexander, Ava Li, Philip Montgomery, Joanne D Kotz, C Suk-Yee Hon, Benito Munoz, Ted Liefeld, Vlado Dančík, Daniel A Haber, Clary B Clish, Joshua A Bittker, Michelle Palmer, Bridget K Wagner, Paul A Clemons, Alykhan F Shamji, and Stuart L Schreiber. Correlating chemical sensitivity and basal gene expression reveals mechanism of action. *Nature Chemical Biology*, 12(2):109–116, dec 2015. doi: 10.1038/nchembio.1986.

[231] Kathleen I. Pishas, Susan J. Neuhaus, Mark T. Clayer, Andreas W. Schreiber, David M. Lawrence, Michelle Perugini, Robert J. Whitfield, Gelareh Farshid, Jim Manavis, Steve Chryssidis, Bronwen J. Mayo, Rebecca C. Haycox, Kristen Ho, Michael P. Brown, Richard J. D'Andrea, Andreas Evdokiou, David M. Thomas, Jayesh Desai, David F. Callen, and Paul M. Neilsen. Nutlin-3a efficacy in sarcoma predicted by transcriptomic and epigenetic profiling. *Cancer Research*, 74(3):921–931, dec 2013. doi: 10.1158/0008-5472.can-13-2424.

[232] Maryam Zanjirband, Richard J. Edmondson, and John Lunec. Pre-clinical efficacy and synergistic potential of the MDM2-p53 antagonists, nutlin-3 and RG7388, as single agents and in combined treatment with cisplatin in ovarian cancer. *Oncotarget*, 7(26):40115–40134, may 2016. doi: 10.18632/oncotarget.9499.

[233] Kensuke Kumamoto, Elisa A. Spillare, Kaori Fujita, Izumi Horikawa, Taro Yamashita, Ettore Appella, Makoto Nagashima, Seiichi Takenoshita, Jun Yokota, and Curtis C. Harris. Nutlin-3a activates p53 to both down-regulate inhibitor of growth 2 and up-regulate mir-34a, mir-34b, and mir-34c expression, and induce senescence. *Cancer Research*, 68(9):3193–3203, may 2008. doi: 10.1158/0008-5472.can-07-2780.

[234] Manoj Kumawat, Ranu Singh, Irungbam Karuna, Neeraj Ahlawat, and Sushma Ahlawat. Salmonella typhimurium peptidyl-prolyl cis–trans isomerase c (ppiase c) plays a substantial role in protein folding to maintain the protein structure. *World Journal of Microbiology and Biotechnology*, 36(11):168, 2020.

[235] Arye Elfenbein and Michael Simons. Syndecan-4 signaling at a glance. *Journal of cell science*, 126(17):3799–3804, 2013.

[236] Pan Xie, Fu-Qiang Yuan, Ma-Sha Huang, Wei Zhang, Hong-Hao Zhou, Xi Li, and Zhao-Qian Liu. Dcbld2 affects the development of colorectal cancer via emt and angiogenesis and modulates 5-fu drug resistance. *Frontiers in Cell and Developmental Biology*, 9:669285, 2021.

[237] Shuai Cheng, Liang-Yan Wang, Chuan-Hui Wang, Fa-Kai Wang, Bing Zhu, Peng Zhang, and Guo-Hua Wang. Transmembrane protein dcbld2 is correlated with poor prognosis and affects phenotype by regulating epithelial-mesenchymal transition in human glioblastoma cells. *Neuroreport*, 32(6):507–517, 2021.

[238] Jie He, Hongli Huang, Yanlei Du, Dong Peng, Youlian Zhou, Yuyuan Li, Hong Wang, Yongjian Zhou, and Yuqiang Nie. Association of dcbld2 upregulation with tumor progression and poor survival in colorectal cancer. *Cellular Oncology*, 43:409–420, 2020.

[239] Lara Schneider, Tim Kehl, Kristina Thedinga, Nadja Liddy Grammes, Christina Backes, Christopher Mohr, Benjamin Schubert, Kerstin Lenhof, Nico Gerstner, Andreas Daniel Hartkopf, et al. Clinomicstrailbc: a visual analytics tool for breast cancer treatment stratification. *Bioinformatics*, 35(24):5171–5181, 2019. doi: https://doi.org/10.1093/bioinformatics/btz302.

[240] Huy Phan, Lars Hertel, Marco Maass, Philipp Koch, and Alfred Mertins. Car-forest: Joint classification-regression decision forests for overlapping audio event detection. *arXiv preprint arXiv:1607.02306*, 2016. doi: https://doi.org/10.48550/arXiv.1607.02306.

[241] Ben Glocker, Olivier Pauly, Ender Konukoglu, and Antonio Criminisi. Joint classification-regression forests for spatially structured multi-object segmentation. In *European conference on computer vision*, pages 870–881. Springer, 2012. doi: https://doi.org/10.1007/978-3-642-33765-9_62.

[242] Krzysztof Koras, Dilafruz Juraeva, Julian Kreis, Johanna Mazur, Eike Staub, and Ewa Szczurek. Feature selection strategies for drug sensitivity prediction. *Scientific Reports*, 10(1):9377, 2020.

[243] In Sock Jang, Elias Chaibub Neto, Justin Guinney, Stephen H Friend, and Adam A Margolin. Systematic assessment of analytical methods for drug

sensitivity prediction from cancer cell line data. In *Biocomputing 2014*, pages 63–74. World Scientific, 2014.

[244] Francisco Martínez-Jiménez, Ferran Muiños, Inés Sentís, Jordi Deu-Pons, Iker Reyes-Salazar, Claudia Arnedo-Pac, Loris Mularoni, Oriol Pich, Jose Bonet, Hanna Kranas, et al. A compendium of mutational cancer driver genes. *Nature Reviews Cancer*, 20(10):555–572, 2020.

[245] Andrey Nikolaevich Kolmogorov. Sulla determinazione empirica di una legge didistribuzione. *Giornale dell'Instituto Italiano degli Attuari*, 4:83–91, 1933.

[246] Nikolai Vasilieyvich Smirnov. Estimate of deviation between empirical distribution functions in two independent samples. *Bulletin Moscow University*, 2 (2):3–16, 1939.

[247] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289 – 300, 1995. doi: https://doi.org/10.1111/j.2517-6161.1995.tb02031.x.

[248] Monalisa Mandal and Anirban Mukhopadhyay. An improved minimum redundancy maximum relevance approach for feature selection in gene expression data. *Procedia Technology*, 10:20–27, 2013. doi: https://doi.org/10.1016/j.protcy.2013.12.332.

[249] Milos Radovic, Mohamed Ghalwash, Nenad Filipovic, and Zoran Obradovic. Minimum redundancy maximum relevance feature selection approach for temporal gene expression data. *BMC bioinformatics*, 18(1):1–14, 2017.

[250] Jorge M Arevalillo and Hilario Navarro. Exploring correlations in gene expression microarray data for maximum predictive–minimum redundancy biomarker selection and classification. *Computers in Biology and Medicine*, 43(10):1437–1443, 2013.

[251] Ioulia Karagiannaki, Yannis Pantazis, Ekaterini Chatzaki, and Ioannis Tsamardinos. Pathway activity score learning for dimensionality reduction of gene expression data. In *Discovery Science*, pages 246–261. Springer International Publishing, 2020. doi: https://doi.org/10.1007/978-3-030-61527-7_17.

[252] M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77 (1):1–17, 2017. doi: https://doi.org/10.18637/jss.v077.i01. R package version 0.12.1.

[253] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

[254] Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9), 2023.

[255] Jonathan Alvarsson, Staffan Arvidsson McShane, Ulf Norinder, and Ola Spjuth. Predicting with confidence: using conformal prediction in drug discovery. *Journal of Pharmaceutical Sciences*, 110(1):42–49, 2021.

[256] Andrea Morger, Miriam Mathea, Janosch H Achenbach, Antje Wolf, Roland Buesen, Klaus-Juergen Schleifer, Robert Landsiedel, and Andrea Volkamer. Knowtox: pipeline and case study for confident prediction of potential toxic effects of compounds in early phases of development. *Journal of Cheminformatics*, 12(1):24, 2020.

[257] Andrea Morger, Fredrik Svensson, Staffan Arvidsson McShane, Niharika Gauraha, Ulf Norinder, Ola Spjuth, and Andrea Volkamer. Assessing the calibration in toxicological in vitro models with conformal prediction. *Journal of Cheminformatics*, 13(1):35, 2021.

[258] Dane R Liston and Myrtle Davis. Clinically relevant concentrations of anticancer drugs: A guide for nonclinical studiesguide to clinical exposures of anticancer drugs. *Clinical cancer research*, 23(14):3489–3498, 2017.

[259] Ichiro Takeuchi, Quoc Le, Timothy Sears, Alexander Smola, et al. Nonparametric quantile estimation. 2006.

[260] Roger Koenker. *Quantile regression*, volume 38. Cambridge university press, 2005.

[261] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Technical Report*, 1055, 2002.

[262] Bulat Zagidullin, Jehad Aldahdooh, Shuyu Zheng, Wenyu Wang, Yinyin Wang, Joseph Saad, Alina Malyutina, Mohieddin Jafari, Ziaurrehman Tanoli, Alberto Pessia, et al. Drugcomb: an integrative cancer drug combination data portal. *Nucleic acids research*, 47(W1):W43–W51, 2019. doi: https://doi.org/10.1093/nar/gkz337.

[263] Janette Vazquez and Julio C Facelli. Conformal prediction in clinical medical sciences. *Journal of Healthcare Informatics Research*, 6(3):241–252, 2022.

[264] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[265] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

[266] Feifei Li, Changqi Zhao, and Lili Wang. Molecular-targeted agents combination therapy for cancer: developments and potentials. *International journal of cancer*, 134(6):1257–1269, 2014.

[267] PS Chowdhury, K Chamoto, and T Honjo. Combination therapy strategies for improving pd-1 blockade efficacy: a new era in cancer immunotherapy. *Journal of internal medicine*, 283(2):110–120, 2018.

[268] Amila Suraweera, Kenneth J O'Byrne, and Derek J Richard. Combination therapy with histone deacetylase inhibitors (hdaci) for the treatment of cancer: achieving the full therapeutic potential of hdaci. *Frontiers in oncology*, 8:92, 2018.

[269] Yihui Li, David Earl Hostallero, and Amin Emad. Interpretable deep learning architectures for improving drug response prediction performance: myth or reality? *Bioinformatics*, 39(6):btad390, 2023.

[270] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[271] Franz Pernkopf, Robert Peharz, and Sebastian Tschiatschek. Introduction to probabilistic graphical models. In *Academic Press Library in Signal Processing*, volume 1, pages 989–1064. Elsevier, 2014.

[272] Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes, 2014.

[273] Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks. *arXiv preprint arXiv:1901.03704*, 2019.

[274] American Cancer Society. Oncogenes, tumor suppressor genes, and dna repair genes. `https://www.cancer.org/cancer/understanding-cancer/genes-and-cancer/oncogenes-tumor-suppressor-genes.html`, 2024. [Online; accessed 17-January-2024].

[275] Pan Pantziarka, Rica Capistrano I, Arno De Potter, Liese Vandeborne, and Gauthier Bouche. An open access database of licensed cancer drugs. *Frontiers in pharmacology*, 12:627574, 2021.

[276] David G Covell, Ruili Huang, and Anders Wallqvist. Anticancer medicines in development: assessment of bioactivity profiles within the national cancer institute anticancer screening data. *Molecular cancer therapeutics*, 6(8):2261–2270, 2007. doi: https://doi.org/10.1158/1535-7163.MCT-06-0787.

[277] David L Masica and Rachel Karchin. Collections of simultaneously altered genes as biomarkers of cancer cell drug response. *Cancer research*, 73(6): 1699–1708, 2013. doi: https://doi.org/10.1158/0008-5472.CAN-12-3122.

[278] MP Menden. In silico models of drug response in cancer cell lines based on various molecular descriptors. *University of Cambridge*, 2019(9):15918, 2016.

[279] Kristina Preuer, Richard PI Lewis, Sepp Hochreiter, Andreas Bender, Krishna C Bulusu, and Günter Klambauer. Deepsynergy: predicting anti-cancer drug synergy with deep learning. *Bioinformatics*, 34(9):1538–1546, 2018. doi: https://doi.org/10.1093/bioinformatics/btx806.

[280] Xubin Li, Elisabeth K Dowling, Gonghong Yan, Zeynep Dereli, Behnaz Bozorgui, Parisa Imarinad, Jacob H Elnaggar, Augustin Luna, David G Menter, Patrick G Pilie, Timothy A Yap, Scott Kopetz, Chris Sander, and Anil Korkut.

Precision combination therapies based on recurrent oncogenic co-alterations. *Cancer Discovery*, 2022. doi: doi:10.1158/2159-8290.CD-21-0832.

[281] Lin Zhang, Xing Chen, Na-Na Guan, Hui Liu, and Jian-Qiang Li. A hybrid interpolation weighted collaborative filtering method for anti-cancer drug response prediction. *Frontiers in pharmacology*, 9:1017, 2018. doi: https://doi.org/10.3389/fphar.2018.01017.

[282] Na-Na Guan, Yan Zhao, Chun-Chun Wang, Jian-Qiang Li, Xing Chen, and Xue Piao. Anticancer drug response prediction in cell lines using weighted graph regularized matrix factorization. *Molecular therapy-nucleic acids*, 17: 164–174, 2019. doi: https://doi.org/10.1016/j.omtn.2019.05.017.

[283] Mehreen Ali and Tero Aittokallio. Machine learning and feature selection for drug response prediction in precision oncology applications. *Biophysical reviews*, 11(1):31–39, 2019.

[284] Carlos De Niz, Raziur Rahman, Xiangyuan Zhao, and Ranadip Pal. Algorithms for drug sensitivity prediction. *Algorithms*, 9(4):77, 2016.

[285] Erik Štrumbelj, Zoran Bosnić, Igor Kononenko, Branko Zakotnik, and Cvetka Grašič Kuhar. Explanation and reliability of prediction models: the case of breast cancer recurrence. *Knowledge and information systems*, 24:305–324, 2010.

[286] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[287] A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 148–155, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 155860555X.

[288] Timothy G Whitsett, Ian T Mathews, Michael H Cardone, Ryan J Lena, William E Pierceall, Michael Bittner, Chao Sima, Janine LoBello, Glen J Weiss, and Nhan L Tran. Mcl-1 mediates tweak/fn14-induced non–small cell

lung cancer survival and therapeutic response. *Molecular Cancer Research*, 12 (4):550–559, 2014.

[289] Tanya Stoyanova, Nilotpal Roy, Dragana Kopanja, Srilata Bagchi, and Pradip Raychaudhuri. Ddb2 decides cell fate following dna damage. *Proceedings of the National Academy of Sciences*, 106(26):10690–10695, 2009.

[290] Shuqin Xing, Yafei Wang, Kaiwen Hu, Fen Wang, Tao Sun, and Quanwang Li. Wgcna reveals key gene modules regulated by the combined treatment of colon cancer with phy906 and cpt11. *Bioscience reports*, 40(9):BSR20200935, 2020.

[291] Florence Coussy, Rania El-Botty, Sophie Château-Joubert, Ahmed Dahmani, Elodie Montaudon, Sophie Leboucher, Ludivine Morisset, Pierre Painsec, Laura Sourd, Léa Huguet, et al. Brcaness, slfn11, and rb1 loss predict response to topoisomerase i inhibitors in triple-negative breast cancers. *Science Translational Medicine*, 12(531):eaax2625, 2020.

[292] Julia Krushkal, Yingdong Zhao, Curtis Hose, Anne Monks, James H Doroshow, and Richard Simon. Longitudinal transcriptional response of glycosylation-related genes, regulators, and targets in cancer cell lines treated with 11 antitumor agents. *Cancer Informatics*, 16:1176935117747259, 2017.

[293] Chiao-En Wu, Tsin Shue Koay, Arman Esfandiari, Yi-Hsuan Ho, Penny Lovat, and John Lunec. Atm dependent dusp6 modulation of p53 involved in synergistic targeting of mapk and p53 pathways with trametinib and mdm2 inhibitors in cutaneous melanoma. *Cancers*, 11(1):3, 2018.

[294] Karisa C Schreck, Amy N Allen, Jiawan Wang, and Christine A Pratilas. Combination mek and mtor inhibitor therapy is active in models of glioblastoma. *Neuro-Oncology Advances*, 2(1):vdaa138, 2020.

[295] Matteo Claudio Da Vià, Antonio Giovanni Solimando, Andoni Garitano-Trojaola, Santiago Barrio, Umair Munawar, Susanne Strifler, Larissa Haertle, Nadine Rhodes, Eva Teufel, Cornelia Vogt, et al. Cic mutation as a molecular mechanism of acquired resistance to combined braf-mek inhibition in extramedullary multiple myeloma with central nervous system involvement. *The oncologist*, 25(2):112–118, 2020.

[296] Keith T Flaherty. Moving forward: Making braf-targeted therapy better. *BRAF Targets in Melanoma: Biological Mechanisms, Resistance, and Drug Discovery*, pages 183–201, 2015.

[297] Johanna Eriksson, Vadim Le Joncour, Pirjo Nummela, Tiina Jahkola, Susanna Virolainen, Pirjo Laakkonen, Olli Saksela, and Erkki Hölttä. Gene expression analyses of primary melanomas reveal cthrc1 as an important player in melanoma progression. *Oncotarget*, 7(12):15065, 2016.

[298] JX Li, JM Feng, Y Wang, XH Li, XX Chen, Y Su, YY Shen, Y Chen, B Xiong, CH Yang, et al. The b-rafv600e inhibitor dabrafenib selectively inhibits rip3 and alleviates acetaminophen-induced liver injury. *Cell death & disease*, 5(6): e1278–e1278, 2014.

[299] Marilyn Safran, Irina Dalah, Justin Alexander, Naomi Rosen, Tsippi Iny Stein, Michael Shmoish, Noam Nativ, Iris Bahir, Tirza Doniger, Hagit Krug, et al. Genecards version 3: the human gene integrator. *Database*, 2010, 2010.

[300] Ilaria Iacobucci, Nunzio Iraci, Monica Messina, Annalisa Lonetti, Sabina Chiaretti, Emanuele Valli, Anna Ferrari, Cristina Papayannidis, Francesca Paoloni, Antonella Vitale, et al. Ikaros deletions dictate a unique gene expression signature in patients with adult b-cell acute lymphoblastic leukemia. *PloS one*, 7(7):e40934, 2012.

[301] Zhigang Tan, Jizong Zhao, and Yugang Jiang. Mir-634 sensitizes glioma cells to temozolomide by targeting cyr 61 through raf-erk signaling pathway. *Cancer Medicine*, 7(3):913–921, 2018.

[302] Xin Ge, Min-Hong Pan, Lin Wang, Wei Li, Chengfei Jiang, Jun He, Khaled Abouzid, Ling-Zhi Liu, Zhumei Shi, and Bing-Hua Jiang. Hypoxia-mediated mitochondria apoptosis inhibition induces temozolomide treatment resistance through mir-26a/bad/bax axis. *Cell death & disease*, 9(11):1128, 2018.

[303] Yosef Hochberg and Yoav Benjamini. More powerful procedures for multiple significance testing. *Statistics in medicine*, 9(7):811–818, 1990.

Appendix

A Publication list

- ✯ Schneider, L., Kehl, T., Thedinga, K., Grammes, N. L., Backes, C., Mohr, C., Schubert, B., **Lenhof, K.**, ... & Lenhof, H. P. (2019). ClinOmicsTrailbc: a visual analytics tool for breast cancer treatment stratification. Bioinformatics, 35(24), 5171-5181.

- ✯ Gerstner, N., Kehl, T., **Lenhof, K.**, Müller, A., Mayer, C., Eckhart, L., ... & Lenhof, H. P. (2020). GeneTrail 3: advanced high-throughput enrichment analysis. Nucleic Acids Research, 48(W1), W515-W520.

- ✯ Diener, C., Hart, M., Kehl, T., Rheinheimer, S., Ludwig, N., Krammes, L., Pawusch, S., **Lenhof, K.**, ... & Meese, E. (2020). Quantitative and time-resolved miRNA pattern of early human T cell activation. Nucleic Acids Research, 48(18), 10164-10183.

- ✯ Gerstner, N., Kehl, T., **Lenhof, K.,** Eckhart, L., Schneider, L., Stöckel, D., ... & Lenhof, H. P. (2021). GeneTrail: a framework for the analysis of high-throughput profiles. Frontiers in Molecular Biosciences, 8, 716544.

- ✯ **Lenhof, K.**, Gerstner, N., Kehl, T., Eckhart, L., Schneider, L., & Lenhof, H. P. (2021). MERIDA: a novel Boolean logic-based integer linear program for personalized cancer therapy. Bioinformatics, 37(21), 3881-3888.

- ✯ **Lenhof, K.**, Eckhart, L., Gerstner, N., Kehl, T., & Lenhof, H. P. (2022). Simultaneous regression and classification for drug sensitivity prediction using an advanced random forest method. Scientific Reports, 12(1), 13458.

- ✯ **Lenhof, K.**†, Eckhart, L.†, Rolli, L. M., Volkamer, A., & Lenhof, H. P. (2023). Reliable anti-cancer drug sensitivity prediction and prioritization. (Research-Square Preprint, currently under review in Scientific Reports)

- ✯ Eckhart, L., **Lenhof, K.**, Rolli, L.M., & Lenhof, H.P. (2023). A comprehensive benchmarking of machine learning algorithms and dimensionality reduction methods for drug sensitivity prediction. (under submission)

- ✯ **Lenhof, K.**, Eckhart, L.†, Rolli, L.M.†, & Lenhof, H. P. (2024). Trust me if you can: a survey on reliability and interpretability of machine learning approaches for drug sensitivity prediction in cancer. (under submission)

B MERIDA additional information

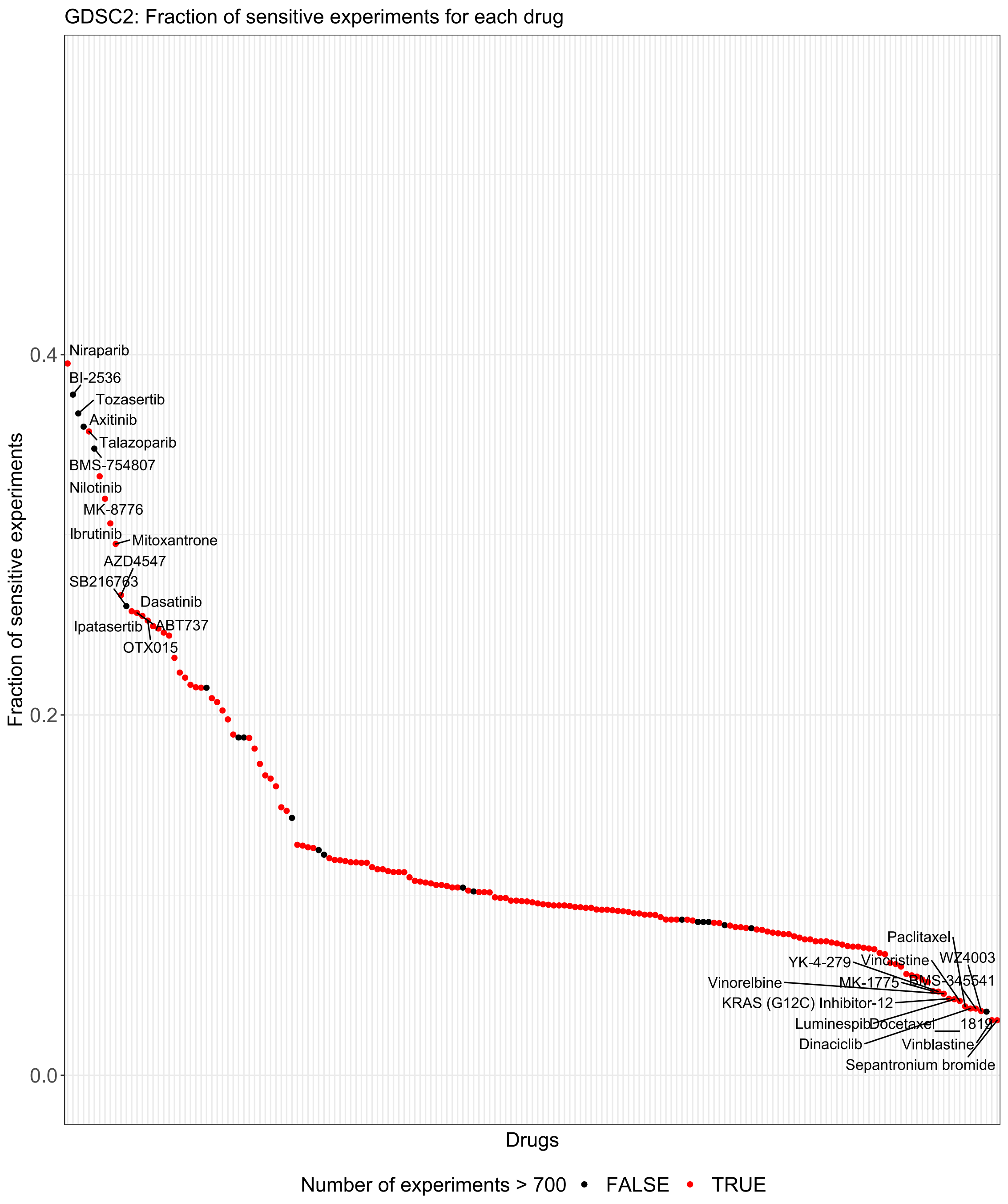


Figure B.1: Sensitive-to-resistant ratio in GDSC2. In this figure, we depict the fraction of sensitive cell lines for each drug from GDSC2 when binarizing the logarithmized IC50 values with the procedure suggested by Knijnenburg et al. [27]. The coloring indicates whether a drug has been screened with more than 700 cell lines (red) or not (black).

Table B.1: This table depicts all biomarkers used as a priori knowledge for Rapamycin.

Feature name	Association
PTEN truncating mutations	sensitive
PTEN p.R173H	sensitive
PTEN p.R130*	sensitive
PTEN p.R233*	sensitive
PTEN p.T167A	sensitive
PTEN p.Y68H	sensitive
PTEN p.D162H	sensitive
PTEN p.K128N	sensitive
PTEN p.C136R	sensitive
PTEN p.Y155C	sensitive
PTEN p.R130Q	sensitive
PTEN p.F341V	sensitive
PTEN p.Q298*	sensitive
PTEN p.R173C	sensitive
PTEN p.Y16*	sensitive
PTEN p.L42R	sensitive
PTEN p.W274*	sensitive
PTEN p.R159S	sensitive
PTEN p.D92H	sensitive
PTEN p.G36R	sensitive
PTEN p.R335*	sensitive
PTEN p.R173P	sensitive
PTEN p.C136Y	sensitive
PTEN p.K128T	sensitive
PTEN p.R130G	sensitive
PTEN p.E307*	sensitive
PTEN p.R47G	sensitive
PTEN p.Q245*	sensitive
PTEN CNV loss	sensitive
FBXW7 truncating mutations	sensitive
NF1 truncating mutations	sensitive
NF1 p.R304*	sensitive
NF1 p.R1204W	sensitive
NF1 p.Y2285*	sensitive
NF1 CNV loss	sensitive
FBXW7 loss-of-function	sensitive
TSC1 frameshift	sensitive
PIK3CA p.E542K	sensitive
STK11 CNV loss	sensitive
FBXW7 CNV loss	sensitive
MTOR p.C1483Y	sensitive

Table B.2: This table depicts all biomarkers used as a priori knowledge for Temsirolimus.

Feature name	Association
PTEN R130*	sensitive
PTEN CNV loss	sensitive
PIK3CA p.H1047R	sensitive
ERBB2 gain-of-function	sensitive
ERBB2 p.S310F	sensitive
ERBB2 p.D277H	sensitive
ERBB2 p.R678Q	sensitive
ERBB2 p.S335C	sensitive
ERBB2 p.L755S	sensitive
ERBB2 p.T798I	sensitive
ERBB2 p.V842I	sensitive
ERBB2 p.S653C	sensitive
ERBB2 CNV gain	sensitive
PIK3CA p.E542K	sensitive
AKT1 p.E17K	sensitive

Table B.3: This table depicts all biomarkers used as a priori knowledge for Dactolisib.

Feature name	Association
PIK3CA p.H1047R	sensitive

Table B.4: This table depicts all biomarkers used as a priori knowledge for Talazoparib.

Feature name	Association
BRCA1 loss-of-function mutations	sensitive
BRCA1 truncating mutations	sensitive
BRCA2 truncating mutations	sensitive

Table B.5: This table depicts all biomarkers used as a priori knowledge for CX-5461.

Feature name	Association
BRCA1 loss-of-function mutations	sensitive
BRCA1 truncating mutations	sensitive
BRCA2 truncating mutations	sensitive

Table B.6: This table depicts all biomarkers used as a priori knowledge for Apitolisib.

Feature name	Association
PIK3CA p.E545K	sensitive
PIK3CA p.E542K	sensitive

Table B.7: This table depicts all biomarkers used as a priori knowledge for Alpelisib.

Feature name	Association
PIK3CA p.H1047R	sensitive
PIK3CA CNV gain	sensitive
PTEN CNV loss	resistant

Table B.8: This table depicts all biomarkers used as a priori knowledge for AZD8186.

Feature name	Association
PTEN CNV loss	sensitive

Table B.9: This table depicts all biomarkers used as a priori knowledge for MK-2206.

Feature name	Association
KRAS p.G12D	sensitive
PIK3CA p.E545K	sensitive
PTEN CNV loss	sensitive
AKT1 p.E17K	not sensitive

Table B.10: This table depicts all biomarkers used as a priori knowledge for Pictilisib.

Feature name	Association
BRAF p.V600E	sensitive
PIK3CA p.E545K	sensitive
ERBB2 CNV gain	sensitive
PIK3CA CNV gain	sensitive

Table B.11: This table depicts all biomarkers used as a priori knowledge for Taselisib.

Feature name	Association
PIK3CA p.H1047R	sensitive
PIK3CA CNV gain	sensitive

Iterative feature selection

The results of our runtime analysis clearly show that MERIDA has a significantly reduced runtime compared to LOBICO. However, since in our real performance analysis, the number of input features is considerably higher (1500 features), the number of possible logic combinations becomes very large and the parameter M had to be chosen carefully to balance between the runtime of the analysis and the expressiveness of the Boolean formulas. Whereas MERIDA could still identify models with up to M = 10 features in an acceptable amount of time, LOBICO could mostly only handle 2 features. In order to simultaneously increase the parameter M and the number of input features for MERIDA without compromising the performance, we tested an iterative approach.

To this end, we used the data sets for OSI-027, PIK-93, and Voxtalisib. Then we randomly drew 400 features for each of the drugs and use these as input features for the analyses described in the following.

For each of the three drugs, we trained models with $M \in \{1, \ldots, 16\}$ in one-shot, i.e. without any iterative selection. In addition, we tested an iterative procedure: we first selected the best model among the models with $M = 1$ to $M = 6$ using Youden's J. Then, we iteratively increased M by 4 putative features and selected the best model using Youden's J until we could fit a model with $M = 16$. Figure B.2 shows the results of this analysis for the cubic weight function. It can be seen, that the overall runtime (including all necessary cross-validation steps) can be reduced by a factor of 25 on average. Moreover, the selected feature sets for the two approaches resemble each other significantly (Fisher's exact test $p\text{-value} < 0.05$). Based on these tests, we decided to perform the iterative analysis in the main text similarly: we initially fit models with M in the range of 1 to 6 and increase M by 4 putative features in each iteration.

OSI-027

M	No iteration	Iteration	P-value
2	Sensitive: SPOP_high_expr, TJP1_low_expr	Sensitive: SPOP_high_expr, TJP1_low_expr	—
3	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: CSNK2A1_high_expr	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: CSNK2A1_high_expr	1.25e-05
4	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: FAF1_low_expr, DIS3_low_expr	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: CSNK2A1_high_expr, PSIP1_low_expr	0.00044
5	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: FAF1_low_expr, DIS3_low_expr, PSIP1_low_expr	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr	9.37e-06
6	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: FAF1_low_expr, DIS3_low_expr, PSIP1_low_expr, HLF_high_expr	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr	4.31e-10
7	Sensitive: SPOP_high_expr, TJP1_low_expr, ARID1B_high_expr Resistant: FAF1_low_expr, DIS3_low_expr, HLF_high_expr, ATRX_low_expr	Sensitive: SPOP_high_expr, TJP1_low_expr Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr, PLXNB2_Unknown	1.44e-06
8	Sensitive: SPOP_high_expr, TJP1_low_expr, ARID1B_high_expr Resistant: FAF1_low_expr, DIS3_low_expr, HLF_high_expr, PSIP1_low_expr, HSP90AA1_Unknown	Sensitive: SPOP_high_expr, TJP1_low_expr, FMR1_Unknown Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr, PLXNB2_Unknown	3.69e-08
9	Sensitive: SPOP_high_expr, TJP1_low_expr, ARID1B_high_expr Resistant: FAF1_low_expr, DIS3_low_expr, HLF_high_expr, PSIP1_low_expr, HSP90AA1_Unknown, ERBB3_Unknown	Sensitive: SPOP_high_expr, TJP1_low_expr, FMR1_Unknown Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr, PLXNB2_Unknown, EP300_low_expr	1.84e-07
10	Sensitive: SPOP_high_expr, TJP1_low_expr, ARID1B_high_expr Resistant: FAF1_low_expr, HLF_high_expr, PSIP1_low_expr, HSP90AA1_Unknown, ERBB3_Unknown, CAD_high_expr, PLXNB2_Unknown	Sensitive: SPOP_high_expr, TJP1_low_expr, FMR1_Unknown Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr, PLXNB2_Unknown, EP300_low_expr, CDK4_high_expr	7.77e-09

M	No iteration	Iteration	P-value
12	Sensitive: SPOP_high_expr, TJP1_low_expr, ARID1B_high_expr Resistant: FAF1_low_expr, HLF_high_expr, PSIP1_low_expr, HSP90AA1_Unknown, ERBB3_Unknown, CAD_high_expr, PLXNB2_Unknown, HSP90AA1_low_expr, CDK4_high_expr	Sensitive: SPOP_high_expr, TJP1_low_expr, FMR1_Unknown Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr, PLXNB2_Unknown, EP300_low_expr, CDK4_high_expr, CAD_high_expr, HSP90AA1_Unknown	7.18e-14
14	Sensitive: SPOP_high_expr, TJP1_low_expr, ARID1B_high_expr, CHEK2_low_expr Resistant: FAF1_low_expr, HLF_high_expr, PSIP1_low_expr, HSP90AA1_Unknown, ERBB3_Unknown, CAD_high_expr, PLXNB2_Unknown, CDK4_high_expr, RTN4_high_expr, EZH2_low_expr	Sensitive: SPOP_high_expr, TJP1_low_expr, FMR1_Unknown, CHEK2_low_expr Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr, PLXNB2_Unknown, EP300_low_expr, CDK4_high_expr, CAD_high_expr, HSP90AA1_Unknown, RTN4_high_expr	1.41e-16
16	Sensitive: SPOP_high_expr, TJP1_low_expr, ARID1B_high_expr, CHEK2_low_expr, SRN1_high_expr Resistant: RTN4_high_expr, ARFGAP2_low_expr, HSP90AA1_Unknown, FAF1_low_expr, CAD_high_expr, PSIP1_low_expr, PLXNB2_Unknown, CTTN_high_expr, CDK4_high_expr, ACSL6_high_expr, ERBB3_Unknown	Sensitive: SPOP_high_expr, TJP1_low_expr, FMR1_Unknown, CHEK2_low_expr Resistant: CSNK2A1_high_expr, PSIP1_low_expr, HLF_high_expr, FAF1_low_expr, PLXNB2_Unknown, EP300_low_expr, CDK4_high_expr, CAD_high_expr, HSP90AA1_Unknown, RTN4_high_expr, HSP90AA1_low_expr, PPP2R1A_low_expr	2.28e-12
Total Time	33135 s	1144 s	

PIK-93

M	No iteration	Iteration	
2	sensitive: PTPRF_low_expr, CTTN_low_expr	sensitive: PTPRF_low_expr, CTTN_low_expr	—
5	sensitive: NTRK1_CNVgain, PPM1D_Unknown, NCKAP1_low_expr, MED23_high_expr, CTTN_low_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain	9.37e-06
8	sensitive: FRG1_high_expr, NTRK1_CNVgain, ELF1_high_expr, PPM1D_Unknown, NCKAP1_low_expr, CTTN_low_expr resistant: DICER1_low_expr, MMP2_high_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain, NCKAP1_low_expr resistant: DICER1_low_expr, MMP2_high_expr	1.41e-10
9	sensitive: FRG1_high_expr, NTRK1_CNVgain, ELF1_high_expr, PPM1D_Unknown, NCKAP1_low_expr, CTTN_low_expr resistant: DICER1_low_expr, MMP2_high_expr, PLXNA1_high_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain, NCKAP1_low_expr resistant: DICER1_low_expr, MMP2_high_expr, DLG1_low_expr	1.26e-09
10	sensitive: ING1_low_expr, NTRK1_CNVgain, PPM1D_Unknown, NCKAP1_low_expr, MED23_high_expr, CTTN_low_expr resistant: DICER1_low_expr, TBX3_high_expr, NTN4_high_expr, ATRX_low_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain, NCKAP1_low_expr resistant: DICER1_low_expr, MMP2_high_expr, DLG1_low_expr, IRF6_high_expr	7.23e-07
11	sensitive: ING1_low_expr, NTRK1_CNVgain, PPM1D_Unknown, NCKAP1_low_expr, MED23_high_expr, CTTN_low_expr resistant: ANK3_high_expr, DICER1_low_expr, TBX3_high_expr, NTN4_high_expr, ATRX_low_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain, NCKAP1_low_expr resistant: DICER1_low_expr, MMP2_high_expr, DLG1_low_expr, IRF6_high_expr, MYD88_Unknown	2.37e-06
12	sensitive: HDAC3_low_expr, NCKAP1_low_expr, MED23_high_expr, CTTN_low_expr, ASXL1_Unknown resistant: TBX3_high_expr, ANK3_high_expr, MED12_low_expr, CAD_low_expr, IRF6_high_expr, FXR1_low_expr, ERBB3_high_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain, NCKAP1_low_expr resistant: DICER1_low_expr, MMP2_high_expr, DLG1_low_expr, IRF6_high_expr, MYD88_Unknown, ATRX_low_expr	0.0039

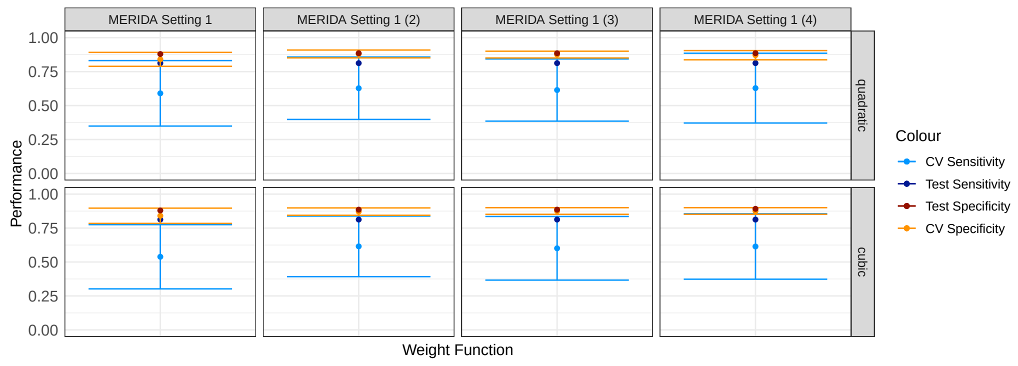
M	No iteration	Iteration	
13	sensitive: HDAC3_low_expr, NCKAP1_low_expr, MED23_high_expr, CTTN_low_expr, ASXL1_Unknown, MNDA_high_expr resistant: TBX3_high_expr, ANK3_high_expr, MED12_low_expr, CAD_low_expr, IRF6_high_expr, FXR1_low_expr, ERBB3_high_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain, NCKAP1_low_expr resistant: DICER1_low_expr, MMP2_high_expr, DLG1_low_expr, IRF6_high_expr, MYD88_Unknown, ATRX_low_expr, NRAS_high_expr	0.0063
16	sensitive: ATR_low_expr, ROBO2_high_expr, FRG1_high_expr, NTRK1_CNVgain, FAT1_low_expr, CTTN_low_expr resistant: AMER1_low_expr, PCDH18_high_expr, DLG1_low_expr, IRF6_high_expr, TBX3_high_expr, FN1_high_expr, MYH10_Unknown, ACACA_low_expr, ATRX_low_expr	sensitive: PTPRF_low_expr, CTTN_low_expr, CCND1_high_expr, PPM1D_Unknown, NTRK1_CNVgain, FAT1_low_expr, NCKAP1_low_expr resistant: DICER1_low_expr, MMP2_high_expr, DLG1_low_expr, IRF6_high_expr, MYD88_Unknown, ATRX_low_expr, ELF1_low_expr, ACACA_low_expr, PCDH18_high_expr	9.43e-09
Total Time	35446 s	1318 s	

Voxtalisib

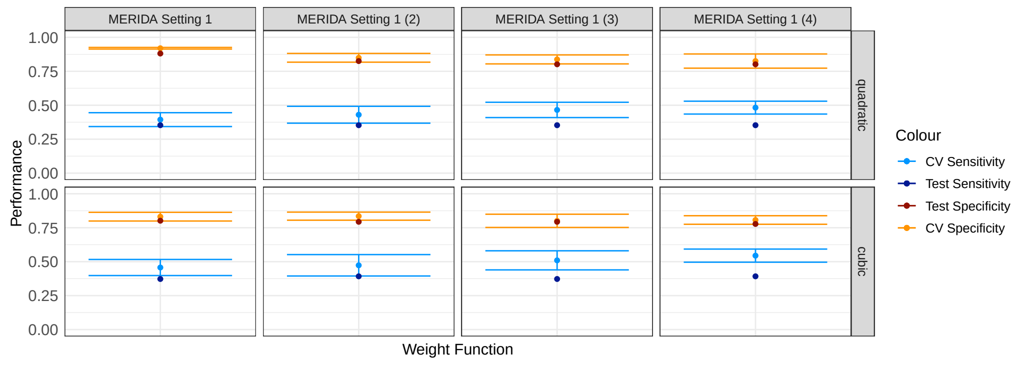
M	No iteration	Iteration	P-value
4	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr	—
8	sensitive: MYC_Unknown, AFF4_low_expr, CREBBP_Loss-of-function, BCL11A_high_expr, SCAI_high_expr resistant: SVEP1_Unknown, PTPRU_high_expr, CHD4_Unknown	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr, STK11_Unknown, FAT2_CNVloss, PRKAR1A_high_expr resistant: AKAP9_Unknown	0.00028
9	sensitive: MYC_Unknown, SPOP_low_expr, CREBBP_Loss-of-function, FAT2_CNVloss, BCL11A_high_expr, SCAI_high_expr resistant: SVEP1_Unknown, AKAP9_Unknown, ANK3_Unknown	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr, STK11_Unknown, FAT2_CNVloss, PRKAR1A_high_expr resistant: AKAP9_Unknown, STK11_low_expr	1.84e-07
10	sensitive: MYC_Unknown, MYH10_high_expr, FAT2_CNVloss, STK11_Unknown, BCL11A_high_expr, SCAI_high_expr, PRKAR1A_high_expr resistant: CHD1L_Unknown, AKAP9_Unknown, STK11_low_expr	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr, STK11_Unknown, FAT2_CNVloss, PRKAR1A_high_expr resistant: AKAP9_Unknown, STK11_low_expr, CHD1L_Unknown	3.8e-20
12	sensitive: MYC_Unknown, MYH10_high_expr, FAT2_CNVloss, STK11_Unknown, BCL11A_high_expr, SCAI_high_expr, PRKAR1A_high_expr resistant: RFC4_high_expr, CHD1L_Unknown, RPL22_low_expr AKAP9_Unknown, STK11_low_expr	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr, STK11_Unknown, FAT2_CNVloss, PRKAR1A_high_expr resistant: AKAP9_Unknown, STK11_low_expr, CHD1L_Unknown, RFC4_high_expr, RPL22_low_expr	3.37e-23
13	sensitive: MYC_Unknown, CREBBP_Loss-of-function, FAT2_CNVloss, MTOR_Neural, STK11_Unknown, BCL11A_high_expr, SCAI_high_expr, PIK3C2B_low_expr resistant: F8_high_expr, AHCTF1_low_expr, MYB_Unknown, RB1_Loss-of-function, CHD4_Unknown	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr, STK11_Unknown, FAT2_CNVloss, PRKAR1A_high_expr, CDK12_Unknown resistant: AKAP9_Unknown, STK11_low_expr, CHD1L_Unknown, RFC4_high_expr, RPL22_low_expr	1.73e-05

M	No iteration	Iteration	P-value
16	sensitive: MYC_Unknown, CREBBP_Loss-of-function, FAT2_CNVloss, MTOR_Neutral, STK11_Unknown, BCL11A_high_expr, SCAI_high_expr, PIK3C2B_low_expr, SF3B1_low_expr resistant: F8_high_expr, DIS3_low_expr, AHCTF1_low_expr, MYB_Unknown, RB1_Loss-of-functiion, DICER1_high_expr, CHD4_Unknown	sensitive: MYC_Unknown, MYH10_high_expr, BCL11A_high_expr, SCAI_high_expr, STK11_Unknown, FAT2_CNVloss, PRKAR1A_high_expr, SF3B1_low_expr resistant: EEF1A1_low_expr, DIS3_low_expr, AKAP9_Unknown, STK11_low_expr, CHD1L_Unknown, RFC4_high_expr, RPL22_low_expr, CHD4_Unknown	9.43e-09
Total Time	10893 s	520 s	

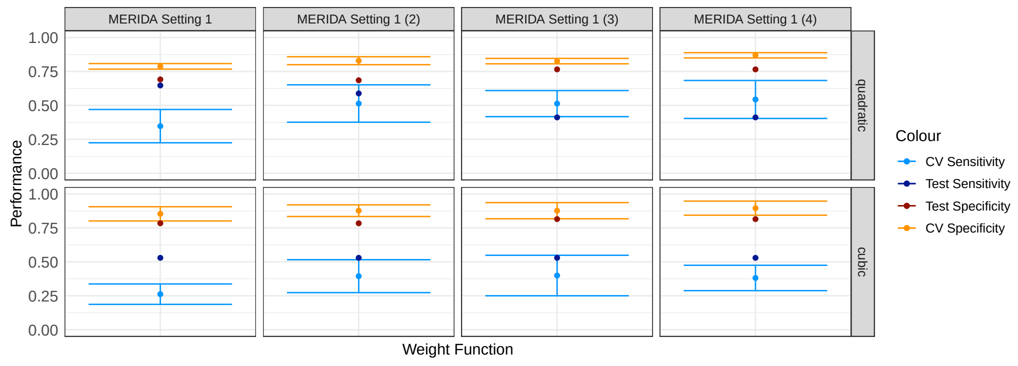
Figure B.2: In this figure, we present a comparison between the selected features of a one-shot approach and an iterative selection for OSI-027, PIK-93, and Voxtalisib using the cubic weight function. We show the feature sets for the best M value in each iteration compared to the model of the same size in the one-shot approach. Common features are marked in magenta, blue features only occur in one of the approaches and black features appear in both approaches but not for the given value of M. We start with the best model among $M = 1$ to $M = 6$ and then iteratively add 4 putative features until we reach a possible model size of $M = 16$. For $M = 16$, we simply show the first model of this size, which is reached in the last iteration. This model does not correspond to the best model of the last iteration. The given p-value is the Fisher's exact test p-value for the intersection of the feature sets. The total time refers to the time needed to fit all models including all M values and CV steps.



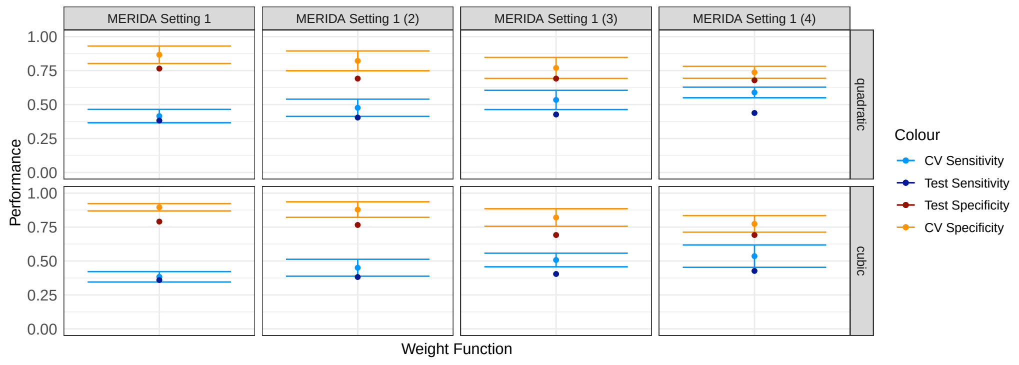
(a) **AZD8055**



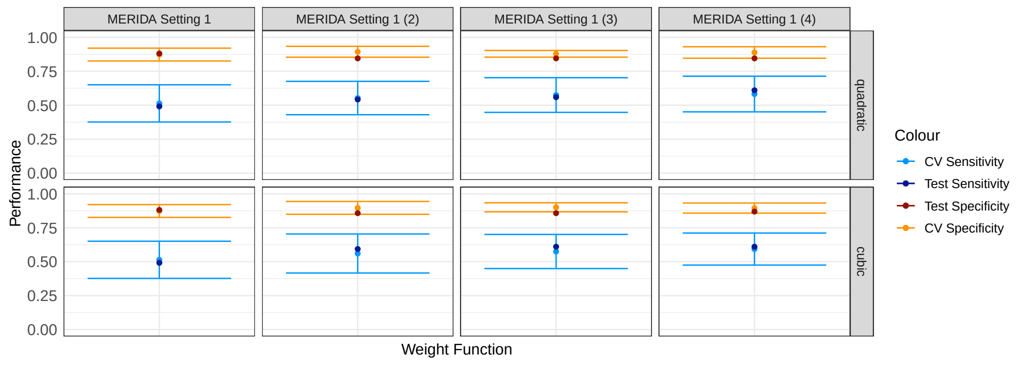
(b) **Omipalisib**



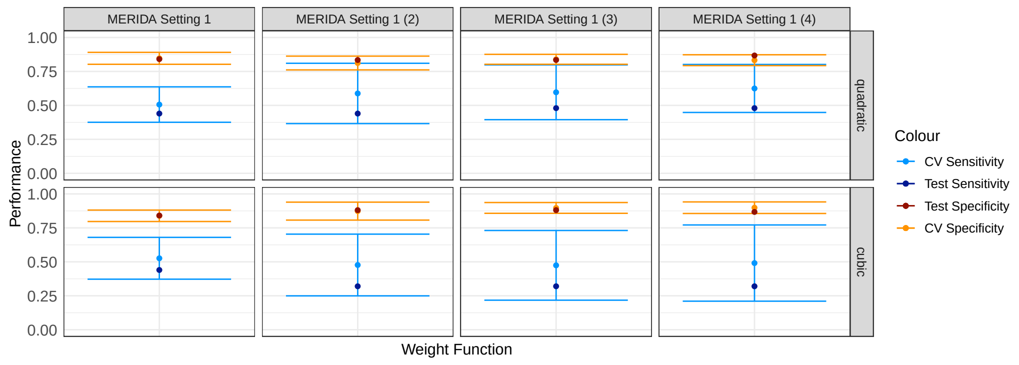
(c) **Voxtalisib**



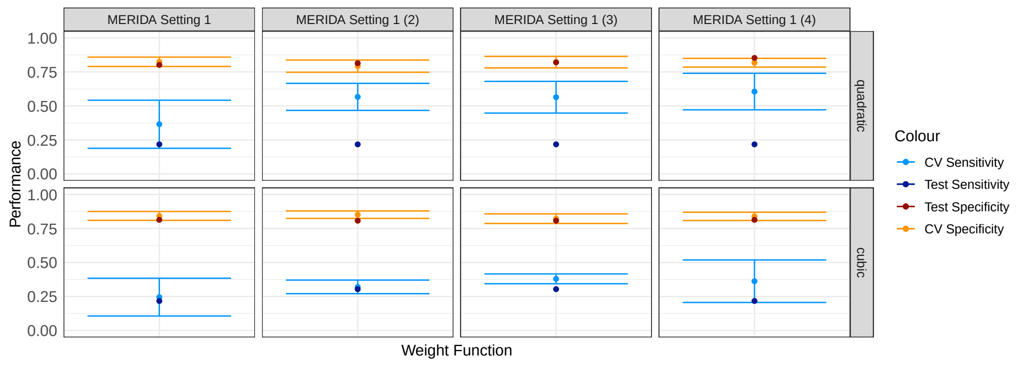
(d) **NSC319726**



(e) **Niraparib**



(f) **AS605240**



(g) Pilaralisib

Figure B.3: In Sub-Figure (a - g), we present the statistical performance for drugs without a priori knowledge during CV as well as on a test set when iteratively using the features from previous applications of our method as a priori knowledge for a new run.

Runtime analysis additional results

We tested MERIDA and LOBICO with different hyperparameters (see Table B.12), different numbers of input features, and the three different weight functions. Note that for LOBICO we could not fit larger models with both $K > 1$ and $M > 1$ in a reasonable amount of time. We performed the computations on a compute server with four Intel(R) Xeon(R) CPU E5-4657L v2 processors with 2.40GHz clock rate. IBM ILOG CPLEX Optimization Studio V12.6.2 for C++ was employed for formulating and solving the ILP. CPLEX was run using 32 cores and a deterministic parallel mode.

We could observe that using MERIDA instead of LOBICO leads to a considerably reduction in runtime (cf. Table B.13). Moreover, considering the quadratic or cubic weight function instead of the linear one does also reduce runtime significantly (cf. Table B.14). For LOBICO, we note that the two parameters K and M seem to influence the runtime differently, i.e. according to our analyses for different K and M (see Figure B.4), it can be seen that the model parameter K increases the runtime more strongly than the model parameter M.

Table B.12: Parameters

MERIDA parameters	LOBICO parameters
$M \in \{2, 4, 8\}$	$(K, M) \in \{(1,1), (2,1), (1,2), (2,2), (1,4), (4,1)\}$

Table B.13: Runtime comparison of a four-feature model from LOBICO versus a four-feature model from MERIDA. The values depict the factor by which MERIDA is faster than LOBICO using its original linear weight function. To obtain a four-feature model with LOBICO, three different parameter settings can be used ($(K, M) \in \{(1,4), (4,1), (2,2)\}$).

	MERIDA linear	MERIDA quadratic	MERIDA cubic
$(K, M) = (1, 4)$	3.61	4.85	6
$(K, M) = (4, 1)$	71.79	98.54	125.31
$(K, M) = (2, 2)$	641.97	876.74	1147.6
Mean of the above three rows	239.13	326.71	426.31

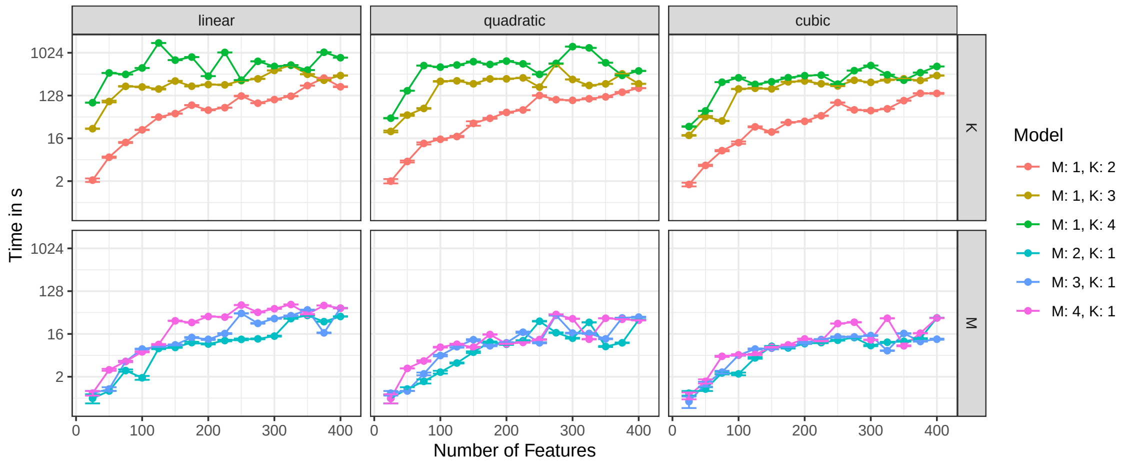


Figure B.4: In this figure, we present the runtime of LOBICO with varying hyperparameters M and K for the three different weight functions.

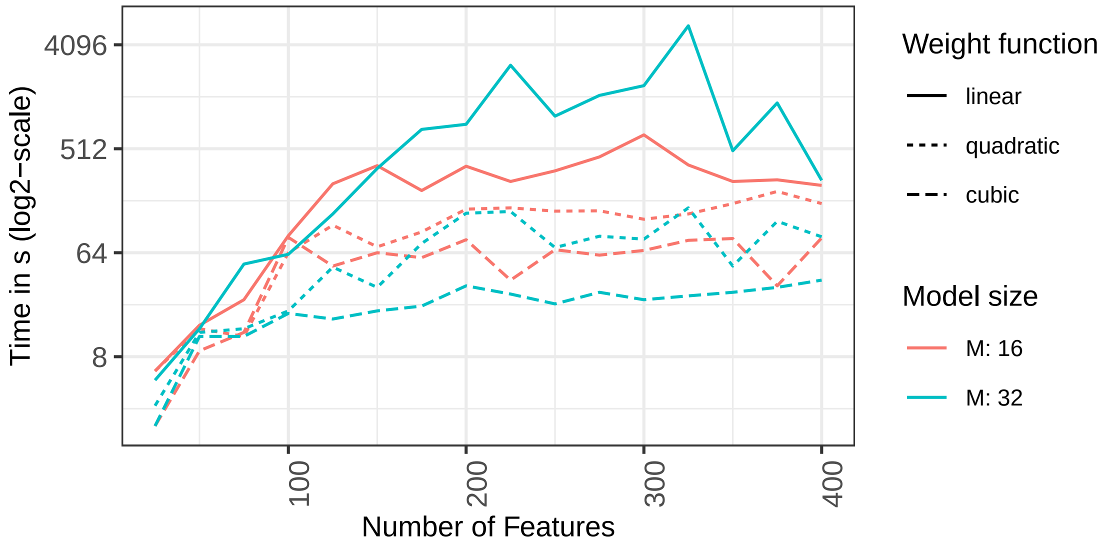


Figure B.5: In this figure, we present the runtime of MERIDA for $M = 16$ and $M = 32$ for the different weight functions.

Table B.14: The table gives the runtime advantage of using a quadratic or cubic weight function instead of using a linear weight function. The values depict the factor by which a model with a quadratic or cubic weight function is faster than a model build with the linear weight function. The given factor is calculated by averaging over all tested numbers of features. The number in brackets are the model parameters. For LOBICO, the first number is the value for K and the second number is the value for M.

	MERIDA (4)	LOBICO $(1,4)$	LOBICO $(4,1)$	LOBICO $(2,2)$
quadratic	1.35	2.25	1.38	2.88
cubic	1.65	2.49	2.55	5.38

Table B.15: This table gives the mean and median runtime of MERIDA across all drugs for the different settings, weight functions, and parameter combinations in the pan-cancer analysis. The mean (median) is calculated by summing up the runtime of the cross-validation and the final fit for one specific drug and then averaging across all drugs.

Setting	Weight function	Model size	Mean runtime (in s)	Median runtime (in s)
Setting 1	cubic	1	5.3170	6
Setting 1	cubic	2	15.7073	10
Setting 1	cubic	3	109.6829	96
Setting 1	cubic	4	314.45	265
Setting 1	cubic	5	1128.3076	963
Setting 1	cubic	6	4970.3421	3589.5
Setting 1	quadratic	1	4.6097	6
Setting 1	quadratic	2	17.8780	11
Setting 1	quadratic	3	114.9024	107
Setting 1	quadratic	4	315.6666	241
Setting 1	quadratic	5	1583.2631	1051
Setting 1	quadratic	6	7657.0277	4104
Setting 1	linear	1	4.1463	5
Setting 1	linear	2	26.2195	15
Setting 1	linear	3	128.4390	120
Setting 1	linear	4	409	215
Setting 1	linear	5	2219.1842	1434.5
Setting 1	linear	6	9679.0285	7176
Setting 2	cubic	1	0	0
Setting 2	cubic	2	3.9090	5
Setting 2	cubic	3	9.9090	9
Setting 2	cubic	4	68.6363	62
Setting 2	cubic	5	137.3636	147
Setting 2	cubic	6	428.5454	322
Setting 2	quadratic	1	0	0
Setting 2	quadratic	2	2.3636	1
Setting 2	quadratic	3	11.3636	10
Setting 2	quadratic	4	79.3636	77
Setting 2	quadratic	5	167.7272	174

Setting 2	quadratic	6	695.3636	438
Setting 2	linear	1	0	0
Setting 2	linear	2	2.3636	1
Setting 2	linear	3	24.6363	16
Setting 2	linear	4	102.3636	93
Setting 2	linear	5	376.1818	220
Setting 2	linear	6	1854.9090	1168
Setting 3	cubic	1	3.8181	3
Setting 3	cubic	2	12.9090	12
Setting 3	cubic	3	105.3636	109
Setting 3	cubic	4	265.1818	230
Setting 3	cubic	5	1100.8181	701
Setting 3	cubic	6	4942.1818	2544
Setting 3	quadratic	1	4.8181	6
Setting 3	quadratic	2	15.9090	12
Setting 3	quadratic	3	107	124
Setting 3	quadratic	4	273.5454	221
Setting 3	quadratic	5	1600.6363	1381
Setting 3	quadratic	6	9044.1818	6007
Setting 3	linear	1	3.7272	4
Setting 3	linear	2	28.7272	16
Setting 3	linear	3	114	99
Setting 3	linear	4	452	179
Setting 3	linear	5	2950.4545	1332
Setting 3	linear	6	8749	7584

Table B.16: This table gives the mean and median runtime of MERIDA across all drugs for the different settings, weight functions, and parameter combinations in the non-haematological cancer cell line analysis. The mean is calculated by summing up the runtime of the cross-validation and the final fit for one specific drug and then averaging across all drugs.

Setting	Weight function	Model size	Mean runtime (in s)	Median runtime (in s)
Setting 1	cubic	1	1.2926	1
Setting 1	cubic	2	12.9024	10
Setting 1	cubic	3	88.7317	62

Setting 1	cubic	4	233.9268	172
Setting 1	cubic	5	774.1707	476
Setting 1	cubic	6	2433.5609	1759
Setting 1	quadratic	1	1.5853	1
Setting 1	quadratic	2	14.4146	10
Setting 1	quadratic	3	70.1463	62
Setting 1	quadratic	4	188.7560	151
Setting 1	quadratic	5	622.6097	445
Setting 1	quadratic	6	2753.7560	1930
Setting 1	linear	1	2.0487	1
Setting 1	linear	2	23.9024	12
Setting 1	linear	3	67.4634	62
Setting 1	linear	4	177.4146	127
Setting 1	linear	5	700.3170	480
Setting 1	linear	6	5473.6829	2277
Setting 2	cubic	1	0	0
Setting 2	cubic	2	1.4545	1
Setting 2	cubic	3	7.0909	7
Setting 2	cubic	4	40.8181	32
Setting 2	cubic	5	90.1818	79
Setting 2	cubic	6	195	145
Setting 2	quadratic	1	0	0
Setting 2	quadratic	2	0.54545	0
Setting 2	quadratic	3	11.0909	9
Setting 2	quadratic	4	54.5454	59
Setting 2	quadratic	5	110.1818	100
Setting 2	quadratic	6	261.1818	133
Setting 2	linear	1	0	0
Setting 2	linear	2	0.7272	0
Setting 2	linear	3	19.4545	10
Setting 2	linear	4	61.1818	54
Setting 2	linear	5	150.3636	101
Setting 2	linear	6	742.8181	232
Setting 3	cubic	1	0.7272	0

Setting 3	cubic	2	10.9090	9
Setting 3	cubic	3	66.2727	68
Setting 3	cubic	4	179.7272	192
Setting 3	cubic	5	498.9090	435
Setting 3	cubic	6	1724.1818	1539
Setting 3	quadratic	1	1.0909	0
Setting 3	quadratic	2	13.2727	8
Setting 3	quadratic	3	65.9090	63
Setting 3	quadratic	4	166.5454	168
Setting 3	quadratic	5	581.3636	458
Setting 3	quadratic	6	2644.7272	2203
Setting 3	linear	1	1	0
Setting 3	linear	2	23.9090	12
Setting 3	linear	3	67	55
Setting 3	linear	4	225.4545	130
Setting 3	linear	5	900.3636	553
Setting 3	linear	6	9887.0909	2176

Table B.17: This table gives the mean and median runtime of MERIDA across all drugs for the different settings, weight functions, and parameter combinations in the haematological cancer cell line analysis. The mean is calculated by summing up the runtime of the cross-validation and the final fit for one specific drug and then averaging across all drugs.

Setting	Weight function	Model size	Mean runtime (in s)	Median runtime (in s)
Setting 1	cubic	1	0	0
Setting 1	cubic	2	0.0243	0
Setting 1	cubic	3	2.1707	1
Setting 1	cubic	4	10.5853	11
Setting 1	cubic	5	20.4878	23
Setting 1	cubic	6	29.3170	31
Setting 1	quadratic	1	0	0
Setting 1	quadratic	2	0.0487	0
Setting 1	quadratic	3	2.9756	2
Setting 1	quadratic	4	11.7804	12
Setting 1	quadratic	5	20.4878	22

Setting 1	quadratic	6	27.3414	27
Setting 1	linear	1	0	0
Setting 1	linear	2	0.2195	0
Setting 1	linear	3	4.7073	4
Setting 1	linear	4	14.3414	16
Setting 1	linear	5	19.9756	21
Setting 1	linear	6	26.5365	27
Setting 2	cubic	1	0	0
Setting 2	cubic	2	0	0
Setting 2	cubic	3	0	0
Setting 2	cubic	4	1.3636	1
Setting 2	cubic	5	8.4545	10
Setting 2	cubic	6	18	21
Setting 2	quadratic	1	0	0
Setting 2	quadratic	2	0	0
Setting 2	quadratic	3	0	0
Setting 2	quadratic	4	1.5454	1
Setting 2	quadratic	5	10	11
Setting 2	quadratic	6	16.0909	20
Setting 2	linear	1	0	0
Setting 2	linear	2	0	0
Setting 2	linear	3	0.0909	0
Setting 2	linear	4	2.2727	2
Setting 2	linear	5	9.4545	9
Setting 2	linear	6	18.6363	22
Setting 3	cubic	1	0	0
Setting 3	cubic	2	0.0909	0
Setting 3	cubic	3	1.3636	1
Setting 3	cubic	4	9.3636	12
Setting 3	cubic	5	19.8181	21
Setting 3	cubic	6	30.4545	35
Setting 3	quadratic	1	0	0
Setting 3	quadratic	2	0	0
Setting 3	quadratic	3	1.9090	1

Setting 3	quadratic	4	11.6363	13
Setting 3	quadratic	5	16.8181	18
Setting 3	quadratic	6	25.2727	28
Setting 3	linear	1	0	0
Setting 3	linear	2	0	0
Setting 3	linear	3	2.9090	2
Setting 3	linear	4	11.2727	13
Setting 3	linear	5	18.0909	19
Setting 3	linear	6	24.3636	25

Table B.18: This table provides the runtime (CV + final fit) of LOBICO for each of the 10 analyzed drugs. In total, LOBICO needed approximately 4.5 days (total runtime) for fitting the shown models. For the calculation of the total runtime, we excluded all models where either a single fit in the cross-validation or the final fitting procedure was higher than 6 hours. Note that these models were also not completely fit. Additionally, we did not fit models for specific parameter combinations for the quadratic and linear weight function if the runtime for the cubic weight function already exceeded this threshold. Similarly, we did not fit a model for the linear weight function if the quadratic weight function exceeded this threshold.

Drug	Setting	Weight function	Model size	Runtime (in s)
AZD8055	Setting 1	cubic	1, 1	27
AZD8055	Setting 1	cubic	1, 2	1036
AZD8055	Setting 1	cubic	2, 1	137
AZD8055	Setting 1	cubic	2, 2	42971
AZD8055	Setting 1	quadratic	1, 1	29
AZD8055	Setting 1	quadratic	1, 2	3251
AZD8055	Setting 1	quadratic	2, 1	143
AZD8055	Setting 1	quadratic	2,2	$> 21\,600$
AZD8055	Setting 1	linear	1, 1	28
AZD8055	Setting 1	linear	1, 2	1916
AZD8055	Setting 1	linear	2, 1	136
AZD8055	Setting 1	linear	2,2	-
Omipalisib	Setting 1	cubic	1, 1	27
Omipalisib	Setting 1	cubic	1, 2	1412
Omipalisib	Setting 1	cubic	2, 1	135
Omipalisib	Setting 1	cubic	2, 2	$> 21\,600$

Omipalisib	Setting 1	quadratic	1, 1	28
Omipalisib	Setting 1	quadratic	1, 2	2507
Omipalisib	Setting 1	quadratic	2, 1	137
Omipalisib	Setting 1	quadratic	2,2	-
Omipalisib	Setting 1	linear	1, 1	31
Omipalisib	Setting 1	linear	1, 2	7317
Omipalisib	Setting 1	linear	2, 1	9530
Omipalisib	Setting 1	linear	2, 2	-
Voxtalisib	Setting 1	cubic	1, 1	28
Voxtalisib	Setting 1	cubic	1, 2	421
Voxtalisib	Setting 1	cubic	2, 1	145
Voxtalisib	Setting 1	cubic	2, 2	35849
Voxtalisib	Setting 1	quadratic	1, 1	26
Voxtalisib	Setting 1	quadratic	1, 2	1887
Voxtalisib	Setting 1	quadratic	2, 1	920
Voxtalisib	Setting 1	quadratic	2,2	$>$ 21 600
Voxtalisib	Setting 1	linear	1, 1	29
Voxtalisib	Setting 1	linear	1, 2	2916
Voxtalisib	Setting 1	linear	2, 1	10353
Voxtalisib	Setting 1	linear	2, 2	-
NSC319726	Setting 1	cubic	1, 1	36
NSC319726	Setting 1	cubic	1, 2	8696
NSC319726	Setting 1	cubic	2, 1	3121
NSC319726	Setting 1	cubic	2, 2	$>$ 21 600
NSC319726	Setting 1	quadratic	1, 1	26
NSC319726	Setting 1	quadratic	1, 2	6859
NSC319726	Setting 1	quadratic	2, 1	8616
NSC319726	Setting 1	quadratic	2, 2	-
NSC319726	Setting 1	linear	1, 1	25
NSC319726	Setting 1	linear	1, 2	49149
NSC319726	Setting 1	linear	2, 1	16323
NSC319726	Setting 1	linear	2,2	-
Niraparib	Setting 1	cubic	1, 1	24
Niraparib	Setting 1	cubic	1, 2	708

Niraparib	Setting 1	cubic	2, 1	103
Niraparib	Setting 1	cubic	2, 2	> 21 600
Niraparib	Setting 1	quadratic	1, 1	26
Niraparib	Setting 1	quadratic	1, 2	1428
Niraparib	Setting 1	quadratic	2, 1	102
Niraparib	Setting 1	quadratic	2,2	-
Niraparib	Setting 1	linear	1, 1	23
Niraparib	Setting 1	linear	1, 2	2499
Niraparib	Setting 1	linear	2, 1	168
Niraparib	Setting 1	linear	2, 2	-
Temsirolimus	Setting 3	cubic	1, 1	34
Temsirolimus	Setting 3	cubic	1, 2	2545
Temsirolimus	Setting 3	cubic	2, 1	176
Temsirolimus	Setting 3	cubic	2,2	> 21 600
Temsirolimus	Setting 3	quadratic	1, 1	26
Temsirolimus	Setting 3	quadratic	1, 2	3338
Temsirolimus	Setting 3	quadratic	2, 1	2901
Temsirolimus	Setting 3	quadratic	2, 2	-
Temsirolimus	Setting 3	linear	1, 1	27
Temsirolimus	Setting 3	linear	1, 2	5280
Temsirolimus	Setting 3	linear	2, 1	15451
Temsirolimus	Setting 3	linear	2, 2	-
CX-5461	Setting 3	cubic	1, 1	26
CX-5461	Setting 3	cubic	1, 2	2212
CX-5461	Setting 3	cubic	2, 1	515
CX-5461	Setting 3	cubic	2, 2	> 21 600
CX-5461	Setting 3	quadratic	1, 1	31
CX-5461	Setting 3	quadratic	1, 2	7170
CX-5461	Setting 3	quadratic	2, 1	5794
CX-5461	Setting 3	quadratic	2, 2	-
CX-5461	Setting 3	linear	1, 1	28
CX-5461	Setting 3	linear	1, 2	31725
CX-5461	Setting 3	linear	2, 1	21984
CX-5461	Setting 3	linear	2, 2	-

Rapamycin	Setting 3	cubic	1, 1	25
Rapamycin	Setting 3	cubic	1, 2	44
Rapamycin	Setting 3	cubic	2, 1	115
Rapamycin	Setting 3	cubic	2, 2	9440
Rapamycin	Setting 3	quadratic	1, 1	23
Rapamycin	Setting 3	quadratic	1, 2	337
Rapamycin	Setting 3	quadratic	2, 1	111
Rapamycin	Setting 3	quadratic	2, 2	20823
Rapamycin	Setting 3	linear	1, 1	30
Rapamycin	Setting 3	linear	1, 2	1927
Rapamycin	Setting 3	linear	2, 1	110
Rapamycin	Setting 3	linear	2, 2	$> 21\,600$
Dactolisib	Setting 3	cubic	1, 1	24
Dactolisib	Setting 3	cubic	1, 2	217
Dactolisib	Setting 3	cubic	2, 1	107
Dactolisib	Setting 3	cubic	2, 2	11968
Dactolisib	Setting 3	quadratic	1, 1	25
Dactolisib	Setting 3	quadratic	1, 2	1181
Dactolisib	Setting 3	quadratic	2, 1	110
Dactolisib	Setting 3	quadratic	2, 2	15602
Dactolisib	Setting 3	linear	1, 1	24
Dactolisib	Setting 3	linear	1, 2	1900
Dactolisib	Setting 3	linear	2, 1	119
Dactolisib	Setting 3	linear	2, 2	$> 21\,600$
Talazoparib	Setting 3	cubic	1, 1	24
Talazoparib	Setting 3	cubic	1, 2	1078
Talazoparib	Setting 3	cubic	2, 1	131
Talazoparib	Setting 3	cubic	2,2	$> 21\,600$
Talazoparib	Setting 3	quadratic	1, 1	23
Talazoparib	Setting 3	quadratic	1, 2	956
Talazoparib	Setting 3	quadratic	2, 1	125
Talazoparib	Setting 3	quadratic	2,2	-
Talazoparib	Setting 3	linear	1, 1	23
Talazoparib	Setting 3	linear	1, 2	1284

Talazoparib	Setting 3	linear	2, 1	5258
Talzoparib	Setting 3	linear	2, 2	-

Table B.19: This table provides the runtime (CV + final fit) of MERIDA for each of the 10 drugs analyzed in the comparison between LOBICO and MERIDA. In total, MERIDA needed approximately 3 days for fitting the shown models. No models were excluded for the calculation of the total runtime.

Drug	Setting	Weight function	Model size	Runtime (in s)
AZD8055	Setting 1	cubic	1	7
AZD8055	Setting 1	cubic	2	10
AZD8055	Setting 1	cubic	3	137
AZD8055	Setting 1	cubic	4	327
AZD8055	Setting 1	cubic	5	1234
AZD8055	Setting 1	cubic	6	4742
AZD8055	Setting 1	quadratic	1	6
AZD8055	Setting 1	quadratic	2	9
AZD8055	Setting 1	quadratic	3	115
AZD8055	Setting 1	quadratic	4	256
AZD8055	Setting 1	quadratic	5	1043
AZD8055	Setting 1	quadratic	6	4188
AZD8055	Setting 1	linear	1	6
AZD8055	Setting 1	linear	2	22
AZD8055	Setting 1	linear	3	143
AZD8055	Setting 1	linear	4	260
AZD8055	Setting 1	linear	5	1581
AZD8055	Setting 1	linear	6	8986
Omipalisib	Setting 1	cubic	1	6
Omipalisib	Setting 1	cubic	2	14
Omipalisib	Setting 1	cubic	3	53
Omipalisib	Setting 1	cubic	4	203
Omipalisib	Setting 1	cubic	5	317
Omipalisib	Setting 1	cubic	6	1228
Omipalisib	Setting 1	quadratic	1	5
Omipalisib	Setting 1	quadratic	2	19

Omipalisib	Setting 1	quadratic	3	133
Omipalisib	Setting 1	quadratic	4	212
Omipalisib	Setting 1	quadratic	5	728
Omipalisib	Setting 1	quadratic	6	2838
Omipalisib	Setting 1	linear	1	4
Omipalisib	Setting 1	linear	2	57
Omipalisib	Setting 1	linear	3	217
Omipalisib	Setting 1	linear	4	1221
Omipalisib	Setting 1	linear	5	7524
Omipalisib	Setting 1	linear	6	39933
Voxtalisib	Setting 1	cubic	1	6
Voxtalisib	Setting 1	cubic	2	8
Voxtalisib	Setting 1	cubic	3	99
Voxtalisib	Setting 1	cubic	4	219
Voxtalisib	Setting 1	cubic	5	1160
Voxtalisib	Setting 1	cubic	6	6790
Voxtalisib	Setting 1	quadratic	1	6
Voxtalisib	Setting 1	quadratic	2	22
Voxtalisib	Setting 1	quadratic	3	117
Voxtalisib	Setting 1	quadratic	4	241
Voxtalisib	Setting 1	quadratic	5	818
Voxtalisib	Setting 1	quadratic	6	5589
Voxtalisib	Setting 1	linear	1	7
Voxtalisib	Setting 1	linear	2	41
Voxtalisib	Setting 1	linear	3	119
Voxtalisib	Setting 1	linear	4	165
Voxtalisib	Setting 1	linear	5	610
Voxtalisib	Setting 1	linear	6	4254
NSC319726	Setting 1	cubic	1	3
NSC319726	Setting 1	cubic	2	6
NSC319726	Setting 1	cubic	3	32
NSC319726	Setting 1	cubic	4	93
NSC319726	Setting 1	cubic	5	153
NSC319726	Setting 1	cubic	6	470

NSC319726	Setting 1	quadratic	1	4
NSC319726	Setting 1	quadratic	2	12
NSC319726	Setting 1	quadratic	3	93
NSC319726	Setting 1	quadratic	4	135
NSC319726	Setting 1	quadratic	5	467
NSC319726	Setting 1	quadratic	6	1734
NSC319726	Setting 1	linear	1	1
NSC319726	Setting 1	linear	2	30
NSC319726	Setting 1	linear	3	125
NSC319726	Setting 1	linear	4	235
NSC319726	Setting 1	linear	5	1436
NSC319726	Setting 1	linear	6	8995
Niraparib	Setting 1	cubic	1	1
Niraparib	Setting 1	cubic	2	3
Niraparib	Setting 1	cubic	3	15
Niraparib	Setting 1	cubic	4	65
Niraparib	Setting 1	cubic	5	134
Niraparib	Setting 1	cubic	6	317
Niraparib	Setting 1	quadratic	1	0
Niraparib	Setting 1	quadratic	2	5
Niraparib	Setting 1	quadratic	3	28
Niraparib	Setting 1	quadratic	4	79
Niraparib	Setting 1	quadratic	5	142
Niraparib	Setting 1	quadratic	6	372
Niraparib	Setting 1	linear	1	1
Niraparib	Setting 1	linear	2	7
Niraparib	Setting 1	linear	3	52
Niraparib	Setting 1	linear	4	117
Niraparib	Setting 1	linear	5	201
Niraparib	Setting 1	linear	6	1100
Temsirolimus	Setting 3	cubic	1	7
Temsirolimus	Setting 3	cubic	2	16
Temsirolimus	Setting 3	cubic	3	122
Temsirolimus	Setting 3	cubic	4	315

Temsirolimus	Setting 3	cubic	5	1955
Temsirolimus	Setting 3	cubic	6	13281
Temsirolimus	Setting 3	quadratic	1	6
Temsirolimus	Setting 3	quadratic	2	40
Temsirolimus	Setting 3	quadratic	3	184
Temsirolimus	Setting 3	quadratic	4	330
Temsirolimus	Setting 3	quadratic	5	3298
Temsirolimus	Setting 3	quadratic	6	19885
Temsirolimus	Setting 3	linear	1	4
Temsirolimus	Setting 3	linear	2	89
Temsirolimus	Setting 3	linear	3	185
Temsirolimus	Setting 3	linear	4	889
Temsirolimus	Setting 3	linear	5	4035
Temsirolimus	Setting 3	linear	6	31163
CX-5461	Setting 3	cubic	1	5
CX-5461	Setting 3	cubic	2	7
CX-5461	Setting 3	cubic	3	109
CX-5461	Setting 3	cubic	4	198
CX-5461	Setting 3	cubic	5	701
CX-5461	Setting 3	cubic	6	2544
CX-5461	Setting 3	quadratic	1	6
CX-5461	Setting 3	quadratic	2	18
CX-5461	Setting 3	quadratic	3	168
CX-5461	Setting 3	quadratic	4	357
CX-5461	Setting 3	quadratic	5	2725
CX-5461	Setting 3	quadratic	6	6852
CX-5461	Setting 3	linear	1	6
CX-5461	Setting 3	linear	2	51
CX-5461	Setting 3	linear	3	235
CX-5461	Setting 3	linear	4	2132
CX-5461	Setting 3	linear	5	14777
Rapamycin	Setting 3	cubic	1	3
Rapamycin	Setting 3	cubic	2	12
Rapamycin	Setting 3	cubic	3	186

Rapamycin	Setting 3	cubic	4	367
Rapamycin	Setting 3	cubic	5	1873
Rapamycin	Setting 3	cubic	6	6656
Rapamycin	Setting 3	quadratic	1	1
Rapamycin	Setting 3	quadratic	2	8
Rapamycin	Setting 3	quadratic	3	126
Rapamycin	Setting 3	quadratic	4	385
Rapamycin	Setting 3	quadratic	5	1757
Rapamycin	Setting 3	quadratic	6	12140
Rapamycin	Setting 3	linear	1	1
Rapamycin	Setting 3	linear	2	12
Rapamycin	Setting 3	linear	3	101
Rapamycin	Setting 3	linear	4	225
Rapamycin	Setting 3	linear	5	1302
Rapamycin	Setting 3	linear	6	7584
Dactolisib	Setting 3	cubic	1	1
Dactolisib	Setting 3	cubic	2	5
Dactolisib	Setting 3	cubic	3	54
Dactolisib	Setting 3	cubic	4	185
Dactolisib	Setting 3	cubic	5	338
Dactolisib	Setting 3	cubic	6	1719
Dactolisib	Setting 3	quadratic	1	6
Dactolisib	Setting 3	quadratic	2	6
Dactolisib	Setting 3	quadratic	3	81
Dactolisib	Setting 3	quadratic	4	197
Dactolisib	Setting 3	quadratic	5	528
Dactolisib	Setting 3	quadratic	6	2469
Dactolisib	Setting 3	linear	1	6
Dactolisib	Setting 3	linear	2	14
Dactolisib	Setting 3	linear	3	107
Dactolisib	Setting 3	linear	4	179
Dactolisib	Setting 3	linear	5	1045
Dactolisib	Setting 3	linear	6	3689
Talazoparib	Setting 3	cubic	1	3

Talazoparib	Setting 3	cubic	2	5
Talazoparib	Setting 3	cubic	3	21
Talazoparib	Setting 3	cubic	4	94
Talazoparib	Setting 3	cubic	5	172
Talazoparib	Setting 3	cubic	6	317
Talazoparib	Setting 3	quadratic	1	6
Talazoparib	Setting 3	quadratic	2	7
Talazoparib	Setting 3	quadratic	3	27
Talazoparib	Setting 3	quadratic	4	84
Talazoparib	Setting 3	quadratic	5	174
Talazoparib	Setting 3	quadratic	6	365
Talazoparib	Setting 3	linear	1	5
Talazoparib	Setting 3	linear	2	12
Talazoparib	Setting 3	linear	3	68
Talazoparib	Setting 3	linear	4	131
Talazoparib	Setting 3	linear	5	323
Talazoparib	Setting 3	linear	6	3916

RF/KNN results

Figures B.6 to B.12 depict the performance for random forests and k-nearest neighbors using both of the mentioned model selection criteria. For k-nearest neighbors, the ROC selection criterion leads to models with high specificity and very low sensitivity for all of the investigated 41 drugs. Notably, almost only drugs that are balanced with respect to sensitive and resistant cell lines (CX-5461, NSC319726, Niraparib, Omipalisib, Talazoparib) achieve sensitivities different from 0. By selecting the best model based on sensitivity alone, the average sensitivity could be improved (see Figure B.8 and B.9). However, it is still much lower than for our MERIDA method (cf. Figure 5.7 and 5.8).
For random forests, it can similarly be seen that the average sensitivity is comparatively low and is also only marginally improved by using sensitivity as selection criterion. In general, drugs with less class imbalance also seem to achieve a better prediction performance.

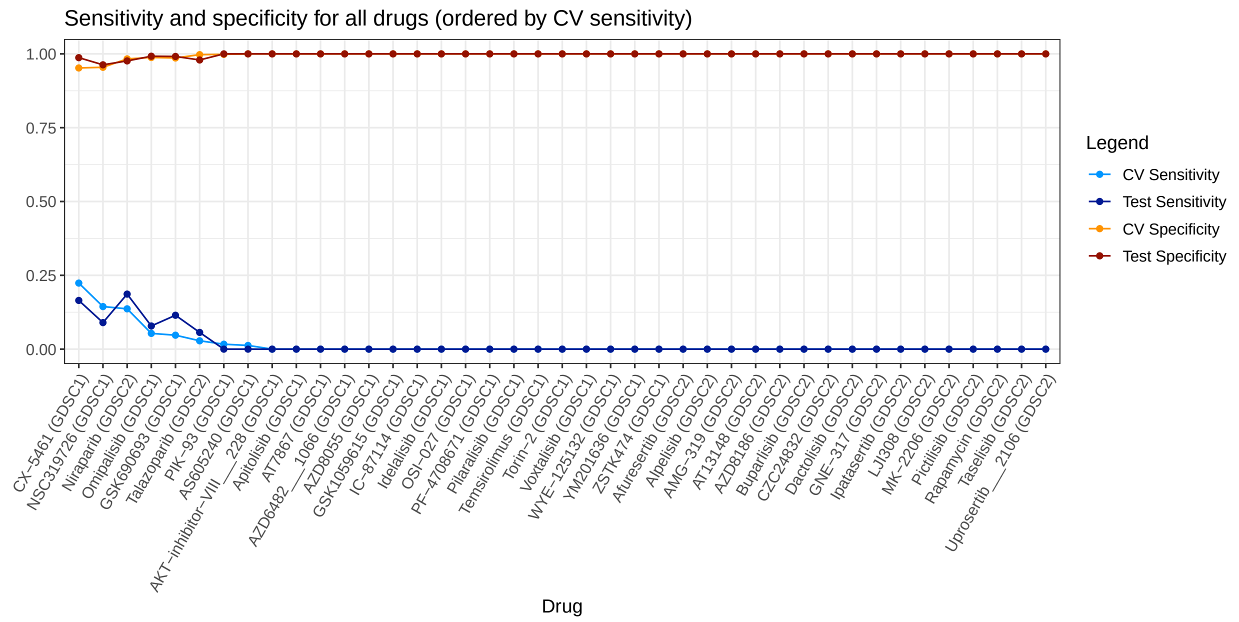


Figure B.6: This figure depicts the statistical performance of the best KNN model in Setting 1 for all 41 drugs using the ROC as selection criterion.

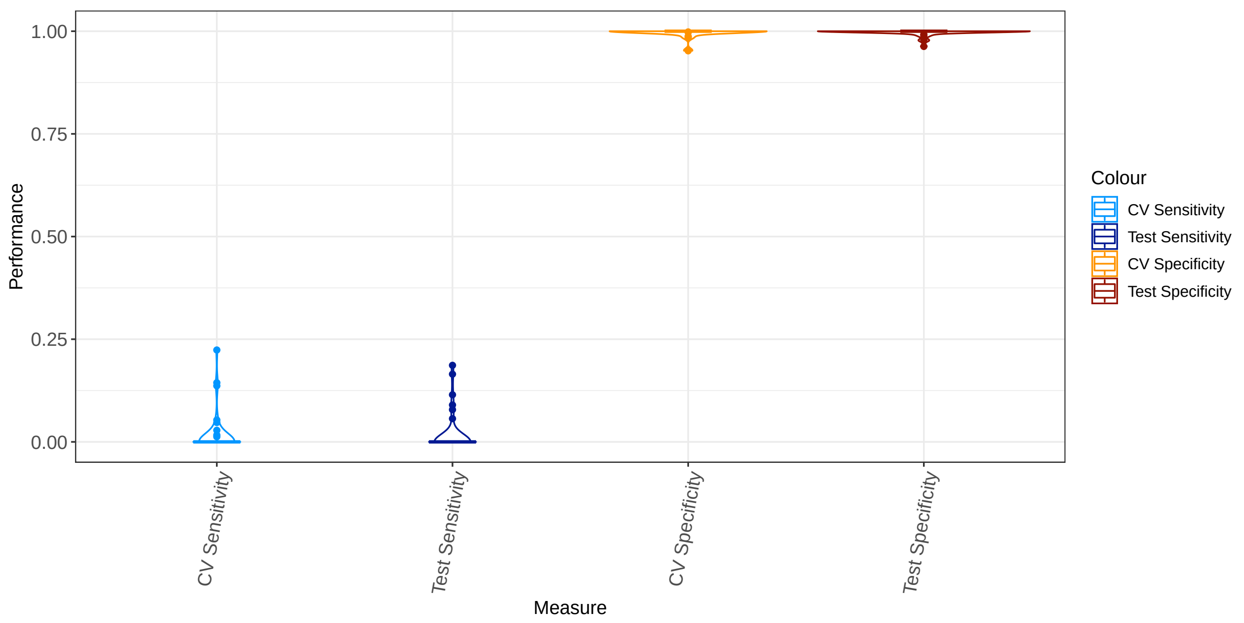


Figure B.7: This figure depicts the averaged statistical performance of the best KNN model in Setting 1 for all 41 drugs using the ROC as selection criterion.

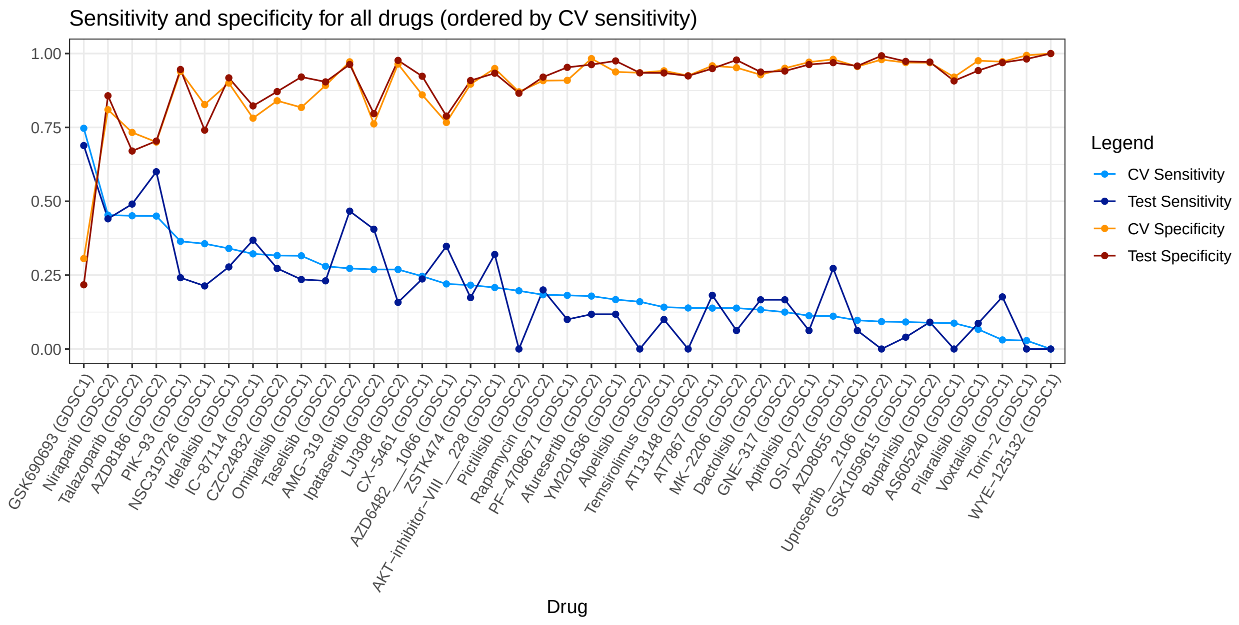


Figure B.8: This figure depicts the statistical performance of the best KNN model in Setting 1 for all 41 drugs using sensitivity as selection criterion.

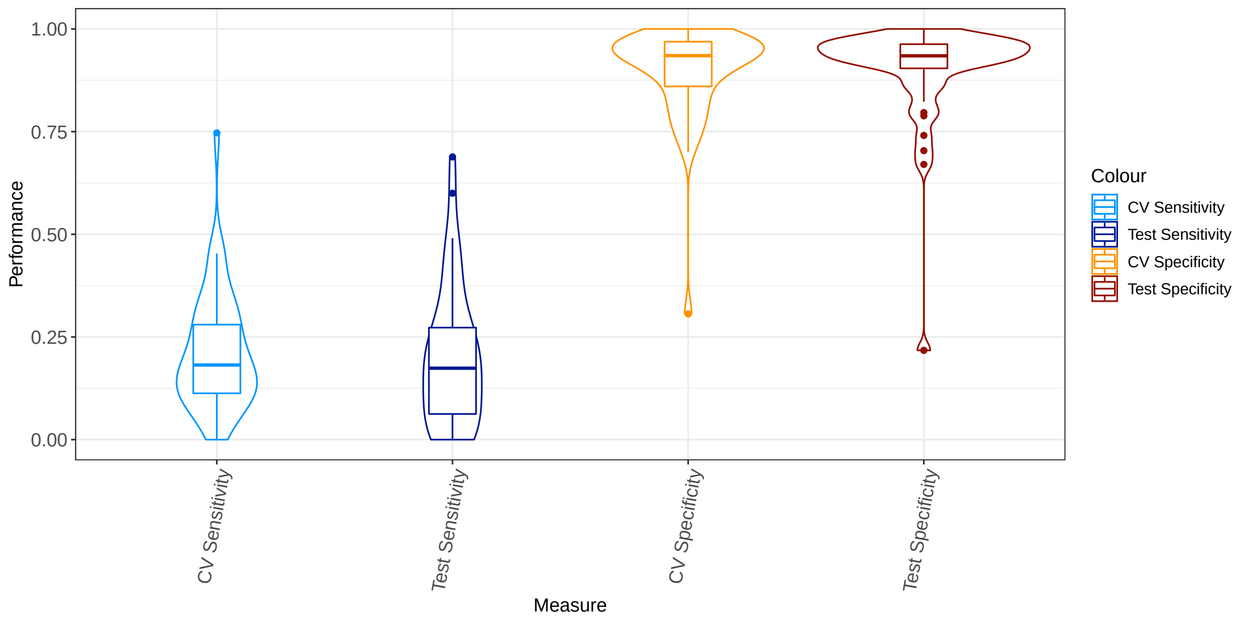


Figure B.9: This figure depicts the averaged statistical performance of the best KNN model in Setting 1 for all 41 drugs using sensitivity as selection criterion.

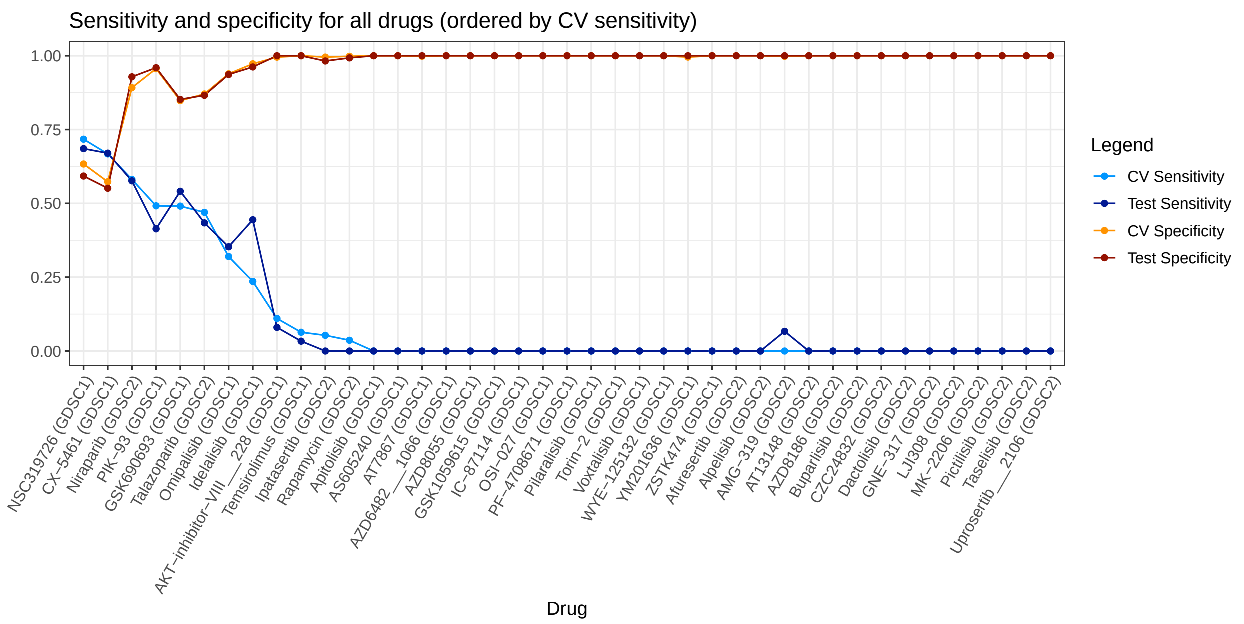


Figure B.10: This figure depicts the statistical performance of the best RF model in Setting 1 for all 41 drugs using the ROC as selection criterion.

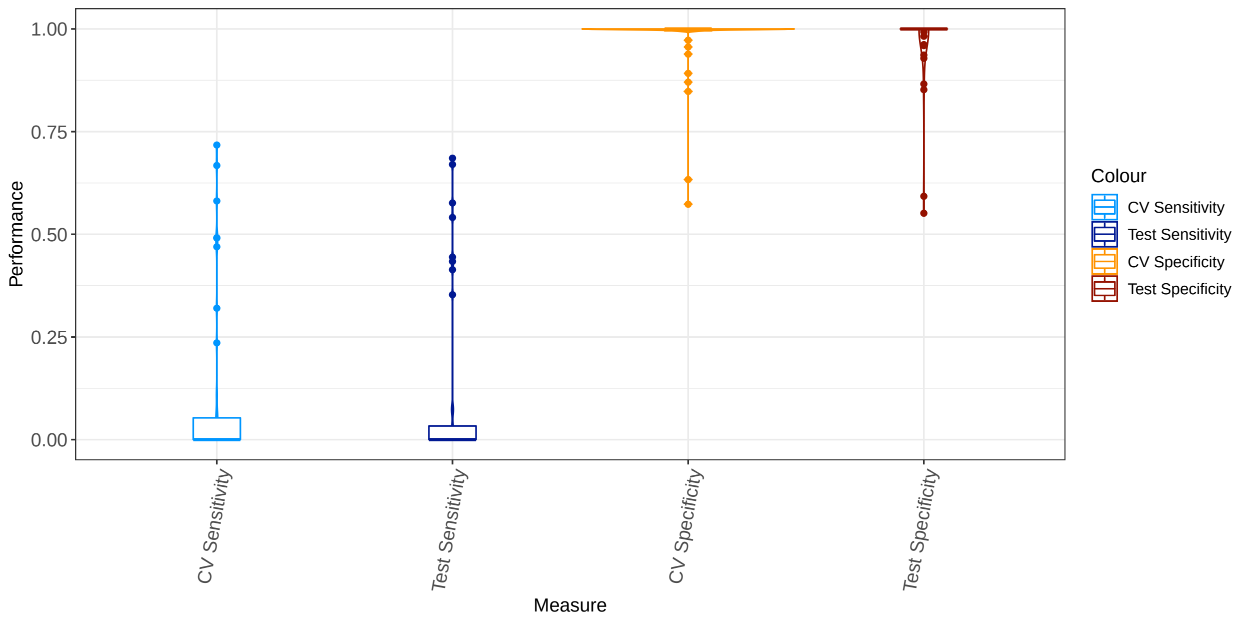


Figure B.11: This figure depicts the averaged statistical performance of the best RF model in Setting 1 for all 41 drugs using the ROC as selection criterion.

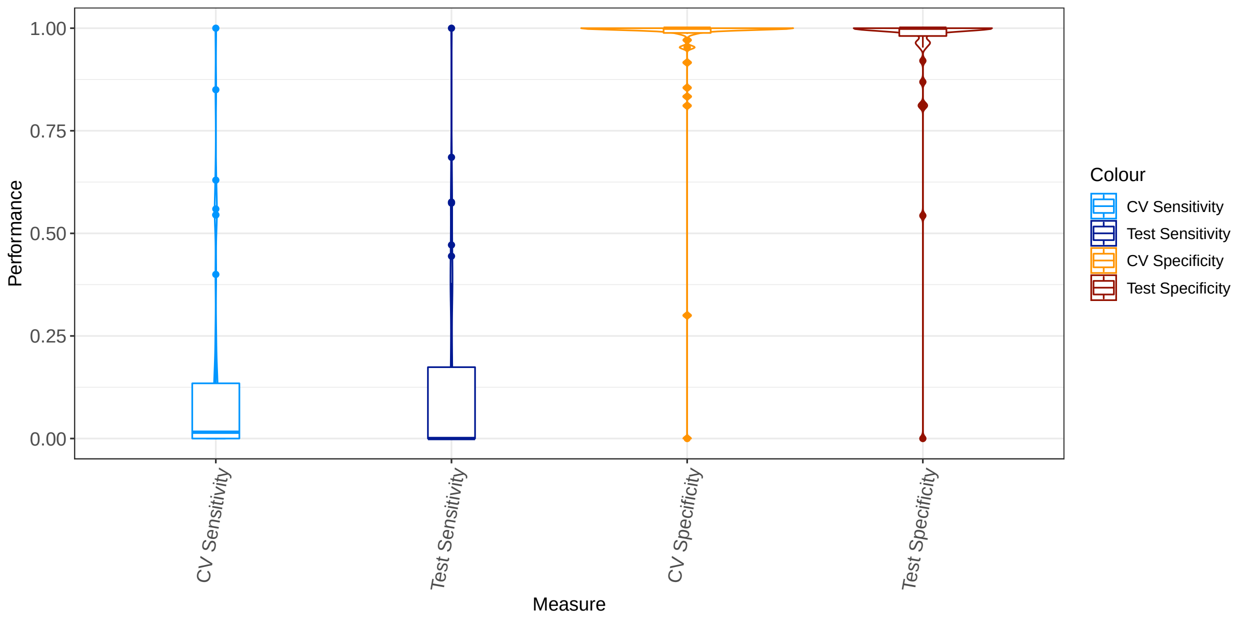


Figure B.12: This figure depicts the averaged statistical performance of the best RF model in Setting 1 for all 41 drugs using sensitivity as selection criterion.

Site specific analysis

The main analysis focuses on the full pan-cancer data set from the GDSC. However, it is well-known that tissue or organs of origin plays a significant role in drug sensitivity and resistance. Therefore, we also applied our method to two interesting sub-groups of cell lines: haematological cancer cell lines and non-haematological cancer cell lines. In the following, we will first explain how we prepared the data. Then, we shortly present the results with respect to statistical performance and selected biomarkers.

Data preparation

In general, the performance of statistical learning methods strongly depends on the number of available samples in comparison to the number of available features. As already discussed, dimension reduction steps are therefore indispensable in this field of bioinformatics. When subdividing the pan-cancer data set into single sites, most of them will be too small to be analyzed. Therefore, we decided to investigate two groups of cell lines that still have a considerable amount of samples that we determined as described in the following.
When observing the site distribution of cell lines across the drug responses of the different drugs, we noted that one particular group of cell lines, the haematological cancer cell lines (i.e. leukemia, lymphoma, and myeloma), is often found to be very sensitive to drugs (cf. Figure B.13 and B.14 for mTOR pathway inhibitors). We then use this group containing approximately 160 cell lines per drug and the group of the non-haematological cancer cell lines containing approximately 700 cell lines per drug for further analysis.
We prepared the data sets exactly as already described. This means that we first divided the available cell lines into the two aforementioned groups and then performed the preparation of the data sets including literature-based feature selection and annotation. We also analyzed these two groups in the first two of the three settings :

- Setting 1: the a priori knowledge is not included, i.e. the matrix without a priori knowledge is used

- Setting 2: the information about the sensitivity biomarkers is integrated in the input feature matrix as one specific composite feature and the value of the corresponding ILP feature variable is fixed to 1

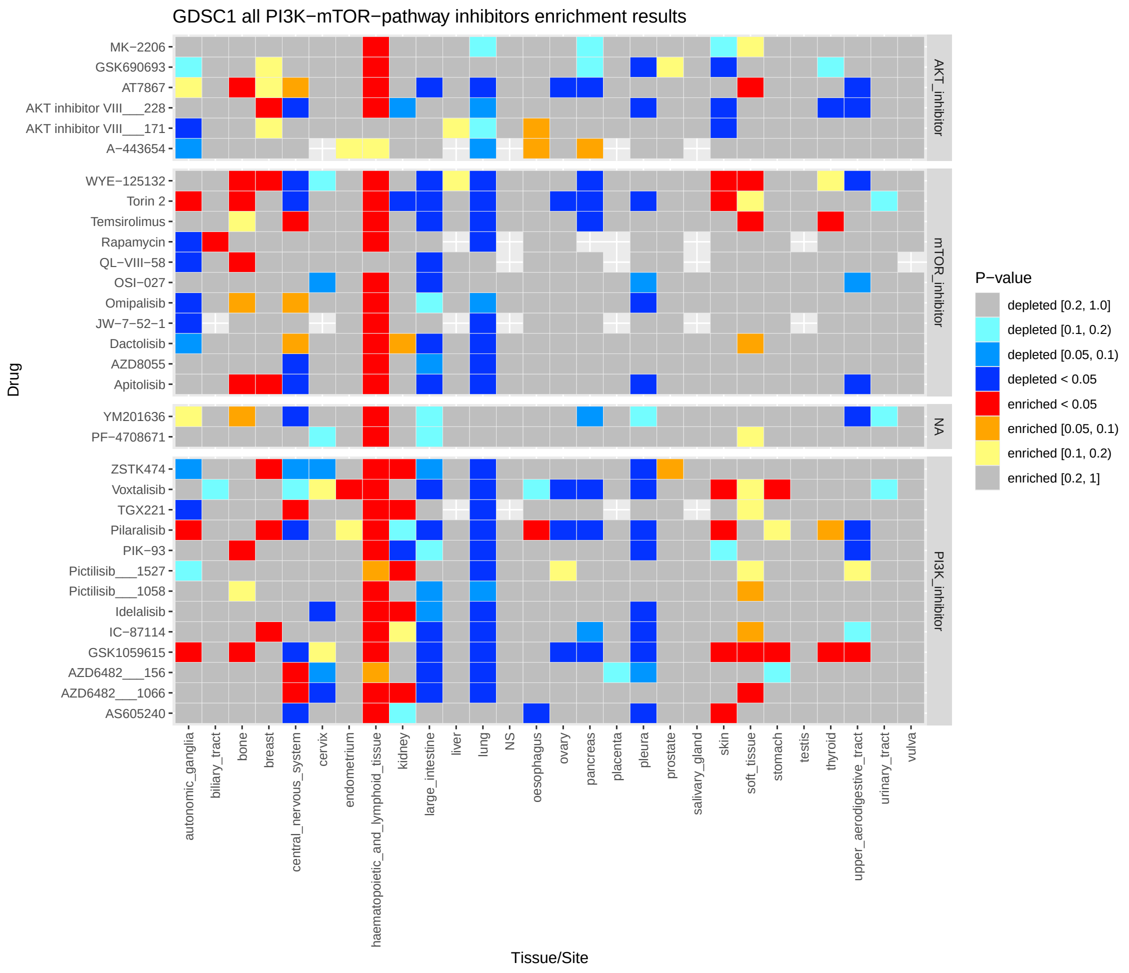


Figure B.13: This figure shows the results of an enrichment analysis for the different sites available in the GDSC database for the mTOR pathway inhibitors in GDSC1. The enrichment analysis was conducted using the respective site as category, and the increasingly sorted logarithmized IC50 lists as ranked lists. The p-values were adjusted using the Benjamini-Yekutieli procedure.

Results

Figures B.15 to B.22 show the statistical performance for MERIDA on the 41 different drugs for the non-haematological and haematological cell lines. For Setting 1, it

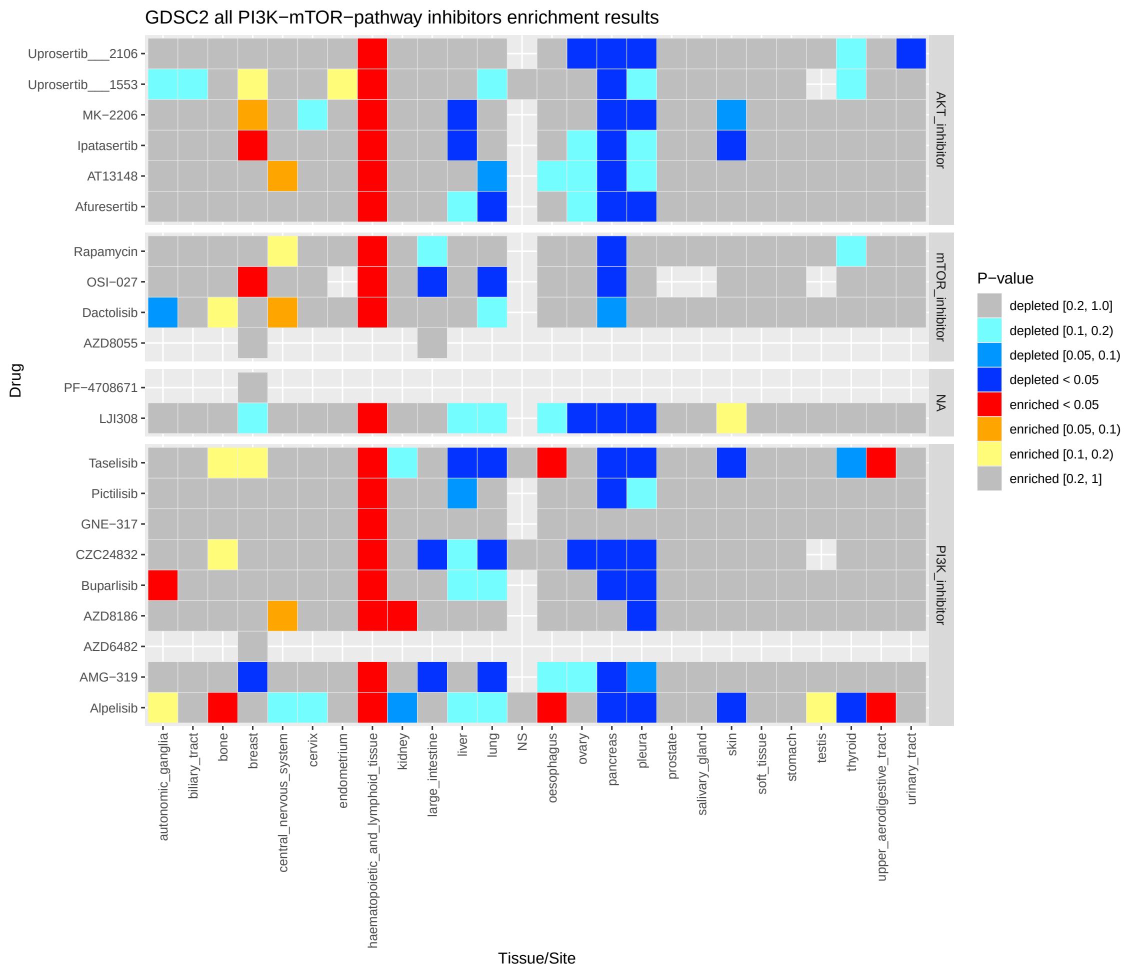


Figure B.14: In analogy to Figure B.13, this figure shows the results of an enrichment analysis for the different sites available in the GDSC database for the mTOR pathway inhibitors in GDSC2. The enrichment analysis was conducted using the respective site as category, and the sorted IC50 lists as ranked lists. The p-values were adjusted using the Benjamini-Yekutieli procedure.

can be seen that both specificity and sensitivity are highest for the pan-cancer analysis (cf. Figure 5.7) and lowest for the haematological cancer analysis (cf. Figure B.18). This effect could generally be attributed to the fact that fewer cell lines are used in both, the haematological cancer analysis and the non-haematological cancer analysis in comparison to the pan-cancer analysis. For Setting 2, it can be noted that the a priori knowledge seems to be more favourable for the pan-cancer cell lines and the non-haematological cancer cell lines in terms of statistical sensitivity, which might indicate that the drug sensitivity of the haematological cancer cell lines is less

determined by the used biomarkers.
We also analyzed the importance of the site of origin in the resulting models. From our results for Setting 1, we can clearly see that the selected biomarkers from the non-haematological cancer group largely resemble those from the analysis using all cell lines (cf. Figure B.23 to B.26). There is still some intersection of the selected biomarkers for the haematological cancer cell lines with the biomarkers for some of the drugs (cf. Figure B.27 and B.28 and Table B.20 to Table B.22). However, the relationship is much less pronounced highlighting the need for site-specific models. The smallest overlap can be observed for the features from the haematological cancer cell lines with the features from the non-haematological cancer cell lines (see Table B.22).

Table B.20: Intersection of selected features between non-haematological cell lines and all cell lines

Drug	Database	Setting	P-value (Fisher's exact test)
Afuresertib	GDSC2	1	3.85E-07
AKT-inhibitor-VIII___228	GDSC1	1	1
Alpelisib	GDSC2	1	2.40E-06
AMG-319	GDSC2	1	6.15E-07
Apitolisib	GDSC1	1	2.90E-05
AS605240	GDSC1	1	8.01E-05
AT13148	GDSC2	1	3.21E-03
AT7867	GDSC1	1	6.58E-06
AZD6482___1066	GDSC1	1	8.60E-03
AZD8055	GDSC1	1	2.79E-04
AZD8186	GDSC2	1	1.62E-04
Buparlisib	GDSC2	1	6.55E-07
CX-5461	GDSC1	1	1.64E-03
CZC24832	GDSC2	1	2.35E-01
Dactolisib	GDSC2	1	4.94E-07
GNE-317	GDSC2	1	7.93E-05
GSK1059615	GDSC1	1	5.47E-08
GSK690693	GDSC1	1	2.93E-07
IC-87114	GDSC1	1	2.27E-01

Idelalisib	GDSC1	1	5.11E-02
Ipatasertib	GDSC2	1	1.74E-08
LJI308	GDSC2	1	6.53E-03
MK-2206	GDSC2	1	1.18E-02
Niraparib	GDSC2	1	5.07E-03
NSC319726	GDSC1	1	2.75E-12
Omipalisib	GDSC1	1	6.71E-03
OSI-027	GDSC1	1	5.52E-07
PF-4708671	GDSC1	1	7.27E-08
Pictilisib	GDSC2	1	2.58E-04
PIK-93	GDSC1	1	1.64E-03
Pilaralisib	GDSC1	1	7.78E-09
Rapamycin	GDSC2	1	1.90E-02
Talazoparib	GDSC2	1	7.41E-06
Taselisib	GDSC2	1	2.28E-11
Temsirolimus	GDSC1	1	8.23E-03
Torin-2	GDSC1	1	3.66E-07
Uprosertib___2106	GDSC2	1	2.38E-06
Voxtalisib	GDSC1	1	6.98E-06
WYE-125132	GDSC1	1	7.87E-11
YM201636	GDSC1	1	9.90E-03
ZSTK474	GDSC1	1	1

Table B.21: Intersection of selected features between haematological cell lines and all cell lines

Drug	Database	Setting	P-value (Fisher's exact test)
Afuresertib	GDSC2	1	1.66E-01
AKT-inhibitor-VIII___228	GDSC1	1	1
Alpelisib	GDSC2	1	1.81E-01
AMG-319	GDSC2	1	1
Apitolisib	GDSC1	1	1
AS605240	GDSC1	1	1.92E-01
AT13148	GDSC2	1	1

AT7867	GDSC1	1	1.29E-01
AZD6482___1066	GDSC1	1	1
AZD8055	GDSC1	1	1
AZD8186	GDSC2	1	2.03E-02
Buparlisib	GDSC2	1	1
CX-5461	GDSC1	1	1
CZC24832	GDSC2	1	1.39E-01
Dactolisib	GDSC2	1	1.15E-02
GNE-317	GDSC2	1	1
GSK1059615	GDSC1	1	1.45E-01
GSK690693	GDSC1	1	1
IC-87114	GDSC1	1	1
Idelalisib	GDSC1	1	1
Ipatasertib	GDSC2	1	1
LJI308	GDSC2	1	1
MK-2206	GDSC2	1	1
Niraparib	GDSC2	1	1
NSC319726	GDSC1	1	1
Omipalisib	GDSC1	1	1
OSI-027	GDSC1	1	1.46E-01
PF-4708671	GDSC1	1	2.63E-01
Pictilisib	GDSC2	1	3.92E-02
PIK-93	GDSC1	1	1.22E-01
Pilaralisib	GDSC1	1	1
Rapamycin	GDSC2	1	1
Talazoparib	GDSC2	1	1.37E-01
Taselisib	GDSC2	1	1
Temsirolimus	GDSC1	1	1.34E-01
Torin-2	GDSC1	1	1.19E-01
Uprosertib___2106	GDSC2	1	2.56E-01
Voxtalisib	GDSC1	1	1.16E-01
WYE-125132	GDSC1	1	2.93E-02
YM201636	GDSC1	1	1
ZSTK474	GDSC1	1	1

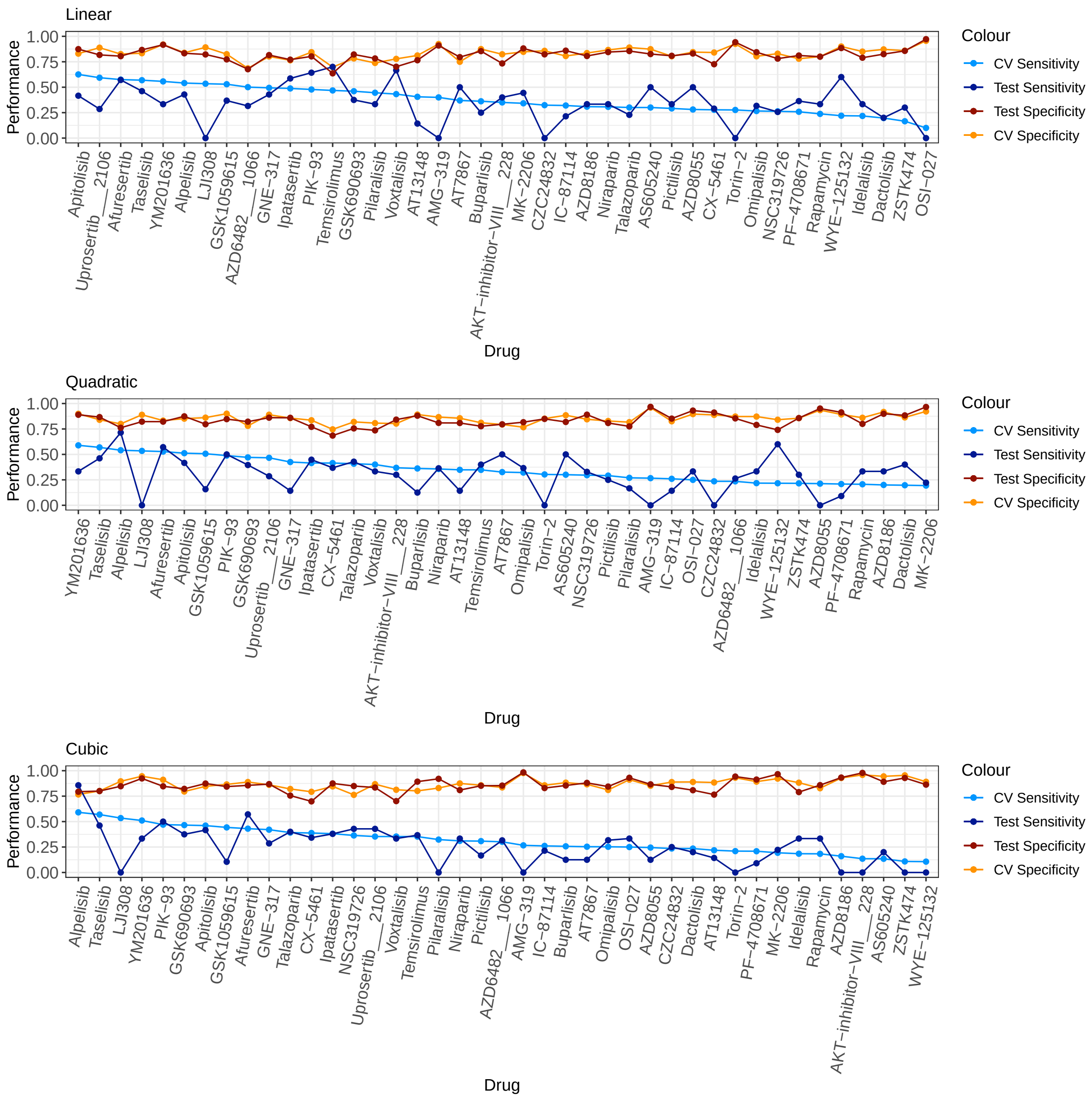


Figure B.15: This figure depicts the statistical performance of the best MERIDA model in Setting 1 for all 41 drugs for the non-haematological cell lines.

Table B.22: Intersection of selected features between non-haematological cell lines and haematological cell lines

Drug	Database	Setting	P-value (Fisher's exact test)
Afuresertib	GDSC2	1	1.76E-01

AKT-inhibitor-VIII___228	GDSC1	1	1
Alpelisib	GDSC2	1	1.73E-01
AMG-319	GDSC2	1	1
Apitolisib	GDSC1	1	1
AS605240	GDSC1	1	1
AT13148	GDSC2	1	1
AT7867	GDSC1	1	2.22E-01
AZD6482___1066	GDSC1	1	1
AZD8055	GDSC1	1	1
AZD8186	GDSC2	1	2.51E-01
Buparlisib	GDSC2	1	1
CX-5461	GDSC1	1	1
CZC24832	GDSC2	1	1
Dactolisib	GDSC2	1	1
GNE-317	GDSC2	1	1
GSK1059615	GDSC1	1	1
GSK690693	GDSC1	1	1
IC-87114	GDSC1	1	2.10E-01
Idelalisib	GDSC1	1	1
Ipatasertib	GDSC2	1	1
LJI308	GDSC2	1	1
MK-2206	GDSC2	1	1
Niraparib	GDSC2	1	1
NSC319726	GDSC1	1	1
Omipalisib	GDSC1	1	1
OSI-027	GDSC1	1	3.63E-03
PF-4708671	GDSC1	1	2.41E-01
Pictilisib	GDSC2	1	1
PIK-93	GDSC1	1	1
Pilaralisib	GDSC1	1	1
Rapamycin	GDSC2	1	1
Talazoparib	GDSC2	1	1.76E-01
Taselisib	GDSC2	1	1
Temsirolimus	GDSC1	1	1

Torin-2	GDSC1	1	1
Uprosertib___2106	GDSC2	1	2.24E-01
Voxtalisib	GDSC1	1	8.97E-02
WYE-125132	GDSC1	1	2.52E-01
YM201636	GDSC1	1	1
ZSTK474	GDSC1	1	1

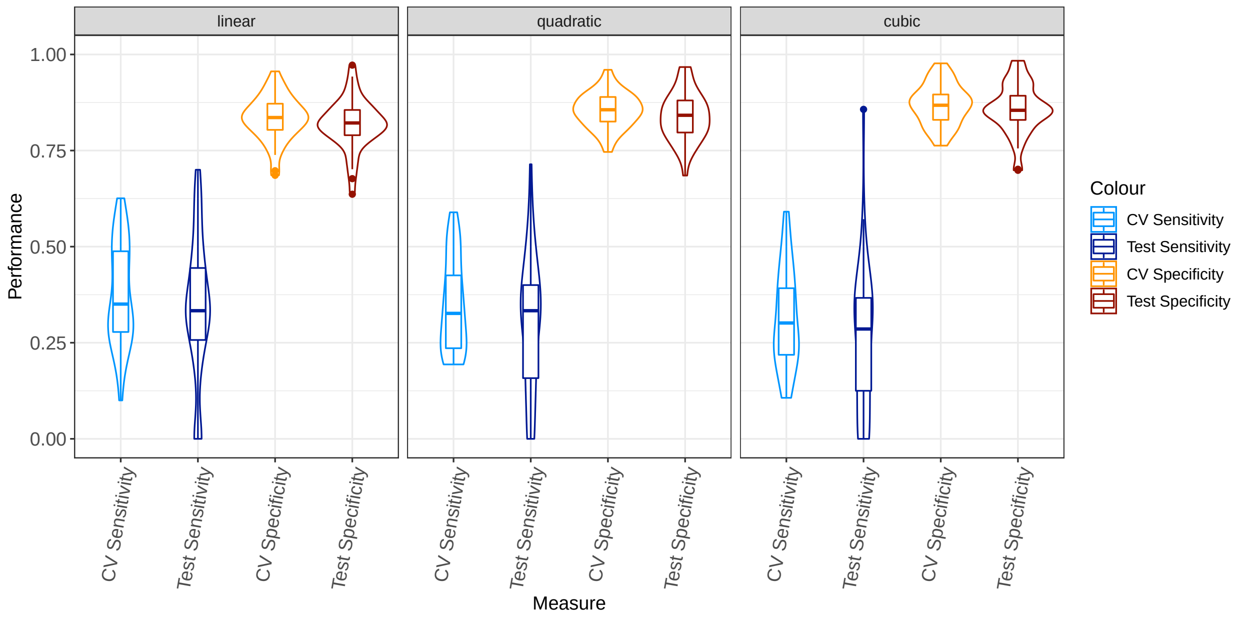


Figure B.16: This figure depicts the averaged statistical performance of the best MERIDA model in Setting 1 for all 41 drugs for the non-haematological cell lines.

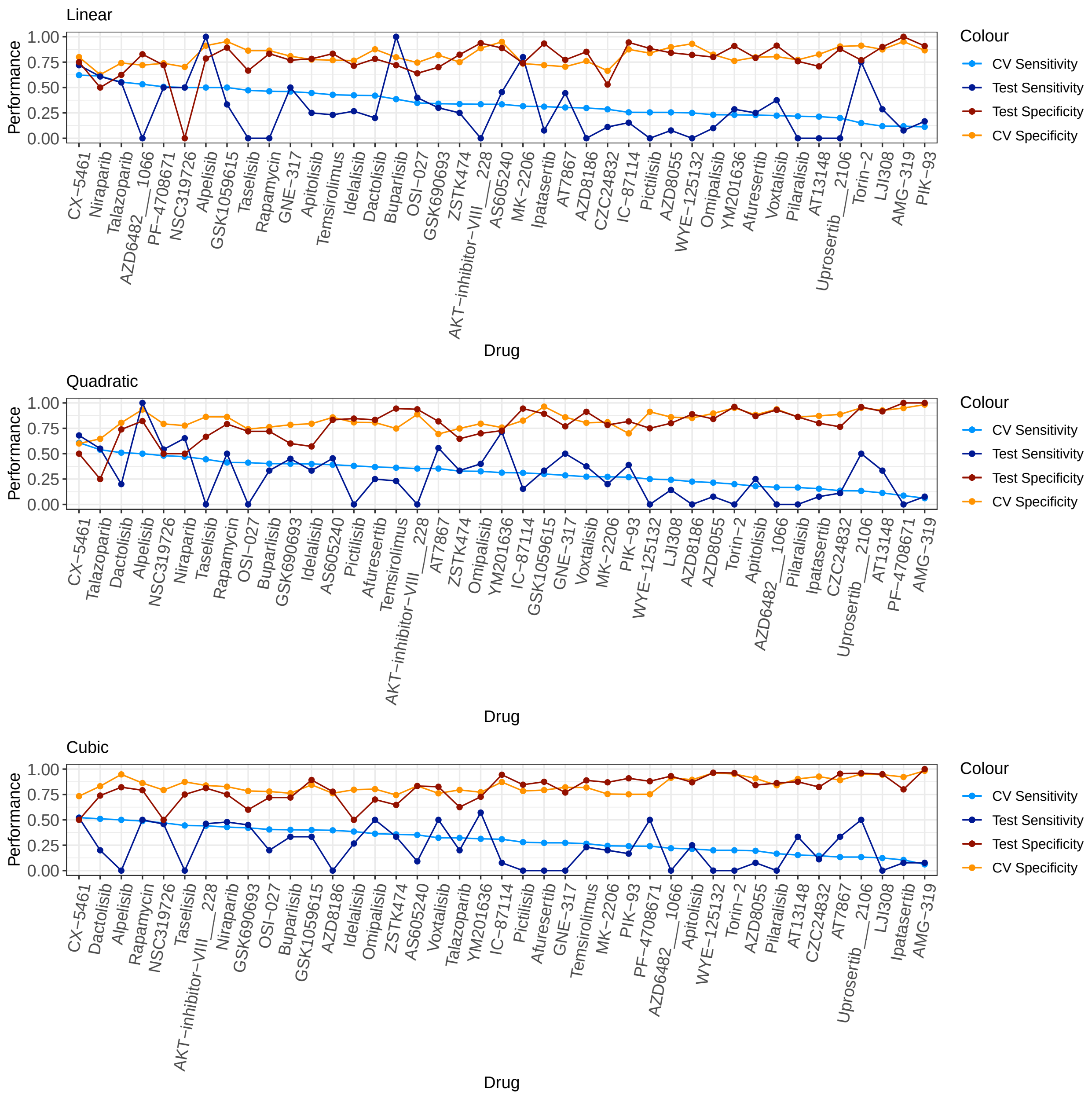


Figure B.17: This figure depicts the statistical performance of the best MERIDA model in Setting 1 for all 41 drugs for the haematological cell lines.

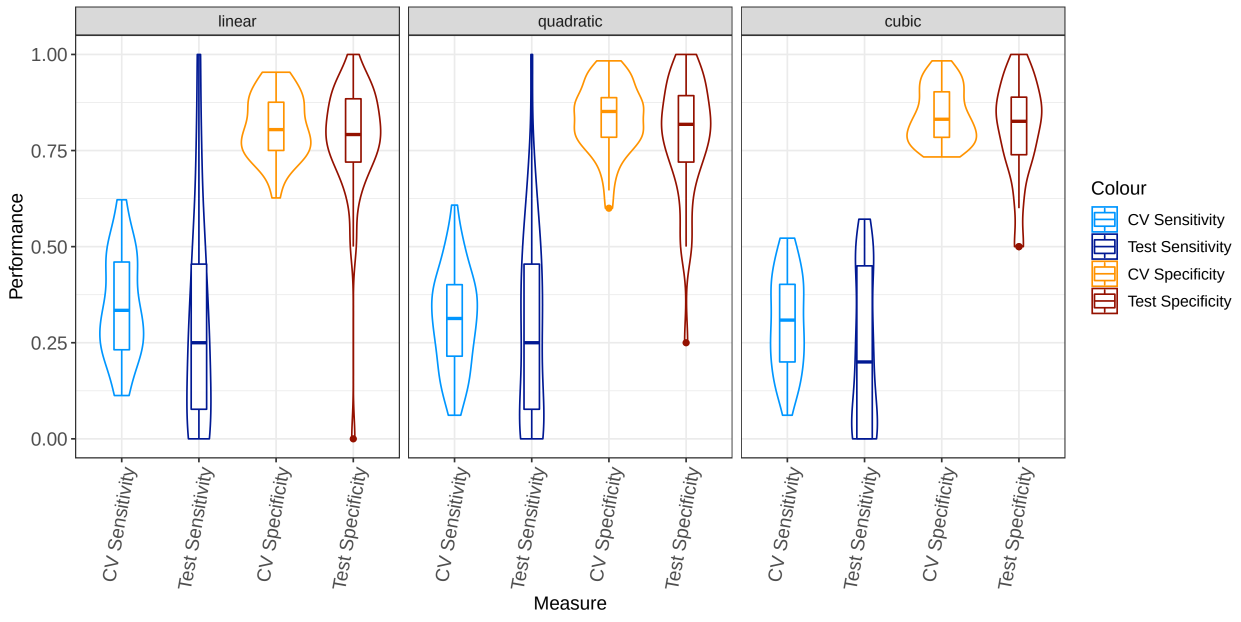


Figure B.18: This figure depicts the averaged statistical performance of the best MERIDA model in Setting 1 for all 41 drugs for the haematological cell lines.

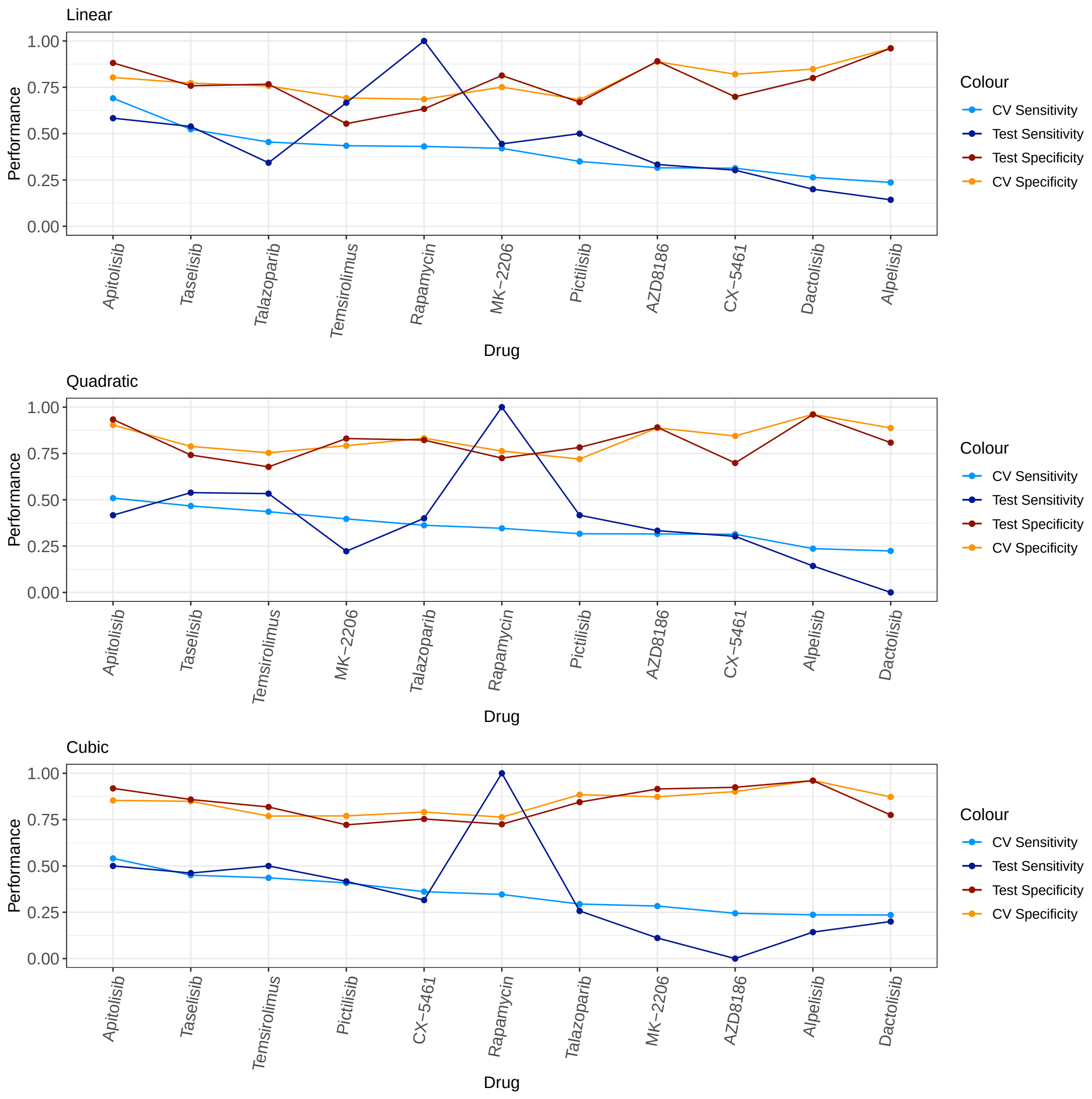


Figure B.19: This figure depicts the statistical performance of the best MERIDA model in Setting 2 for non-haematological cell lines.

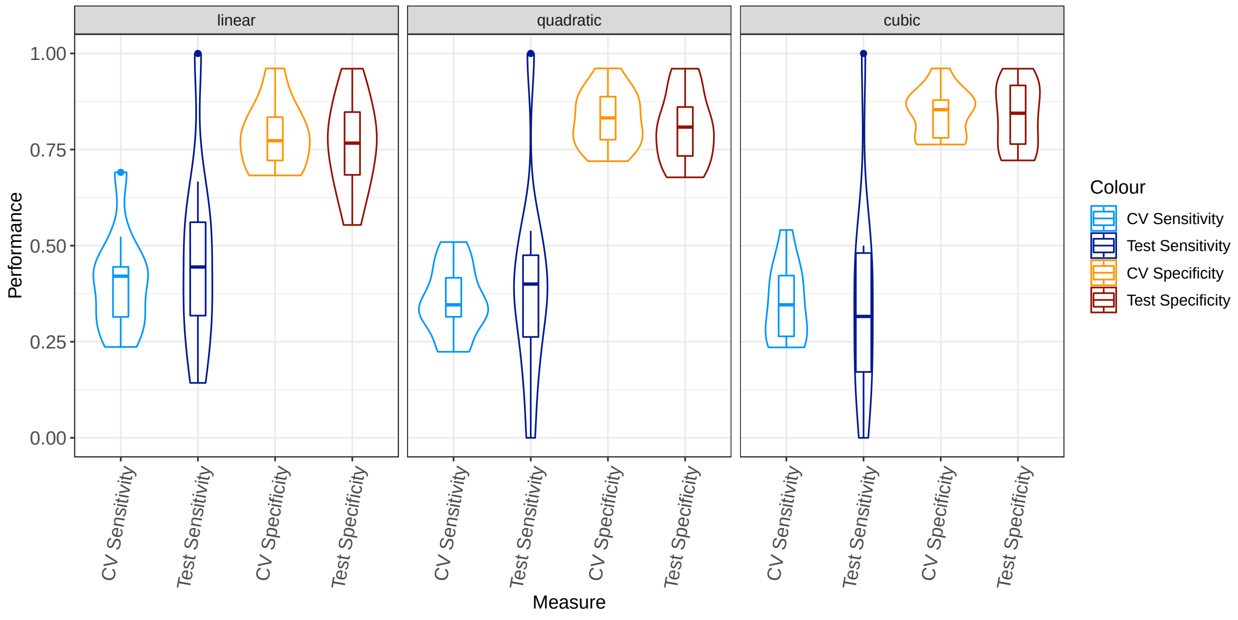


Figure B.20: This figure depicts the averaged statistical performance of the best MERIDA model in Setting 2 for non-haematological cell lines.

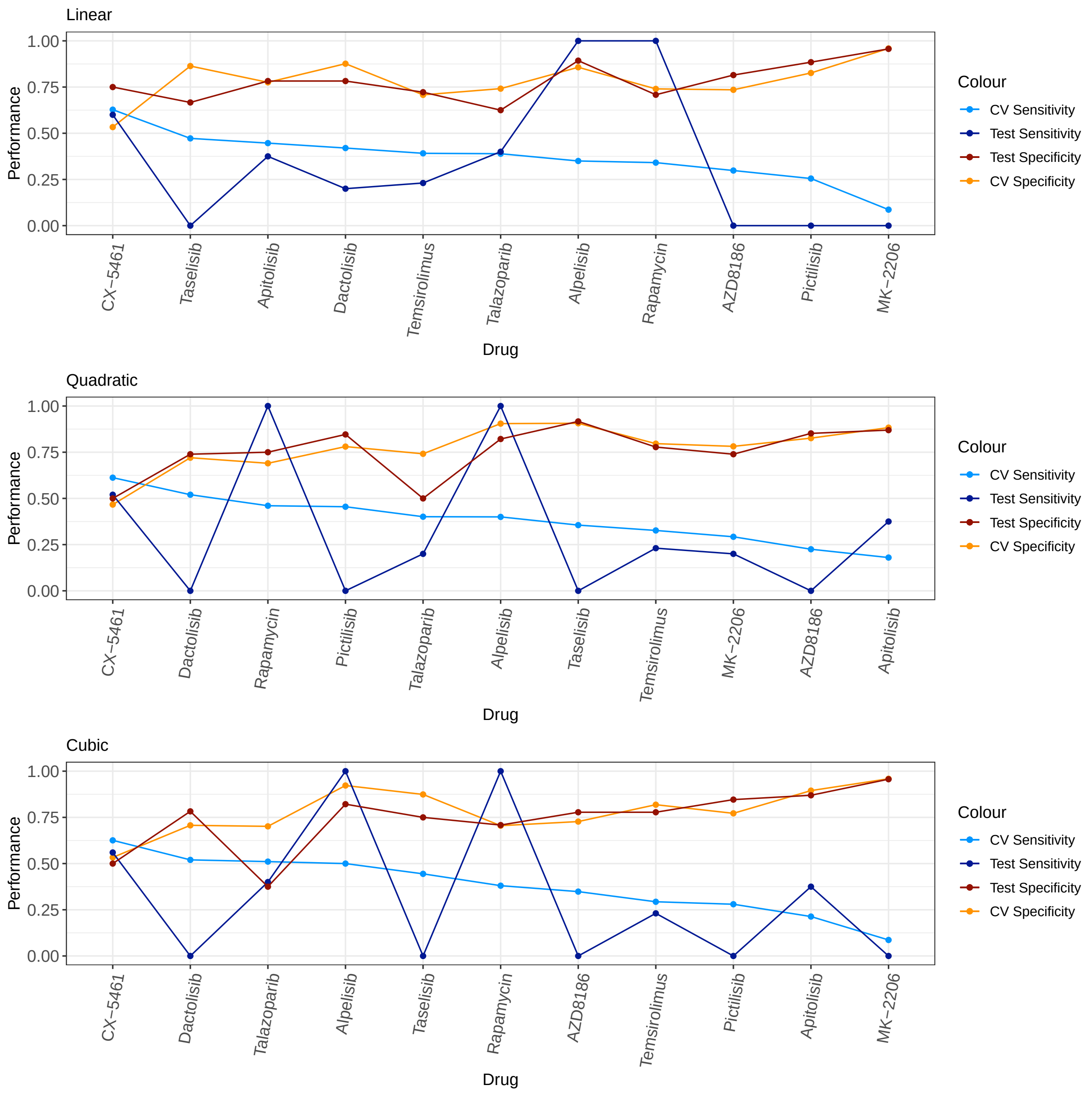


Figure B.21: This figure depicts the statistical performance of the best MERIDA model in Setting 2 for haematological cell lines.

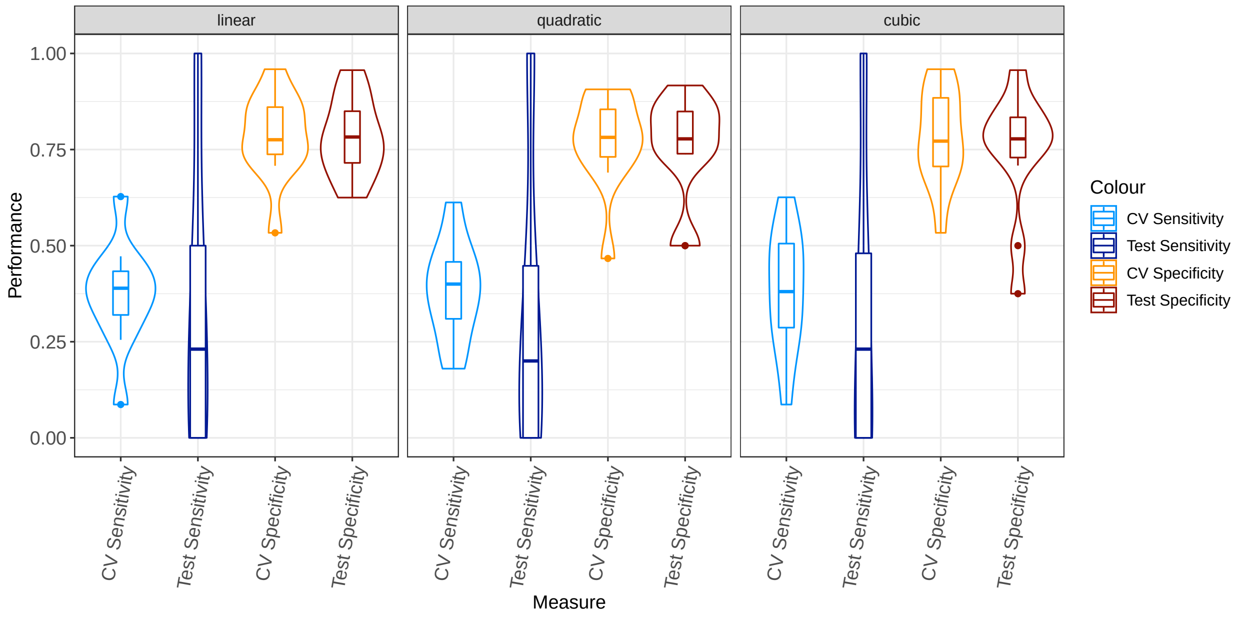


Figure B.22: This figure depicts the averaged statistical performance of the best MERIDA model in Setting 2 for the haematological cell lines.

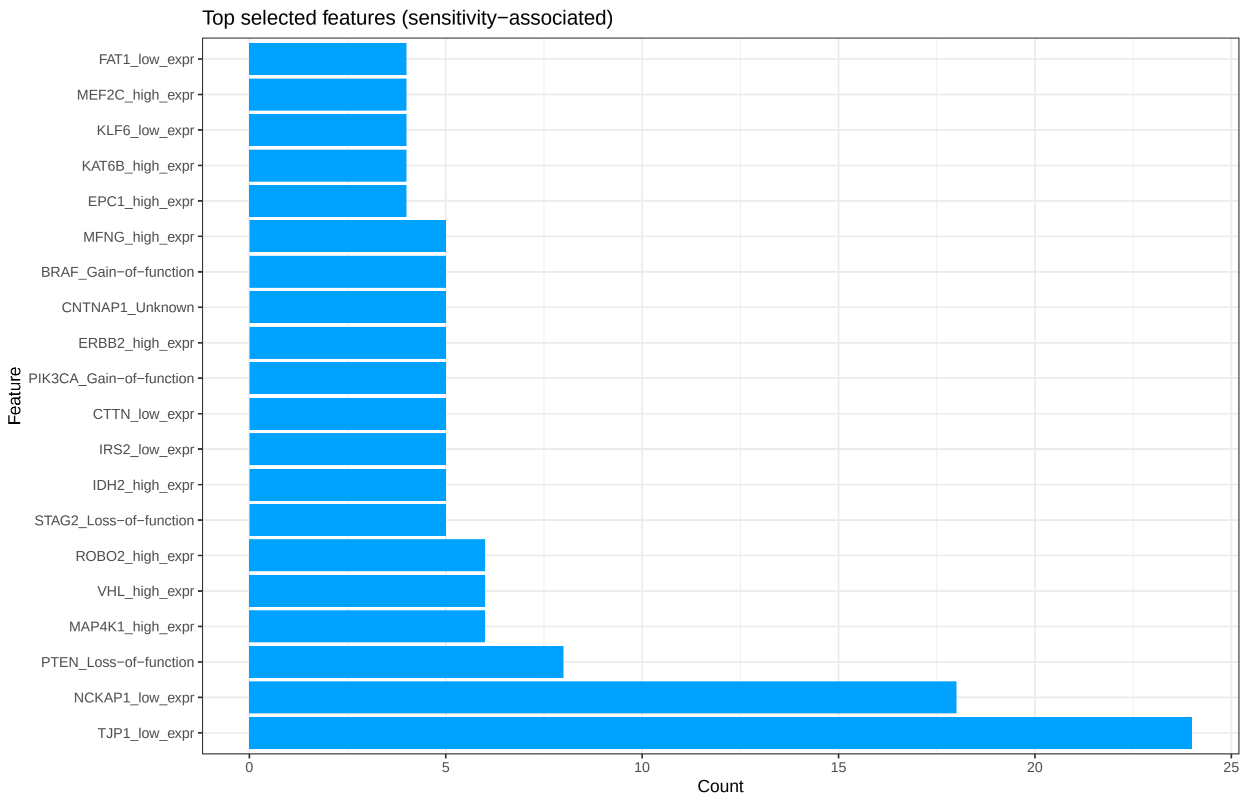


Figure B.23: This figure depicts the the frequency of the top 20 selected sensitivity-associated features (using all models of size 1 to 6 per drug) across all mTOR pathway inhibitors (37) for Setting 1 in the pan-cancer analysis.

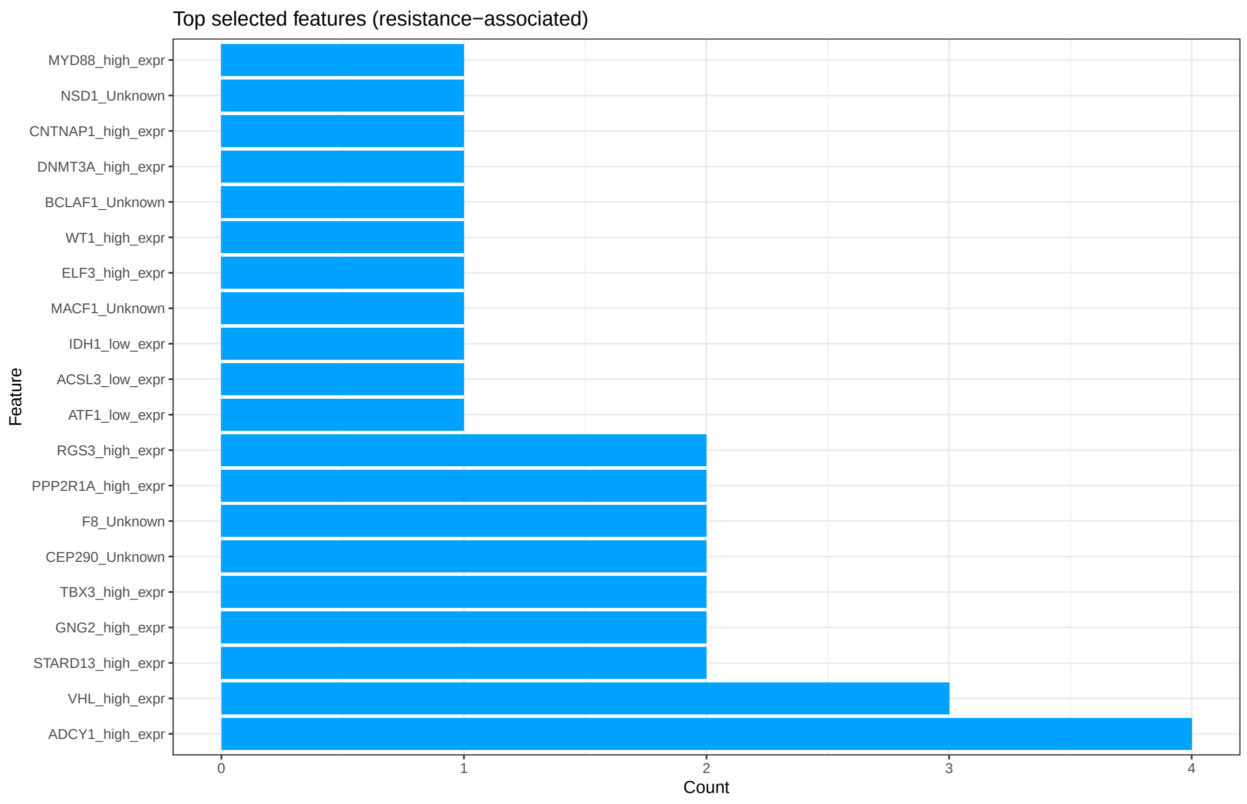


Figure B.24: This figure depicts the the frequency of the top 20 selected resistance-associated features (using all models of size 1 to 6 per drug) across all mTOR pathway inhibitors (37) for Setting 1 in the pan-cancer analysis.

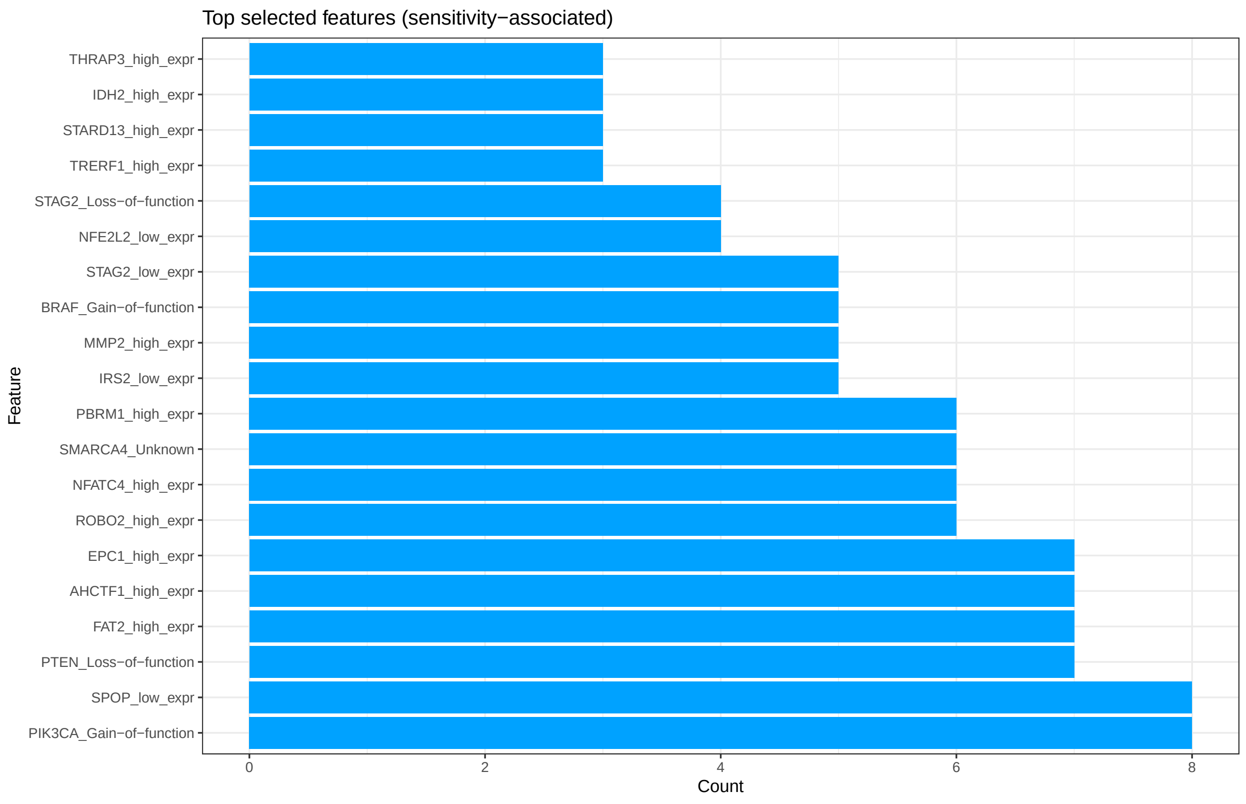


Figure B.25: This figure depicts the the frequency of the top 20 selected sensitivity-associated features (using all models of size 1 to 6 per drug) across all mTOR pathway inhibitors (37) for Setting 1 for the non-haematological cancer cell lines.

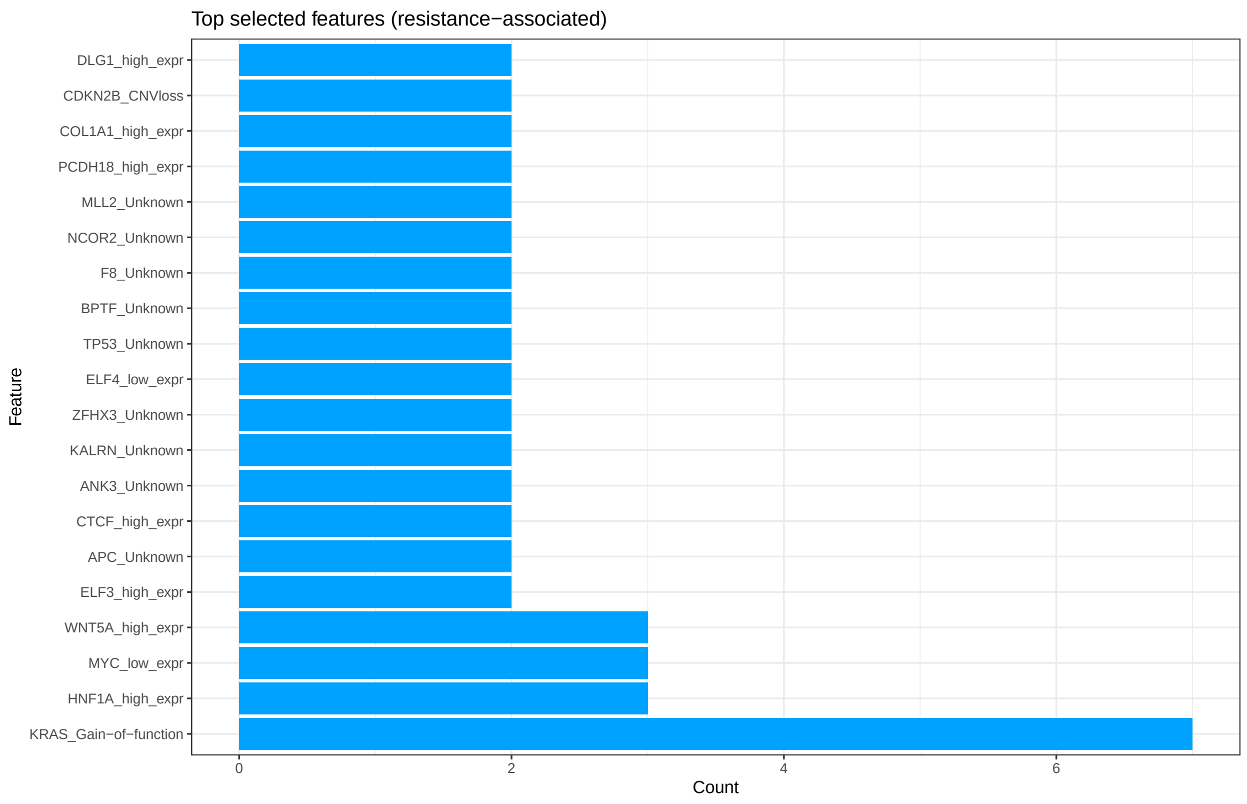


Figure B.26: This figure depicts the the frequency of the top 20 selected resistance-associated features (using all models of size 1 to 6 per drug) across all mTOR pathway inhibitors (37) for Setting 1 for the non-haematological cancer cell lines.

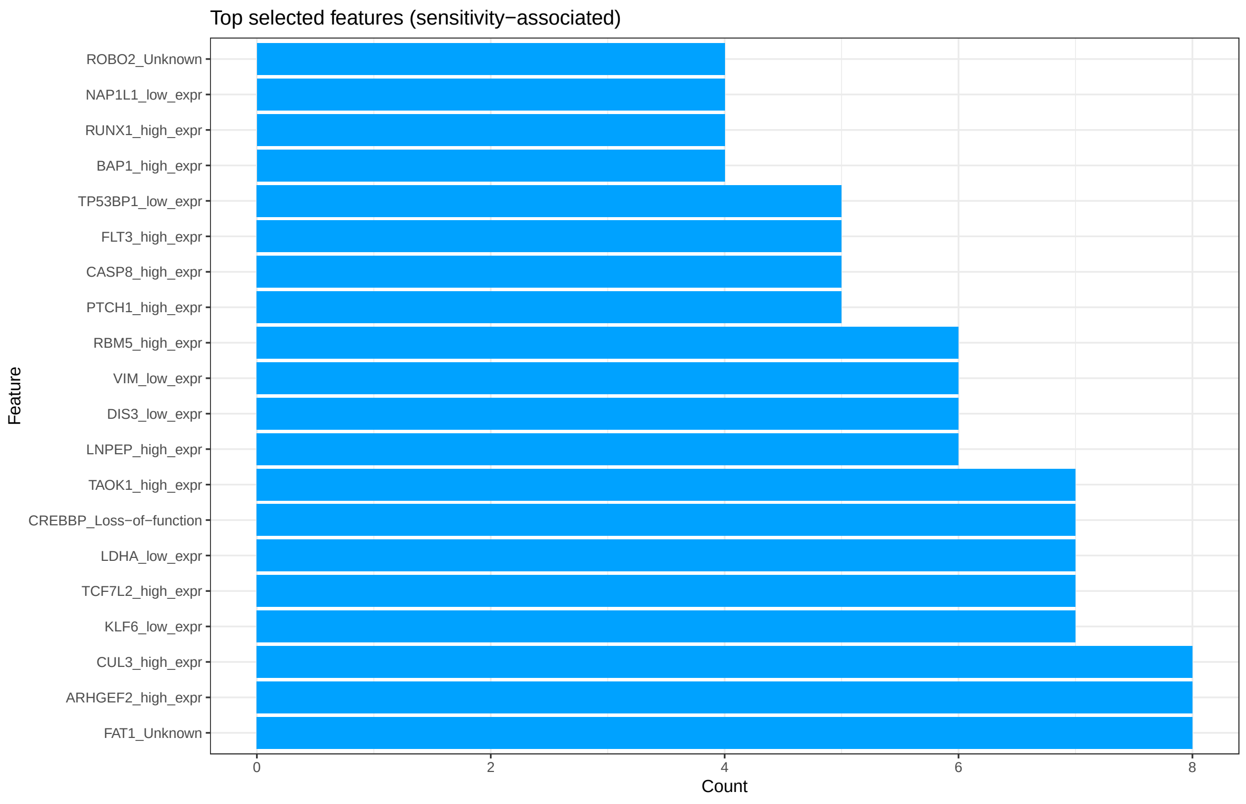


Figure B.27: This figure depicts the the frequency of the top 20 selected sensitivity-associated features (using all models of size 1 to 6 per drug) across all mTOR pathway inhibitors (37) for Setting 1 for the haematological cancer cell lines.

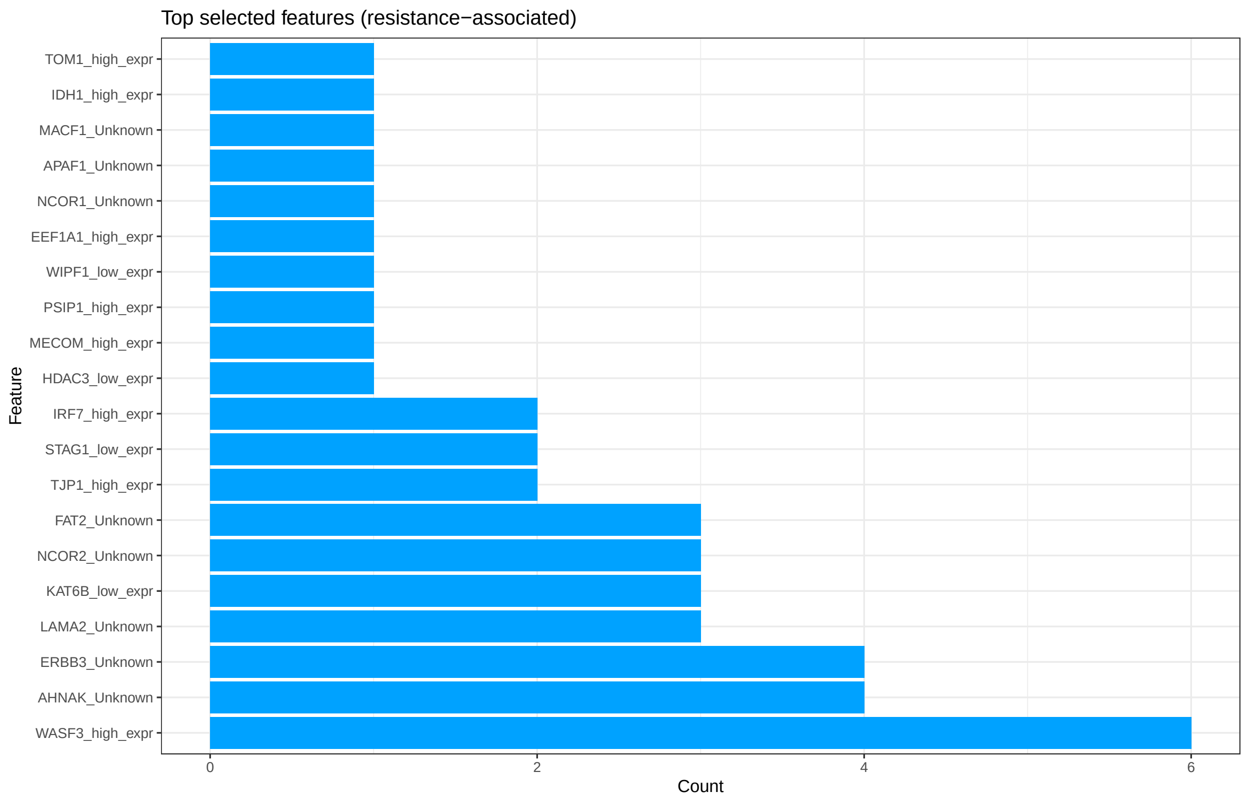
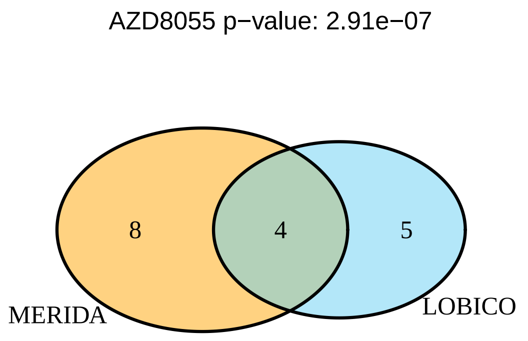
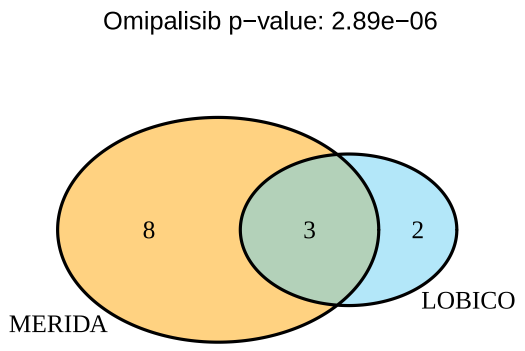


Figure B.28: This figure depicts the the frequency of the top 20 selected resistance-associated features (using all models of size 1 to 6 per drug) across all mTOR pathway inhibitors (37) for Setting 1 for the haematological cancer cell lines.

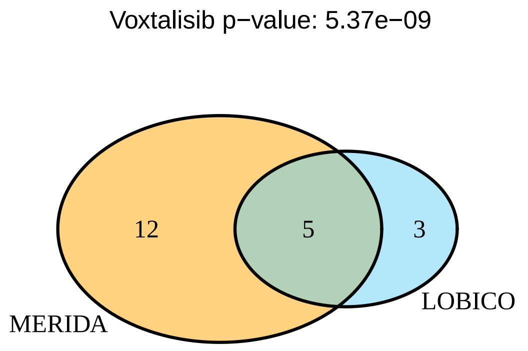
Pan-cancer analysis features



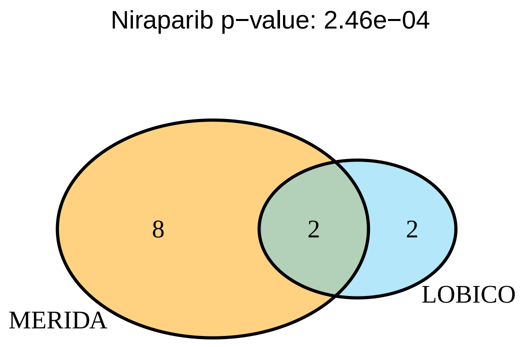
(a) AZD8055



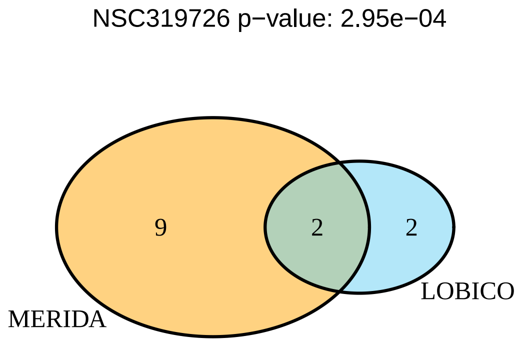
(b) Omipalisib



(c) Voxtalisib

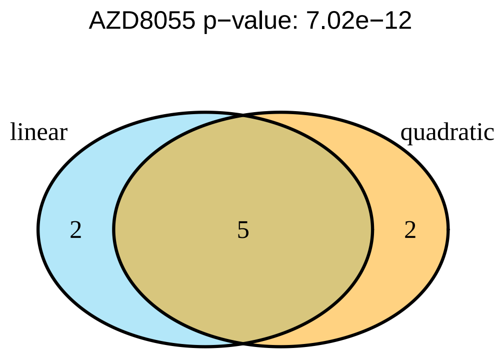


(d) Niraparib

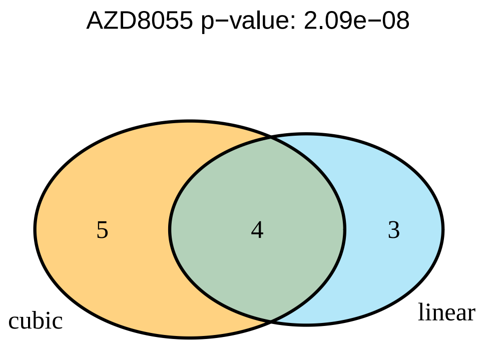


(e) NSC319726

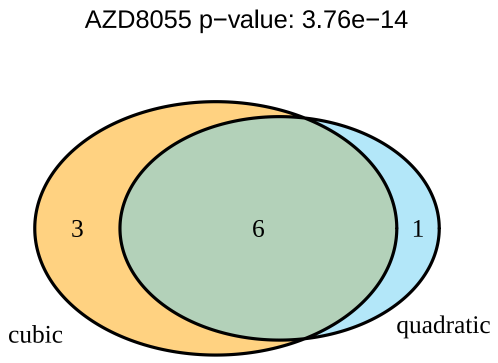
Figure B.29: In Sub-Figure (a - e), we present Venn diagrams with the respective Fisher's exact test overlap p-value for the selected features between LOBICO and MERIDA.



(a) AZD8055 linear vs quadratic



(b) AZD8055 linear vs cubic



(c) AZD8055 quadratic vs cubic

Figure B.30: In Sub-Figure (a - c), we present Venn diagrams with the respective Fisher's exact test overlap p-value for the selected features of MERIDA for AZD8055 between the three different weight functions. The remaining overlap p-values for all other drugs can be found in Table B.23.

Table B.23: Similarity of selected features for different weight functions for MERIDA

Drug	Database	Setting	Compared weight functions	P-value (Fisher's exact test)
AZD8055	GDSC1	1	linear, quadratic	7.02E-12
AZD8055	GDSC1	1	linear, cubic	2.09E-08
AZD8055	GDSC1	1	quadratic, cubic	3.76E-14
CX-5461	GDSC1	2	linear, quadratic	4.95E-14
CX-5461	GDSC1	2	linear, cubic	1.15E-08
CX-5461	GDSC1	2	quadratic, cubic	1.24E-14
CX-5461	GDSC1	3	linear, quadratic	4.21E-06

CX-5461	GDSC1	3	linear, cubic	2.77E-04
CX-5461	GDSC1	3	quadratic, cubic	9.29E-14
CX-5461	GDSC1	1	linear, quadratic	1.98E-04
CX-5461	GDSC1	1	linear, cubic	2.77E-04
CX-5461	GDSC1	1	quadratic, cubic	4.42E-16
Dactolisib	GDSC2	2	linear, quadratic	6.19E-22
Dactolisib	GDSC2	2	linear, cubic	2.72E-18
Dactolisib	GDSC2	2	quadratic, cubic	3.40E-20
Dactolisib	GDSC2	3	linear, quadratic	2.47E-20
Dactolisib	GDSC2	3	linear, cubic	4.07E-17
Dactolisib	GDSC2	3	quadratic, cubic	3.34E-22
Dactolisib	GDSC2	1	linear, quadratic	2.49E-20
Dactolisib	GDSC2	1	linear, cubic	4.09E-17
Dactolisib	GDSC2	1	quadratic, cubic	3.36E-22
Niraparib	GDSC2	1	linear, quadratic	1.30E-11
Niraparib	GDSC2	1	linear, cubic	1.30E-11
Niraparib	GDSC2	1	quadratic, cubic	7.01E-17
NSC319726	GDSC1	1	linear, quadratic	2.02E-06
NSC319726	GDSC1	1	linear, cubic	2.02E-06
NSC319726	GDSC1	1	quadratic, cubic	6.62E-17
Omipalisib	GDSC1	1	linear, quadratic	5.74E-09
Omipalisib	GDSC1	1	linear, cubic	2.15E-06
Omipalisib	GDSC1	1	quadratic, cubic	3.08E-15
Rapamycin	GDSC2	2	linear, quadratic	1.41E-14
Rapamycin	GDSC2	2	linear, cubic	1.69E-10
Rapamycin	GDSC2	2	quadratic, cubic	9.03E-16
Rapamycin	GDSC2	3	linear, quadratic	1.31E-05
Rapamycin	GDSC2	3	linear, cubic	1.17E-03
Rapamycin	GDSC2	3	quadratic, cubic	1.31E-05
Rapamycin	GDSC2	1	linear, quadratic	7.73E-06
Rapamycin	GDSC2	1	linear, cubic	1.23E-05
Rapamycin	GDSC2	1	quadratic, cubic	1.37E-14
Talazoparib	GDSC2	2	linear, quadratic	9.17E-11
Talazoparib	GDSC2	2	linear, cubic	3.73E-08

Talazoparib	GDSC2	2	quadratic, cubic	7.67E-12
Talazoparib	GDSC2	3	linear, quadratic	1.14E-22
Talazoparib	GDSC2	3	linear, cubic	8.24E-18
Talazoparib	GDSC2	3	quadratic, cubic	2.50E-20
Talazoparib	GDSC2	1	linear, quadratic	1.13E-22
Talazoparib	GDSC2	1	linear, cubic	8.20E-18
Talazoparib	GDSC2	1	quadratic, cubic	2.49E-20
Temsirolimus	GDSC1	2	linear, quadratic	1.24E-14
Temsirolimus	GDSC1	2	linear, cubic	1.86E-11
Temsirolimus	GDSC1	2	quadratic, cubic	1.90E-17
Temsirolimus	GDSC1	3	linear, quadratic	6.98E-12
Temsirolimus	GDSC1	3	linear, cubic	1.16E-08
Temsirolimus	GDSC1	3	quadratic, cubic	1.24E-14
Temsirolimus	GDSC1	1	linear, quadratic	6.93E-12
Temsirolimus	GDSC1	1	linear, cubic	1.15E-08
Temsirolimus	GDSC1	1	quadratic, cubic	1.24E-14
Voxtalisib	GDSC1	1	linear, quadratic	2.44E-12
Voxtalisib	GDSC1	1	linear, cubic	2.89E-07
Voxtalisib	GDSC1	1	quadratic, cubic	1.15E-19

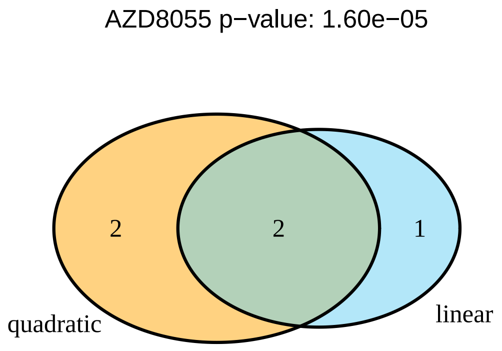
Table B.24: Similarity of selected features for different weight functions for LOBICO

Drug	Database	Setting	Compared weight functions	P-value (Fisher's exact test)
AZD8055	GDSC1	1	linear, quadratic	1.60E-05
AZD8055	GDSC1	1	linear, cubic	5.60E-05
AZD8055	GDSC1	1	quadratic, cubic	2.49E-07
CX-5461	GDSC1	3	linear, quadratic	7.98E-06
CX-5461	GDSC1	3	linear, cubic	5.98E-03
CX-5461	GDSC1	3	quadratic, cubic	7.98E-06
Dactolisib	GDSC2	3	linear, quadratic	1.87E-08
Dactolisib	GDSC2	3	linear, cubic	1.87E-08
Dactolisib	GDSC2	3	quadratic, cubic	1.73E-14
Niraparib	GDSC2	1	linear, quadratic	8.25E-06
Niraparib	GDSC2	1	linear, cubic	8.25E-06

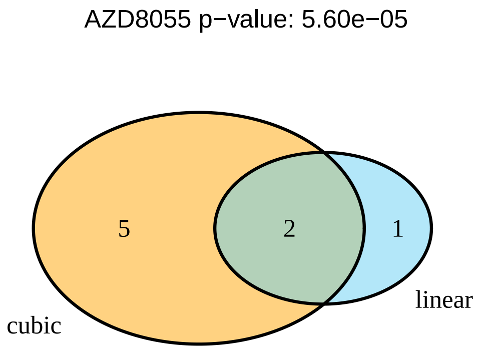
Niraparib	GDSC2	1	quadratic, cubic	1.87E-09
NSC319726	GDSC1	1	linear, quadratic	1.81E-09
NSC319726	GDSC1	1	linear, cubic	8.10E-06
NSC319726	GDSC1	1	quadratic, cubic	8.10E-06
Omipalisib	GDSC1	1	linear, quadratic	5.98E-03
Omipalisib	GDSC1	1	linear, cubic	5.98E-03
Omipalisib	GDSC1	1	quadratic, cubic	1.77E-09
Rapamycin	GDSC2	3	linear, quadratic	1.80E-10
Rapamycin	GDSC2	3	linear, cubic	2.64E-07
Rapamycin	GDSC2	3	quadratic, cubic	3.52E-15
Talazoparib	GDSC2	3	linear, quadratic	8.30E-06
Talazoparib	GDSC2	3	linear, cubic	8.30E-06
Talazoparib	GDSC2	3	quadratic, cubic	1.88E-09
Temsirolimus	GDSC1	3	linear, quadratic	7.99E-06
Temsirolimus	GDSC1	3	linear, cubic	1
Temsirolimus	GDSC1	3	quadratic, cubic	1
Voxtalisib	GDSC1	1	linear, quadratic	1
Voxtalisib	GDSC1	1	linear, cubic	1.19E-02
Voxtalisib	GDSC1	1	quadratic, cubic	3.55E-08

Table B.25: Comparison between selected features for LOBICO vs MERIDA

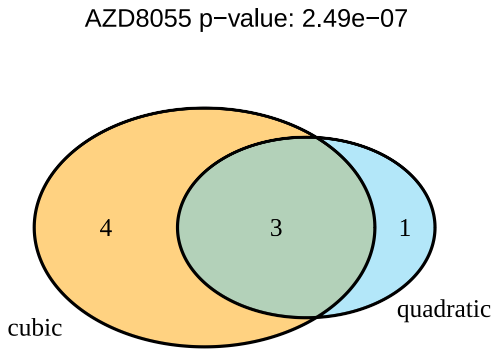
Drug	Database	Setting	P-value (Fisher's exact test)
AZD8055	GDSC1	1	2.91E-07
CX-5461	GDSC1	3	6.39E-06
Dactolisib	GDSC2	3	8.42E-06
Niraparib	GDSC2	1	2.46E-04
NSC319726	GDSC1	1	2.95E-04
Omipalisib	GDSC1	1	2.89E-06
Rapamycin	GDSC2	3	2.11E-13
Talazoparib	GDSC2	3	4.27E-04
Temsirolimus	GDSC1	3	1.53E-10
Voxtalisib	GDSC1	1	5.37E-09



(a) **AZD8055 linear vs quadratic**



(b) **AZD8055 linear vs cubic**



(c) **AZD8055 quadratic vs cubic**

Figure B.31: In Sub-Figure (a - c), we present Venn diagrams with the respective Fisher's exact test overlap p-value for the selected features of LOBICO for AZD8055 between the three different weight functions. The remaining overlap p-values for all other drugs can be found in Table B.24.

Enrichment analysis results

We employ the GeneTrail C++ library for performing a Gene Set Enrichment Analysis (GSEA) like analysis with the custom categories and ranked lists as described above, i.e. we test for an enrichment of a specific category at the top or bottom of a ranked cell line list. We use the Benjamini-Hochberg procedure ([303]) to adjust the resulting p-values. We consider results as significant at the significance level $\alpha = 0.05$.
Tables B.26 + B.27 show the result of this analysis for the genes that have been selected by MERIDA.

Table B.26: The table shows the number of times a lowly expressed gene has been enriched
or depleted.

Gene	#Enriched GDSC1	#Enriched GDSC2	#Depleted GDSC1	#Depleted GDSC2
IDH1	8	33	3	1
PIK3CB	65	40	8	0
FRG1	0	0	20	56
CYTH4	0	0	0	0
TJP1	258	156	13	0
PTPRF	205	150	12	1
MAP4K1	0	0	0	0
IREB2	8	0	16	18
MLL2	1	1	0	4
DDX5	3	0	40	4
WIPF1	0	0	0	0
SMARCB1	0	0	20	16
RGS3	0	0	0	0
HDAC9	0	0	0	0
EPC1	1	0	4	13
AHCTF1	1	1	9	2
ADCY1	0	0	0	0
ROBO2	0	0	0	0

ERBB2	0	0	0	0
PPP6C	2	0	38	86
FOXA1	0	0	0	0
TBX3	0	0	0	0
KAT6B	0	0	5	3
SOX9	0	0	0	0
JMY	0	0	3	0
GNG2	0	0	0	0
VHL	6	0	35	46
ACO1	44	63	8	0
HDAC3	0	0	1	0
SFPQ	0	0	2	14
NCKAP1	239	156	10	0
FKBP5	0	0	0	0
NTRK2	0	0	0	0
PPP2R5A	16	15	35	3
HSP90AB1	0	0	9	0
XPO1	0	0	9	1
PTGS1	0	0	0	0
CNOT3	0	0	1	6
STARD13	0	0	0	0
IDH2	6	0	0	0
FMR1	1	5	2	1
ZNF638	0	0	16	10
ADAM10	10	0	24	7
TCF4	0	0	0	0
TCF7L2	0	0	0	0
CDC73	1	0	0	0
PBRM1	3	0	62	42
ARHGEF6	0	0	0	0
CCND1	33	42	14	0
BCL11A	0	0	0	0
KLF6	111	119	7	0
CLASP2	0	0	14	11

CASP8	32	58	30	1
FBXW7	0	0	2	2
HLA-B	34	65	66	5
MYC	16	12	85	7
DNMT3A	0	0	0	0
MFNG	0	0	0	0
RBM10	2	1	67	11
MYH9	15	45	11	2
TCF12	1	0	0	0
UPF3B	0	0	57	38
ARFGEF2	5	9	5	0
ARID2	0	0	56	16
NAP1L1	0	1	52	58
AHNAK	173	144	15	3

Table B.27: The table shows the number of times a highly expressed gene has been enriched
or depleted.

Gene	#Enriched GDSC1	#Enriched GDSC2	#Depleted GDSC1	#Depleted GDSC2
IDH1	0	1	28	98
PIK3CB	11	5	2	1
FRG1	184	139	6	0
CYTH4	205	152	4	0
TJP1	0	0	0	0
PTPRF	0	0	7	0
MAP4K1	191	151	15	3
IREB2	155	123	5	0
MLL2	143	113	2	0
DDX5	14	0	0	0
WIPF1	193	150	13	1
SMARCB1	193	146	9	2
RGS3	24	4	51	43
HDAC9	10	0	1	2
EPC1	177	129	6	1
AHCTF1	53	20	6	0
ADCY1	22	33	58	6
ROBO2	112	117	5	1
ERBB2	20	9	57	48
PPP6C	38	20	8	1
FOXA1	1	6	184	109
TBX3	3	0	56	30
KAT6B	153	127	7	2
SOX9	0	0	24	16
JMY	40	97	4	0
GNG2	138	122	8	1
VHL	128	140	6	0
ACO1	11	1	70	50
HDAC3	26	17	1	0
SFPQ	80	94	6	0

NCKAP1	0	0	0	0
FKBP5	192	150	10	2
NTRK2	0	0	24	3
PPP2R5A	10	4	2	0
HSP90AB1	6	33	3	0
XPO1	17	12	5	1
PTGS1	37	10	0	1
CNOT3	15	30	1	0
STARD13	28	23	0	0
IDH2	33	58	12	2
FMR1	2	1	20	2
ZNF638	62	44	6	0
ADAM10	2	0	27	21
TCF4	146	131	14	1
TCF7L2	8	2	38	55
CDC73	3	1	9	3
PBRM1	181	119	6	1
ARHGEF6	237	151	9	0
CCND1	8	2	5	1
BCL11A	200	146	8	0
KLF6	6	2	59	41
CLASP2	133	120	10	2
CASP8	105	89	0	0
FBXW7	147	137	11	0
HLA-B	0	0	0	0
MYC	0	0	0	0
DNMT3A	135	126	11	2
MFNG	224	151	13	1
RBM10	76	60	17	2
MYH9	18	1	120	70
TCF12	106	92	20	5
UPF3B	72	93	16	2
ARFGEF2	0	0	107	116
ARID2	162	129	7	1

NAP1L1	153	72	2	0
AHNAK	2	0	23	30

Output rules

Table B.28: MERIDA's output rules for each drug (best model selected based on Youden's J)

Drug	Setting	Weight function	Rule size	Model rule
AZD8055	1	linear	5	FKBP5 high expression ∨ VHL high expression ∨ ROBO2 high expression ∨ TRIO low expression ∨ NCKAP1 low expression
AZD8055	1	quadratic	4	ALK Gain of function ∨ FKBP5 high expression ∨ VHL high expression ∨ NCKAP1 low expression
AZD8055	1 (iteration 4)	quadratic	7	(ALK Gain of function ∨ FKBP5 high expression ∨ VHL high exprression ∨ TRIO low expression ∨ NCKAP1 low expression) ∧¬ (TBX3 high expression ∨ PSME3 low expression)
AZD8055	1	cubic	4	ALK Gain of function ∨ FKBP5 high expression ∨ VHL high expression ∨ NCKAP1 low expression
AZD8055	1 (iteration 4)	cubic	7	(ALK Gain of function ∨ FKBP5 high expression ∨ VHL high expression ∨ PTEN high expression ∨ NCKAP1 low expression) ∧¬ (RBBP7 low expression ∨ HDAC3 low expression)
CX-5461	1	linear	2	CLASP2 high expression ∨ TJP1 low expression

CX-5461	2	linear	2 (5)*	known sensitivity determinants ∨ CLASP2 high expression ∨ TJP1 low expression
CX-5461	3	linear	2	CLASP2 high expression ∨ TJP1 low expression
CX-5461	1	quadratic	2	TCF4 high expression ∨ TJP1 low expression
CX-5461	2	quadratic	2 (5)*	known sensitivity determinants ∨ CLASP2 high expression ∨ TJP1 low expression
CX-5461	3	quadratic	2	TCF4 high expression ∨ TJP1 low expression
CX-5461	1	cubic	4	CLASP2 high expression ∨ PRPF8 high expression ∨ LIMA1 low expression ∨ TJP1 low expression
CX-5461	2	cubic	2 (5)*	known sensitivity determinants ∨ CLASP2 high expression ∨ TJP1 low expression
CX-5461	3	cubic	4	CLASP2 high expression ∨ PRPF8 high expression ∨ LIMA1 low expression ∨ TJP1 low expression
Dactolisib	1	linear	2	MGA Unknown mutation ∨ TJP1 low expression
Dactolisib	2	linear	2 (3)*	MGA Unknown mutation ∨ known sensitivity determinants ∨ TJP1 low expression
Dactolisib	3	linear	2	MGA Unknown mutation ∨ TJP1 low expression
Dactolisib	1	quadratic	2	MGA Unknown mutation ∨ TJP1 low expression

Dactolisib	2	quadratic	2 (3)*	MGA Unknown mutation ∨ known sensitivity determinants ∨ TJP1 low expression
Dactolisib	3	quadratic	2	MGA Unknown mutation ∨ TJP1 low expression
Dactolisib	1	cubic	2	MGA Unknown mutation ∨ TJP1 low expression
Dactolisib	2	cubic	2 (3)*	MGA Unknown mutation ∨ known sensitivity determinants ∨ TJP1 low expression
Dactolisib	3	cubic	2	MGA Unknown mutation ∨ TJP1 low expression
Niraparib	1	linear	5	ARID4B high expression ∨ ARID2 high expression ∨ KLF6 low expression ∨ TJP1 low expression ∨ ARFGEF2 low expression
Niraparib	1	quadratic	4	ARID2 high expression ∨ CASP8 low expression ∨ KLF6 low expression ∨ TJP1 low expression
Niraparib	1 (iteration 4)	quadratic	7	ARID4B high expression ∨ WASF3 high expression ∨ ARID2 high expression ∨ CASP8 low expression ∨ KLF6 low expression ∨ CTTN low expression ∨ TJP1 low expression
Niraparib	1	cubic	4	ARID2 high expression ∨ CASP8 low expression ∨ KLF6 low expression ∨ TJP1 low expression

Niraparib	1 (iteration 4)	cubic	10	(ARID4B high expression $\vee$ WASF3 high expression $\vee$ ARID2 high expression $\vee$ CASP8 low expression $\vee$ KLF6 low expression $\vee$ CTTN low expression $\vee$ TJP1 low expression $\vee$ ARNTL low expression) $\wedge\neg$ (FAT2 high expression $\vee$ ATF1 low expression)
NSC319726	1	linear	5	CDC73 high expression $\vee$ TCF12 high expression $\vee$ TCF4 high expression $\vee$ TJP1 low expression $\vee$ ADAM10 low expression
NSC319726	1	quadratic	4	TCF12 high expression $\vee$ ROBO2 high expression $\vee$ TJP1 low expression $\vee$ ADAM10 low expression
NSC319726	1 (iteration 4)	quadratic	9	(TCF12 high expression $\vee$ TGFBR2 high expression $\vee$ IDH2 high expression $\vee$ ROBO2 high expression $\vee$ KLF6 low expression $\vee$ TJP1 low expression $\vee$ ADAM10 low expression $\vee$ RB1 low expression) $\wedge\neg$ RHOA low expression
NSC319726	1	cubic	3	TCF12 high expression $\vee$ ROBO2 high expression $\vee$ TJP1 low expression

NSC319726	1 (iteration 4)	cubic	6	TCF12 high expression $\vee$ TGFBR2 high expression $\vee$ IDH2 high expression $\vee$ ROBO2 high expression $\vee$ TJP1 low expression $\vee$ ADAM10 low expression
Omipalisib	1	linear	2	PIK3CA Gain of function $\vee$ TJP1 low expression
Omipalisib	1	quadratic	2	AHCTF1 high expression $\vee$ TJP1 low expression
Omipalisib	1 (iteration 4)	quadratic	5	(SH2B3 high expression $\vee$ AHCTF1 high expression $\vee$ STAG2 low expression $\vee$ TJP1 low expression) $\wedge \neg$ BCLAF1 low expression
Omipalisib	1	cubic	4	STAG2 Loss of function $\vee$ SH2B3 high expression $\vee$ AHCTF1 high expression $\vee$ TJP1 low expression
Omipalisib	1 (iteration 4)	cubic	11	(TP53BP1 Loss of function $\vee$ PPP2R1A Unknown mutation $\vee$ STAG2 Loss of function $\vee$ CCNE1 CNV gain $\vee$ SH2B3 high expression $\vee$ AHCTF1 high expression $\vee$ TJP1 low expression $\vee$ ARID1A low expression $\vee$ PTPRF low expression) $\wedge \neg$ (NTN4 Unknown mutation $\vee$ TAOK2 low expression)
Rapamycin	1	linear	2	MAP4K1 high expression $\vee$ PTPRF low expression
Rapamycin	2	linear	1 (42)*	known sensitivity determinants $\vee$ TJP1 low expression

Rapamycin	3	linear	2	MAP4K1 high expression ∨ PTPRF low expression
Rapamycin	1	quadratic	2	MAP4K1 high expression ∨ PTPRF low expression
Rapamycin	2	quadratic	2 (43)*	known sensitivity determinants ∨ NCKAP1 low expression ∨ TJP1 low expression
Rapamycin	3	quadratic	5	(STK4 Unknown mutation ∨ PRKAR1A Unknown mutation ∨ MAP4K1 high expression ∨ PTPRF low expression) ∧¬ ARFGEF2 Unknown mutation
Rapamycin	1	cubic	4	(TBL1XR1 Unknown mutation ∨ CYTH4 high expression ∨ MAP4K1 high expression) ∧¬ WT1 high expression
Rapamycin	2	cubic	2 (43)*	known sensitivity determinants ∨ NCKAP1 low expression ∨ TJP1 low expression
Rapamycin	3	cubic	1	MAP4K1 high expression
Talazoparib	1	linear	3	TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression
Talazoparib	2	linear	2 (5)*	known sensitivity determinants ∨ TCF12 high expression ∨ TJP1 low expression
Talazoparib	3	linear	3	TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression
Talazoparib	1	quadratic	3	TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression

Talazoparib	2	quadratic	3 (6)*	known sensitivity determinants ∨ TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression
Talazoparib	3	quadratic	3	TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression
Talazoparib	1	cubic	3	TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression
Talazoparib	2	cubic	3 (6)*	known sensitivity determinants ∨ TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression
Talazoparib	3	cubic	3	TCF12 high expression ∨ TJP1 low expression ∨ HLA-B low expression
Temsirolimus	1	linear	5	PTEN Loss of function ∨ SMARCB1 high expression ∨ WIPF1 high expression ∨ PRRX1 high expression ∨ DHX9 low expression
Temsirolimus	2	linear	1 (16)*	known sensitivity determinants ∨ TJP1 low expression
Temsirolimus	3	linear	5	PTEN Loss of function ∨ SMARCB1 high expression ∨ WIPF1 high expression ∨ PRRX1 high expression ∨ DHX9 low expression
Temsirolimus	1	quadratic	5	PTEN Loss of function ∨ MMP2 high expression ∨ SMARCB1 high expression ∨ WIPF1 high expression ∨ DHX9 low expression

Temsirolimus	2	quadratic	1 (16)*	known sensitivity determinants ∨ TJP1 low expression
Temsirolimus	3	quadratic	5	PTEN Loss of function ∨ MMP2 high expression ∨ SMARCB1 high expression ∨ WIPF1 high expression ∨ DHX9 low expression
Temsirolimus	1	cubic	4	PTEN Loss of function ∨ MMP2 high expression ∨ SMARCB1 high expression ∨ WIPF1 high expression
Temsirolimus	2	cubic	3 (18)*	(PTEN Loss of function ∨ known sensitivity determinants ∨ MAP4K1 high expression) ∧¬ MAGI2 Unknown mutation
Temsirolimus	3	cubic	4	PTEN Loss of function ∨ MMP2 high expression ∨ SMARCB1 high expression ∨ WIPF1 high expression
Voxtalisib	1	linear	1	TJP1 low expression
Voxtalisib	1	quadratic	5	BRAF Gain of function ∨ PPP2R5A high expression ∨ XPO1 high expression ∨ BCL11A high expression ∨ CSNK1G3 low expression

Voxtalisib	1 (iteration 4)	quadratic	13	(BRAF Gain of function ∨ CDK12 Unknown mutation ∨ TGFBR1 CNV gain ∨ PPP2R5A high expression ∨ CNOT4 high expression ∨ XPO1 high expression ∨ BCL11A high expression ∨ CSNK1G3 low expression) ∧ ¬ (CHD6 Unknown mutation ∨ HDAC9 Unknown mutation ∨ CAT Unknown mutation ∨ SMURF2 high expression ∨ CAST low expression)
Voxtalisib	1	cubic	6	(MITF CNV gain ∨ PPP2R5A high expression ∨ XPO1 high expression ∨ BCL11A high expression ∨ CSNK1G3 low expression) ∧¬ CHD6 Unknown mutation
Voxtalisib	1 (iteration 4)	cubic	10	(MTOR neutral mutations ∨ MITF CNV gain ∨ PPP2R5A high expression ∨ XPO1 high expression ∨ BCL11A high expression ∨ PIK3C2B low expression ∨ CSNK1G3 low expression) ∧¬ (CHD6 Unknown mutation ∨ NCF2 high expression ∨ PIK3R3 high expression)

* Known sensitivity determinants for the respective drugs can be found in Tables B.1 - B.11. The provided rule size denotes the number of terms without known sensitivity determinants, while the number in brackets includes them.

Table B.29: LOBICO's output rules for each drug (best model selected based on Youden's J)

Drug	Setting	Weight function	Rule size	Model rule
AZD8055	1	linear	1	TJP1 low expression
AZD8055	1	quadratic	1	NCKAP1 low expression
AZD8055	1	cubic	1	NCKAP1 low expression
CX-5461	3	linear	2	TJP1 low expression $\vee$ CLASP2 high expression
CX-5461	3	quadratic	2	TJP1 low expression $\vee$ TCF4 high expression
CX-5461	3	cubic	2	TJP1 low expression $\vee$ TCF4 high expression
Dactolisib	3	linear	2	MGA Unknown mutation $\vee$ TJP1 low expression
Dactolisib	3	quadratic	2	MGA Unknown mutation $\vee$ TJP1 low expression
Dactolisib	3	cubic	2	MGA Unknown mutation $\vee$ TJP1 low expression
Niraparib	1	linear	2	TJP1 low expression $\vee$ CASP8 low expression
Niraparib	1	quadratic	2	TJP1 low expression $\vee$ KLF6 low expression
Niraparib	1	cubic	2	TJP1 low expression $\vee$ KLF6 low expression
NSC319726	1	linear	2	TJP1 low expression $\vee$ TCF12 high expression
NSC319726	1	quadratic	2	TJP1 low expression $\vee$ TCF12 high expression
NSC319726	1	cubic	2	TJP1 low expression $\vee$ TCF12 high expression
Omipalisib	1	linear	2	PIK3CA Gain of function $\vee$ TJP1 low expression

Omipalisib	1	quadratic	2	AHCTF1 high expression ∨ TJP1 low expression
Omipalisib	1	cubic	1	TJP1 low expression
Rapamycin	3	linear	2	PTPRF low expression ∨ MAP4K1 high expression
Rapamycin	3	quadratic	2	MAP4K1 high expression ∧¬ PTPRU Unknown mutation
Rapamycin	3	cubic	1	MAP4K1 high expression
Talazoparib	3	linear	2	TCF12 high expression ∨ TJP1 low expression
Talazoparib	3	quadratic	2	TCF12 high expression ∨ TJP1 low expression
Talazoparib	3	cubic	2	TCF12 high expression ∨ TJP1 low expression
Temsirolimus	3	linear	1	TJP1 low expression
Temsirolimus	3	quadratic	1	TJP1 low expression
Temsirolimus	3	cubic	2	MAP4K1 high expression ∨ MMP2 high expression
Voxtalisib	1	linear	1	TJP1 low expression
Voxtalisib	1	quadratic	2	SCAI high expression ∨ NCKAP1 low expression
Voxtalisib	1	cubic	2	NCKAP1 low expression ∧¬ MLH1 high expression

C SAURON-RF additional information

Model	Parameter	Value(s)
Feature selection	K	$\{20, 40, 60, 80, 100\}$
	binning	equal width
	bins	6
Boosting Trees	n.trees	100
	interaction.depth	4
	shrinkage	0.1
	bag.fraction	0.5
	distribution	“gaussian”
	cv.folds	5
Random Forests	n_estimators	500
	min_samples_leaf	15
	max_features	$\frac{\#\text{Features}}{3}$ or $\sqrt(\#\text{Features})$ (regression) (classification)
Elastic Net	alpha	$[0, 1]$
	lambda	10^v, $v \in [-2, 2]$
	standardize	TRUE
	CV folds	5
Neural Network	Loss function	MSE
	Optimizer	Adam
	Learning rate	0.001
	# Hidden layers	1, 2, 3
	# Nodes per hidden layer	same as input layer
	Activation function	tanh (none in output layer)
	Weight initialization	Glorot uniform
	Bias initialization	0.01
	Weight regularization	L2
	Bias regularization	none
	Dropout	10%
	Batch size	128
	Epochs	max. 4000 (early stopping)
	Patience	15 epochs
	Data fraction for validation	20%

Table C.30: Summary of all model parameters used to fit the feature selection algorithm, boosting trees, random forests (including SAURON-RF), elastic nets and neural networks for the generation of the results.

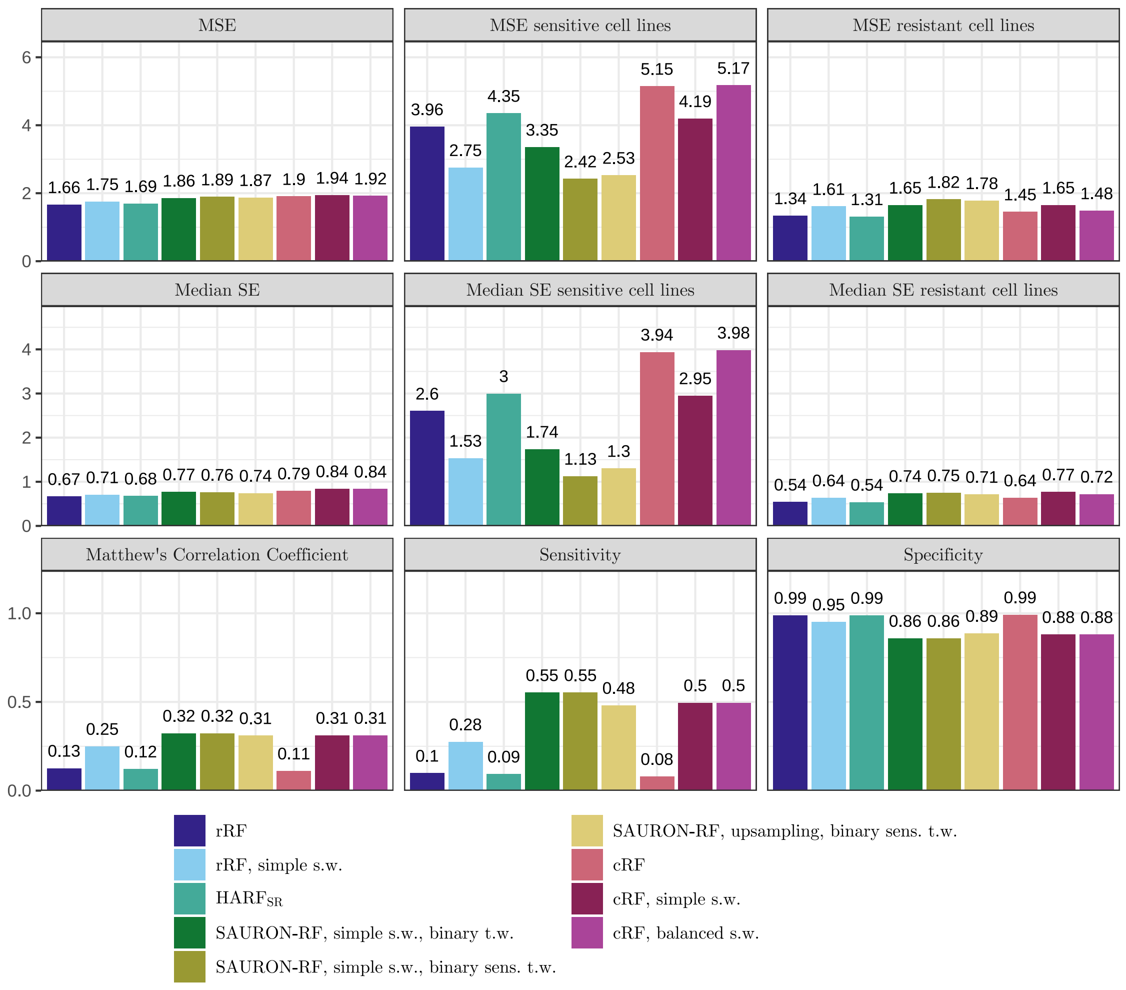


Figure C.32: Random forest test set performance for 40 input features. In this figure, we compare regression random forests, classification random forests, and HARF with our suggested approach SAURON-RF. We show the average test set performance across the 86 different drugs for 40 input features.

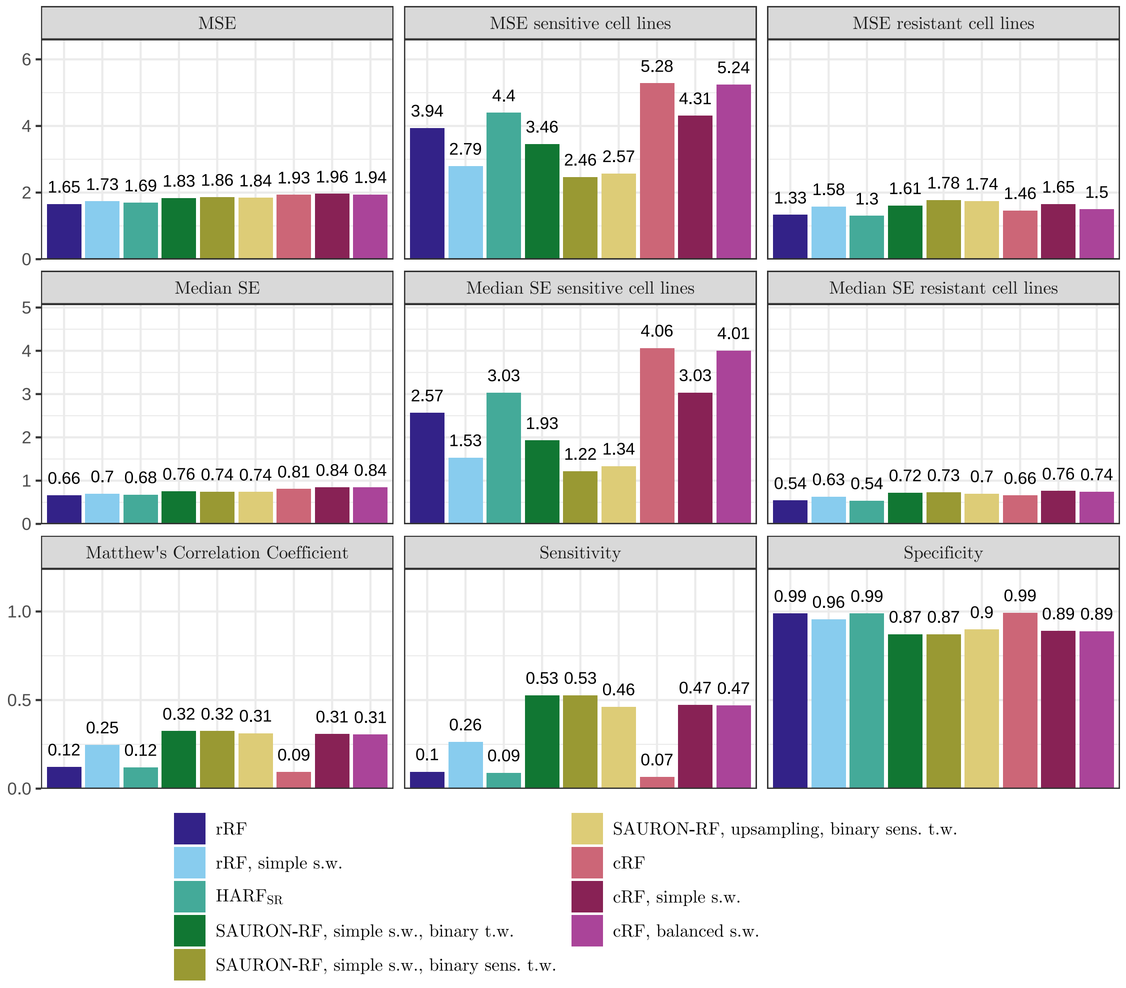


Figure C.33: Random forest test set performance for 60 input features. In this figure, we compare regression random forests, classification random forests, and HARF with our suggested approach SAURON-RF. We show the average test set performance across the 86 different drugs for 60 input features.

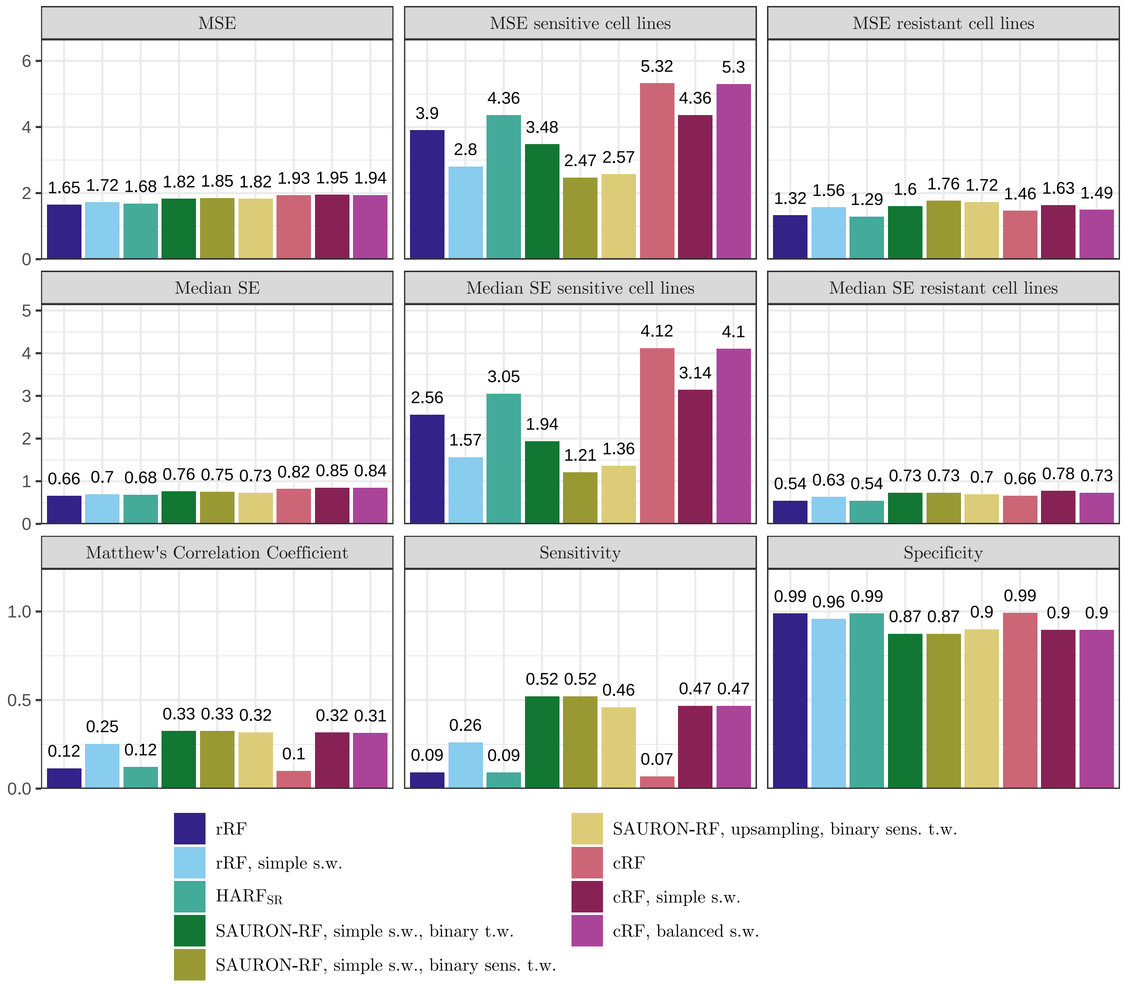


Figure C.34: Random forest test set performance for 80 input features. In this figure, we compare regression random forests, classification random forests, and HARF with our suggested approach SAURON-RF. We show the average test set performance across the 86 different drugs for 80 input features.

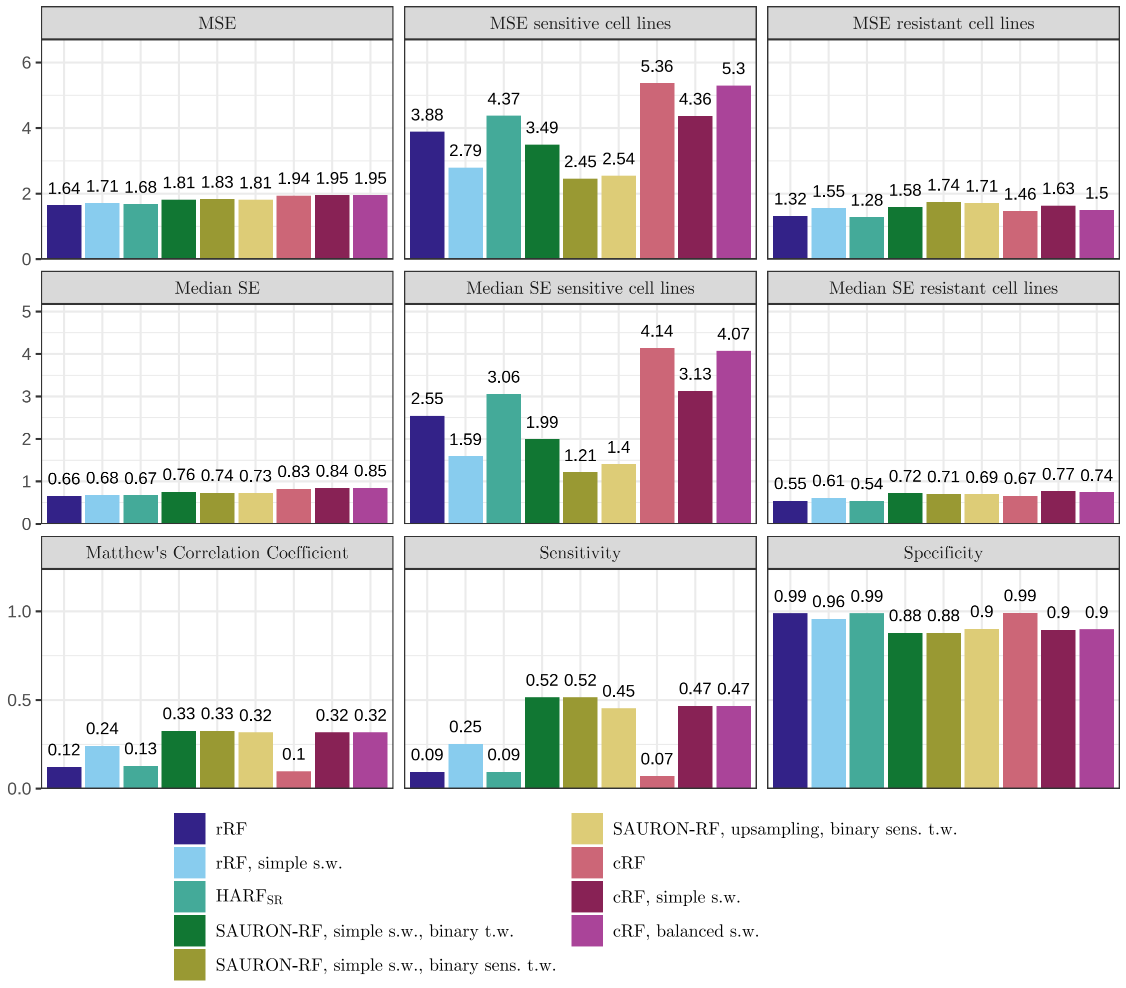


Figure C.35: Random forest test set performance for 100 input features. In this figure, we compare regression random forests, classification random forests, and HARF with our suggested approach SAURON-RF. We show the average test set performance across the 86 different drugs for 100 input features.

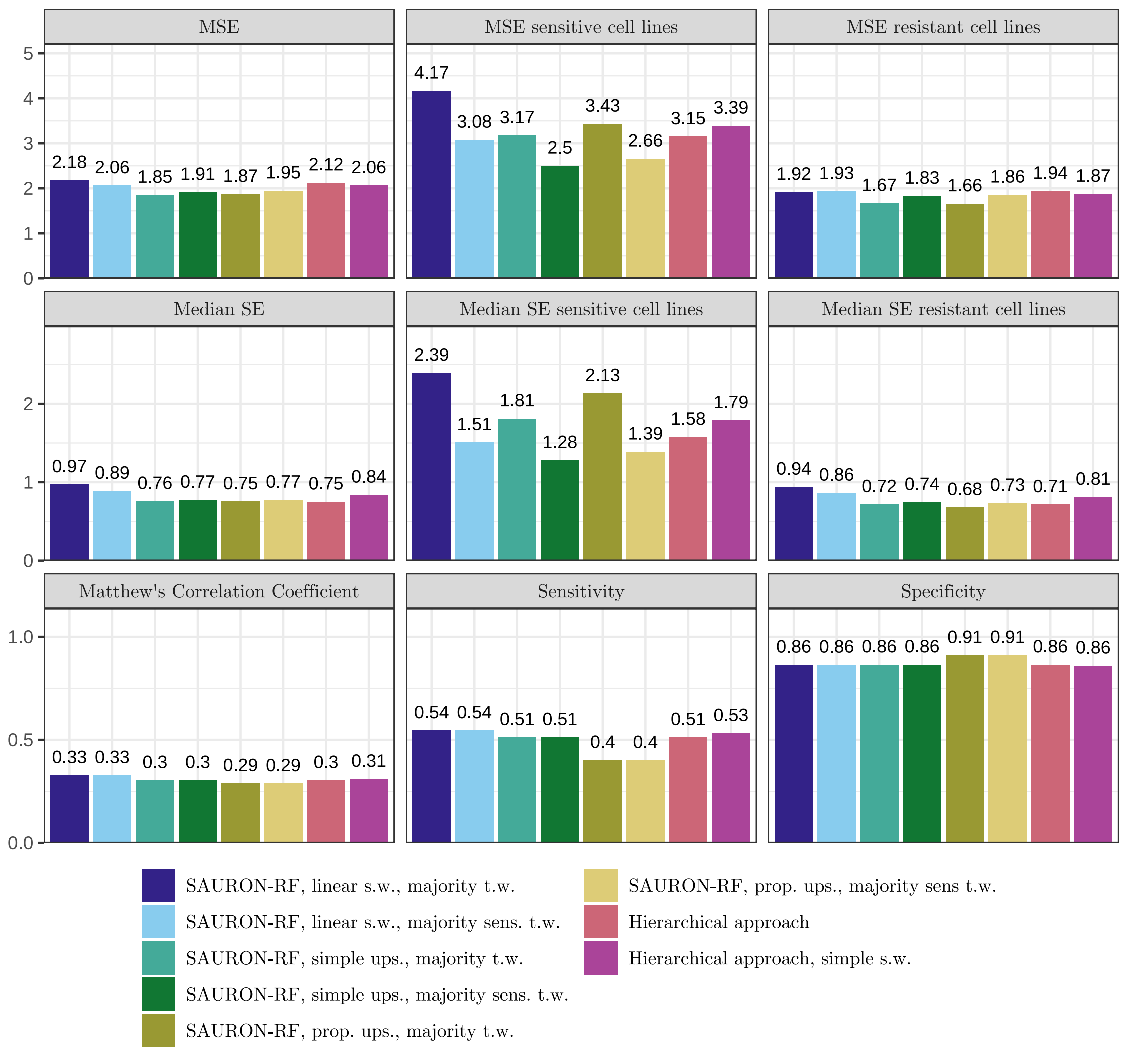


Figure C.36: Test set performance for additional versions of SAURON-RF, including the hierarchical approach. We show the average test set performance across the 86 different drugs for 20 input features.

Drug: ABT737
Target(s): BCL2, BCL-XL, BCL-W, BCL-B, BFL1
Target pathway: Apoptosis regulation

Feature	Feature function (GeneCards [299])	Validated?
TNFRSF12A	involved in extrinsic apopotosis and wound healing regulation	✓ [288]
BCL2	located in the outer mitochondrial membrane and involved in apoptosis	✓ drug target
MIR22HG	involved in wound response	(✓) involved in downregulation of BCL2 ([227, 226])
BLVRB	catalyzes final step of heme metabolism	(✓) low expression associated with obatoclax sensitivity (also BCL2 inhibitor) ([89])
IDH2	catalyzes the oxidative decarboxylation of isocitrate to 2-oxoglutarat	✓ mutations associated with increased sensitivity ([228, 229])

Table C.31: This table lists the five features with highest importance in the prediction model for ABT737 using our best-performing SAURON-RF version with K=60. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Nutlin-3a(-)
Target(s): MDM2
Target pathway: p53 pathway

Feature	Feature function (GeneCards [299])	Validated?
MDM2	nuclear-localized E3 ubiquitin ligase, which can promote tumour formation	✓ drug target
RPS27L	might be a component of the 40S ribosomal subunit	✓ [231]
DDB2	part of protein complex that is involved in nucleotide excision repair and cellular response to DNA damage in general	(✓) maybe yes, has to do with MDM2 and nucleotide excision repair [289], Nutlin-3a(-) treatment increases DDB2 expression substantially ([232])
CYFIP2	participates in T-cell adhesion and p53-dependent induction of apoptosis	(✓) Nutlin-3a(-) treatment increases CYFIP2 expression substantially ([233])
SDC4	is a transmembrane proteoglycan involved in intracellular signaling	×

Table C.32: This table lists the five features with highest importance in the prediction model for Nutlin-3a(-) using our best-performing SAURON-RF version with K=80. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Irinotecan
Target(s): TOP1
Target pathway: DNA replication

Feature	Feature function (GeneCards [299])	Validated?
SLFN11	involved in tRNA binding, defense response to virus, negative regulation of G1/S transition and replication fork arrest	✓ [291]
SDC4	is a transmembrane proteoglycan involved in intracellular signaling	(✓) high plasma levels of SDC1 related to Irinotecan resistance ([292])
NCKAP1L	transmembrane protein, part of the WAVE complex that regulates cell shape, expressed in haematopoietic cells only	×
DAG1	part of complex that links extracellular matrix to cytoskeleton in skeletal muscle	(✓) combination treatment with PHY906 reduces DAG1 expression compared to other treatments ([290])
CELSR1	it is postulated that the corresponding protein acts as receptor involved in contact-mediated communication	×

Table C.33: This table lists the five features with highest importance in the prediction model for Irinotecan using our best-performing SAURON-RF version with K=100. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Venetoclax
Target(s): BCL2
Target pathway: Apoptosis regulation

Feature	Feature function (GeneCards [299])	Validated?
TNFRSF12A	involved in extrinsic apoptosis and wound healing regulation	×
CTTN	involved in actin cytoskeleton and cell shape regulation	×
P4HA2	involved in collagen synthesis and amino acid metabolism	×
CYSTM1	related to the innate immune system	×
PIP4K2C	kinase	×

Table C.34: This table lists the five features with highest importance in the prediction model for Venetoclax using our best-performing SAURON-RF version with K=80. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Temozolomide
Target(s): DNA alkylating agent
Target pathway: DNA replication

Feature	Feature function (GeneCards [299])	Validated?
IKZF1	transcription factor	✓ [300]
CYR61	besides others involved in cell proliferation, chemotaxis and angiogenesis	✓ [301]
SDC4	involved in intracellular signaling	×
CTDSPL	has phosphatase activity	✓ host gene of miR-26a, whose overexpression correlates with poor treatment prognosis ([302])
ASPHD2	has metal ion binding activity	×

Table C.35: This table lists the five features with highest importance in the prediction model for Temozolomide using our best-performing SAURON-RF version with K=100. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: I-BRD9
Target(s): BRD9
Target pathway: Chromatin other

Feature	Feature function (GeneCards [299])	Validated?
YAP1	transcription factor, tumor suppressor	×
PPIC	protein folding	×
DCBLD2	negative regulation of cell growth	×
DDR1	transferase activity	×
LURAP1L	involved in positive regulation of I-kappaB kinase/NF-kappaB signaling	×

Table C.36: This table lists the five features with highest importance in the prediction model for I-BRD9 using our best-performing SAURON-RF version with K=100. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Cytarabine
Target(s): Antimetabolite
Target pathway: Other

Feature	Feature function (GeneCards [299])	Validated?
PPIC	protein folding	×
TUFT1	structural constituent of tooth enamel	×
LMNA	part of nuclear lamina	×
ZNF22	transcription factor	×
ARHGAP19	has GTPase activator activity	×

Table C.37: This table lists the five features with highest importance in the prediction model for Cytarabine using our best-performing SAURON-RF version with K=20. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Trametinib
Target(s): MEK1, MEK2
Target pathway: ERK MAPK signaling

Feature	Feature function (GeneCards [299])	Validated?
DUSP6	has phosphatase activity	✓ [293]
SPRY2	invovled in kinase binding	✓ [294]
BCAS4	part of BLOC-1 complex and associated with breast cancer	×
ETV4	transcription factor	✓ [295]
SLC22A18	has transporter activity	×

Table C.38: This table lists the five features with highest importance in the prediction model for Trametinib using our best-performing SAURON-RF version with K=80. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Dabrafenib
Target(s): BRAF
Target pathway: ERK MAPK signaling

Feature	Feature function (GeneCards [299])	Validated?
BCL2A1	involved in apoptosis	✓ [296]
ARHGAP15	involved in RHO GTPase regulation	×
CTHRC1	may be involved in wound healing	✓ [297]
FAM89A	-	×
MBP	involved in formation and stabilization of myelin membrane	✓ [298]

Table C.39: This table lists the five features with highest importance in the prediction model for Dabrafenib using our best-performing SAURON-RF version with K=100. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

Drug: Epirubicin
Target(s): Anthracycline
Target pathway: DNA replication

Feature	Feature function (GeneCards [299])	Validated?
SEPT6	involved in actin cytoskeleton organization	×
TM4SF1	involved in cell growth and motility	×
GNA11	involved in transmembrane signaling	×
EPHX1	enzyme	×
PLEKHA6	involved in metabolism	×

Table C.40: This table lists the five features with highest importance in the prediction model for Epirubicin using our best-performing SAURON-RF version with K=80. Additionally, the drug target(s) and target pathway (both derived from the GDSC) are shown.

D Benchmarking additional information

Table D.41: Overview of all hyperparameters that were investigated for the training of neural networks.

Parameter	Value(s)
Loss function	MSE
Optimizer	Adam
Learning rate	0.001 (default)
# Hidden layers	1, 2, 3
# Nodes per layer	input: k, output: 1, hidden: evenly spaced between in- and output
Activation function	tanh, ELU (none in output layer)
Weight initialization	Glorot uniform for tanh activation, He normal for ELU
Bias initialization	0.01
Weight regularization	L2
Bias regularization	none (default)
Dropout	10%, 30%
Batch size	64
Epochs	max. 4000 (early stopping using 20% of samples as validation data)
Patience	15 epochs

Table D.42: Overview of all hyperparameters that were used for the training of autoencoders.

Parameter	Value(s)
Loss function	MSE
Optimizer	Adam
Learning rate	0.001 (default for Adam)
# Nodes per layer	input: 17,419; hidden (encoder): 3,484 and 697; bottleneck: k; hidden (decoder): 697 and 3,484; output: 17,419
Activation function	RELU (none in last encoder layer)
Weight initialization	Glorot uniform (default)
Bias initialization	0 (default)
Weight regularization	none (default)
Bias regularization	none (default)
Dropout	none (default)
Batch size	64
Epochs	max. 100 (early stopping using 20% of samples as validation data)
Patience	5 epochs

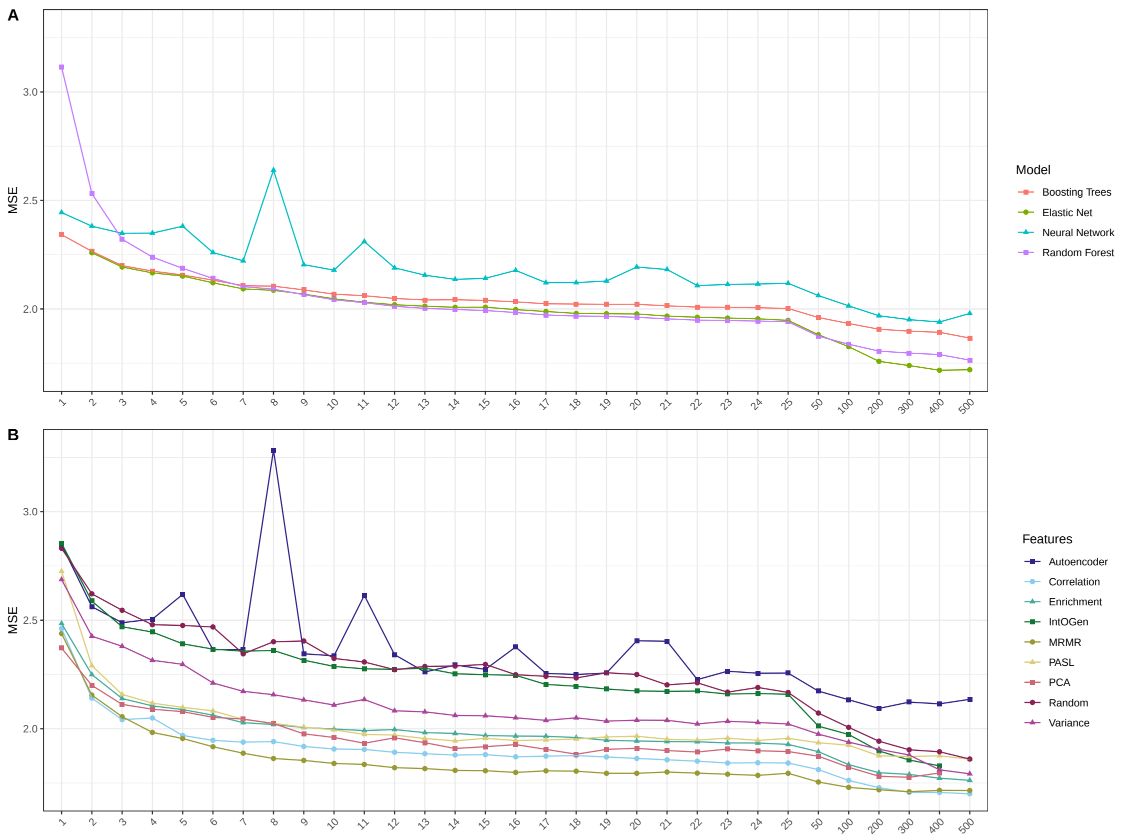


Figure D.37: Average test MSEs for the 50 drugs with most cell lines. In this figure, the average test set mean-squared error (MSE) is depicted. In Figure A, we show the mean-squared error averaged across all drugs and DR techniques, yielding one best-performing ML model for each investigated k. To generate Figure B, we averaged across all drugs and ML methods, resulting in one best performing DR technique per k. .

E *Reliable* SAURON-RF additional information

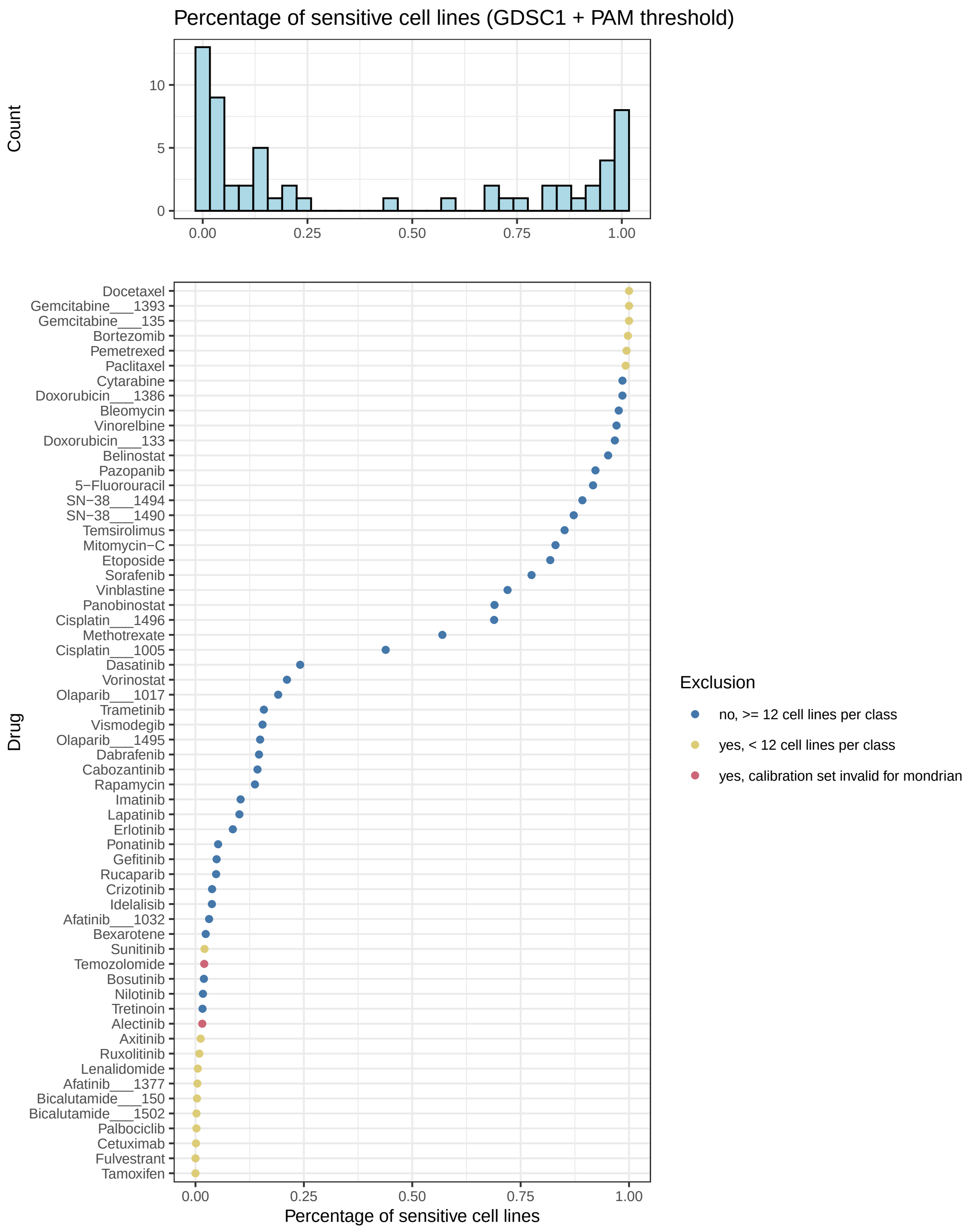


Figure E.38: Percentage of sensitive cell lines of binarized CMax viability in GDSC1. The upper row of this figure shows a histogram for the percentage of sensitive cell lines across all available drugs. The lower row depicts the corresponding percentage for each drug. Moreover, the color of the points indicates which drugs had to be excluded for the analysis and why.

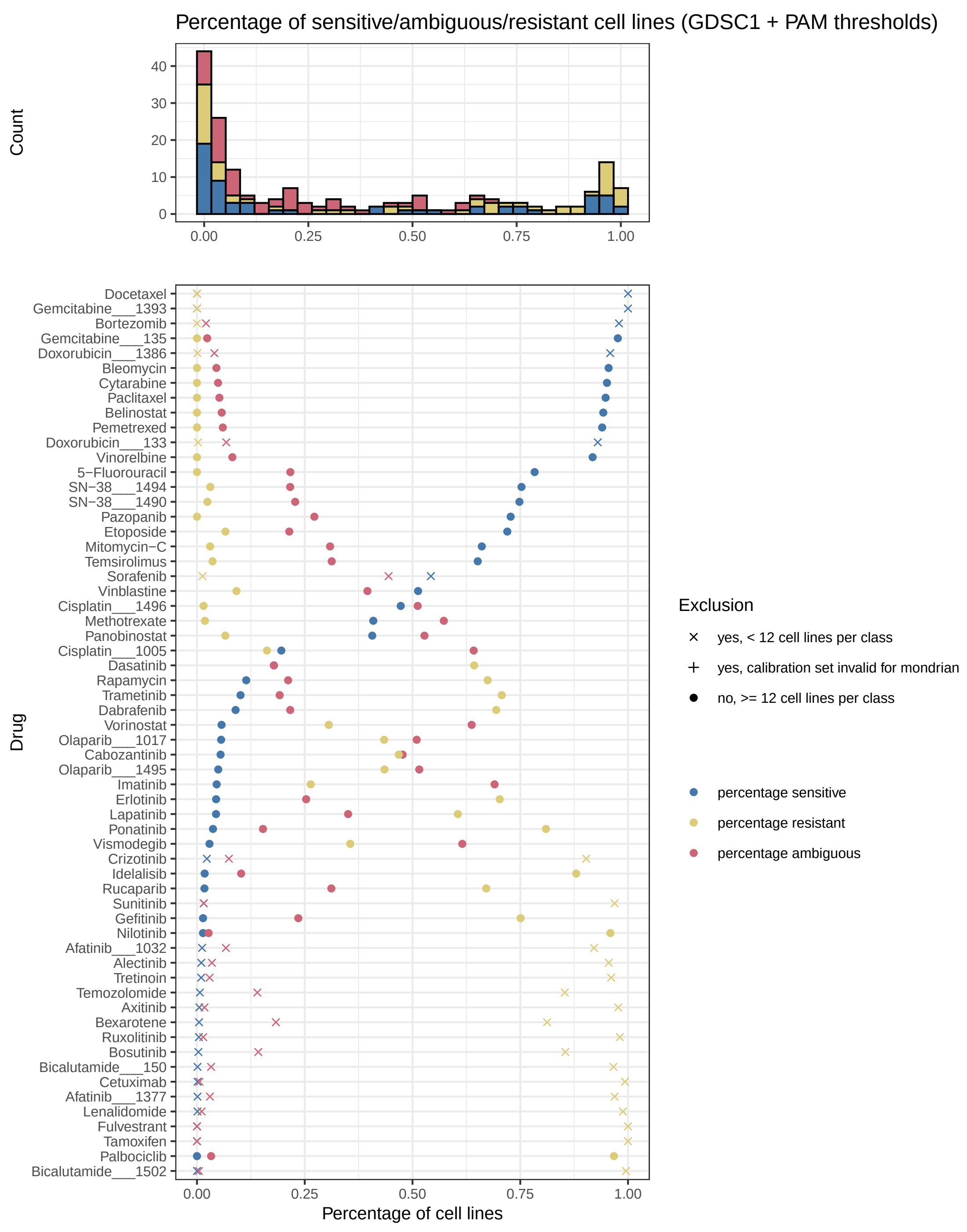


Figure E.39: Percentage of sensitive, ambiguous, and resistant cell lines of ternary CMax viability in GDSC1. The upper row of this figure shows a histogram for the percentage of sensitive, ambiguous and resistant cell lines across all available drugs. The lower row depicts the corresponding percentages for each drug. Moreover, the shape of the points indicates which drugs had to be excluded for the analysis and why.

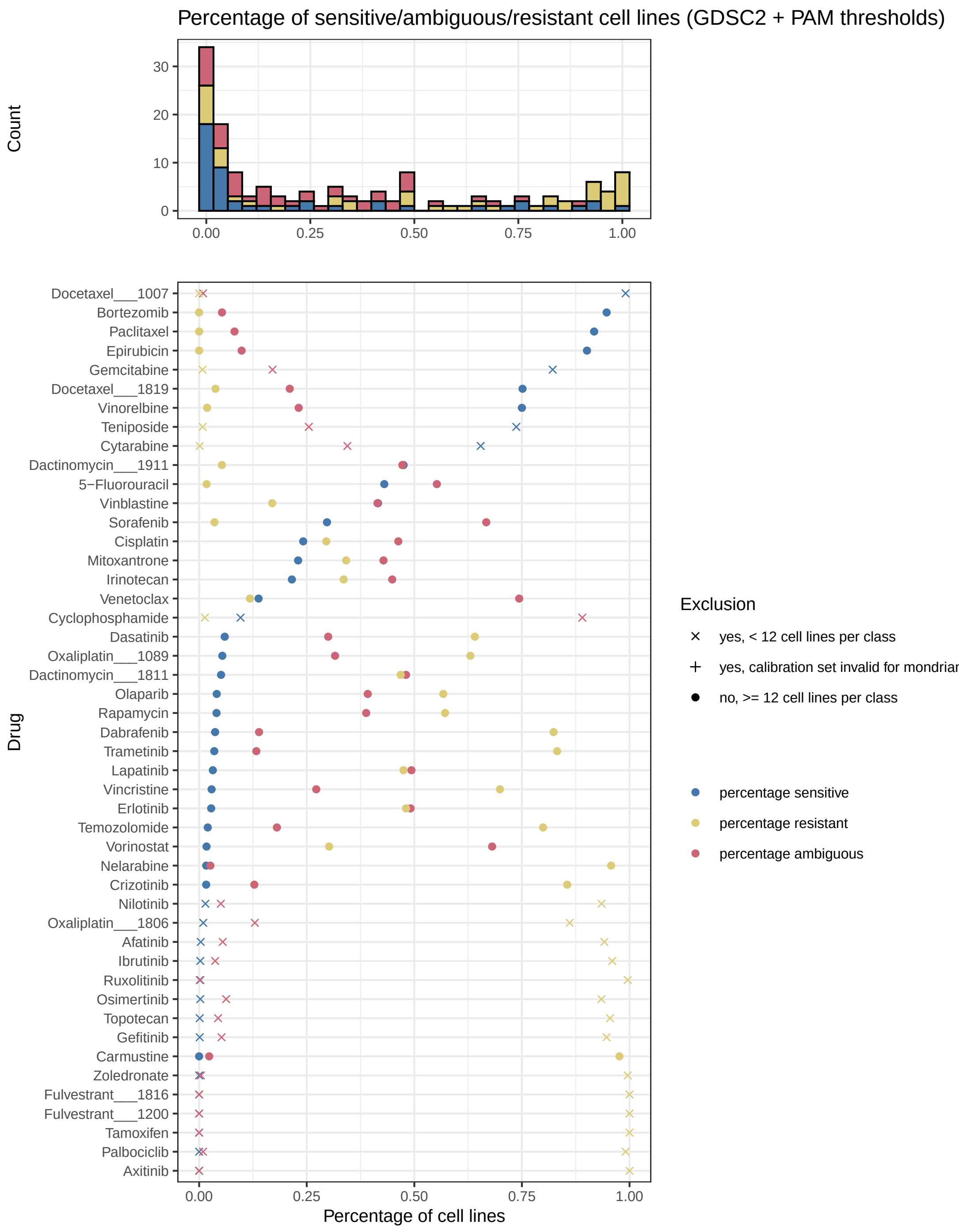
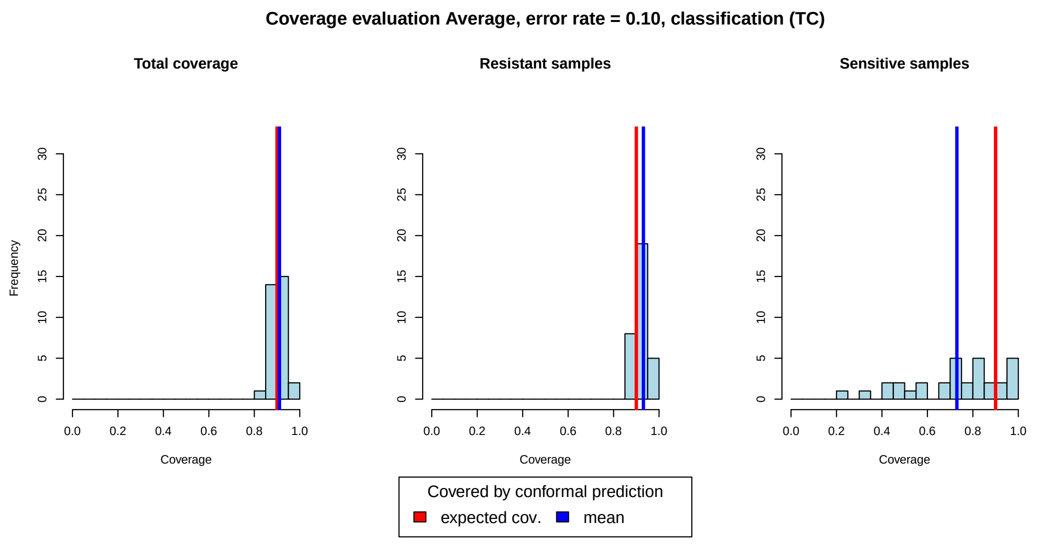
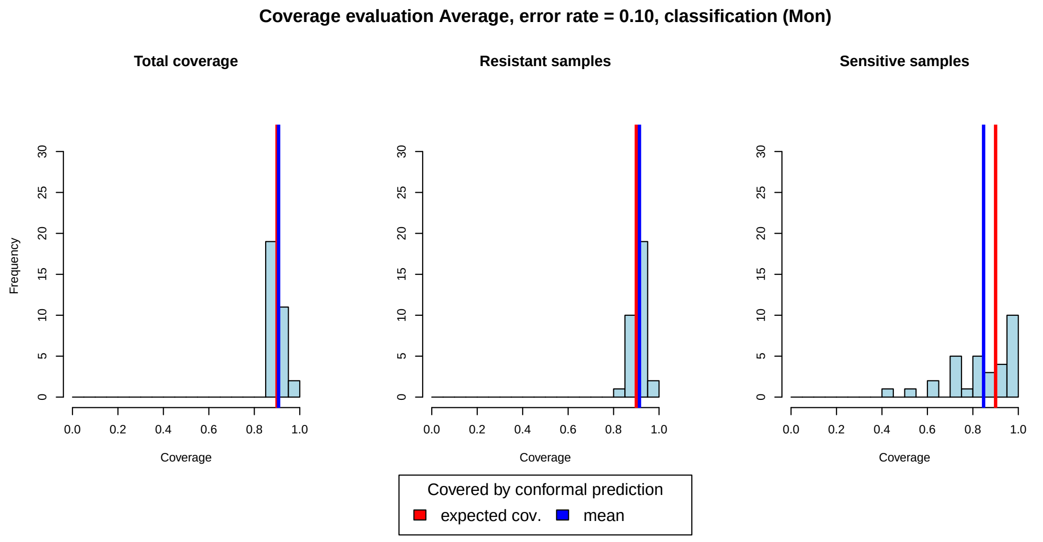


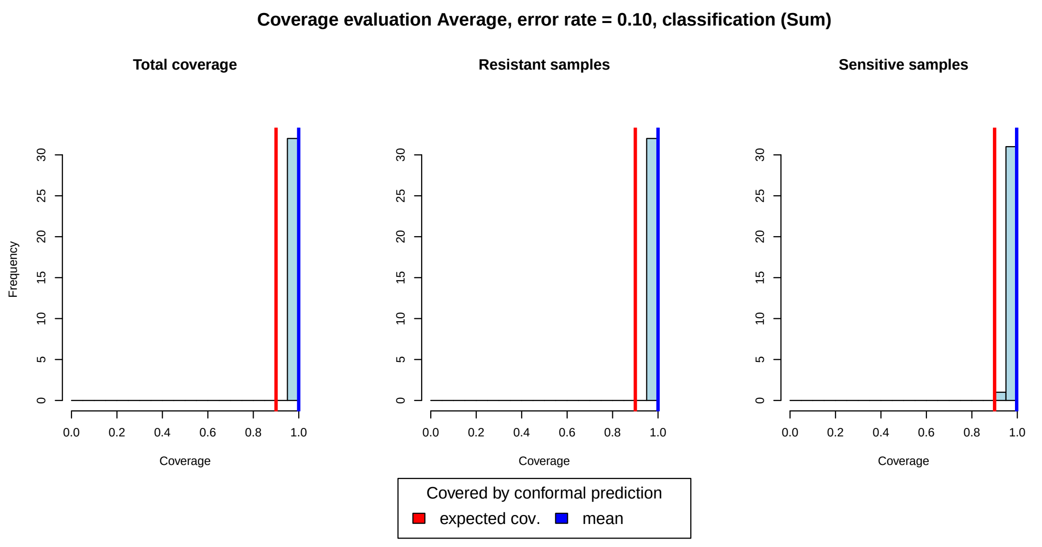
Figure E.40: Percentage of sensitive, ambiguous, and resistant cell lines of ternary CMax viability in GDSC2. The upper row of this figure shows a histogram for the percentage of sensitive, ambiguous and resistant cell lines across all available drugs. The lower row depicts the corresponding percentages for each drug. Moreover, the shape of the points indicates which drugs had to be excluded for the analysis and why.



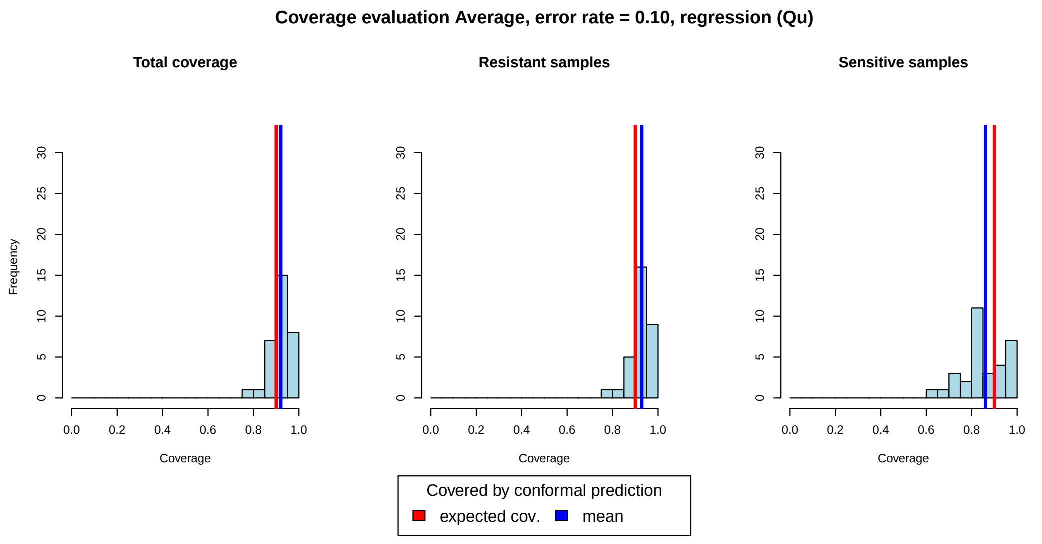
(a) True-class score



(b) Mondrian score

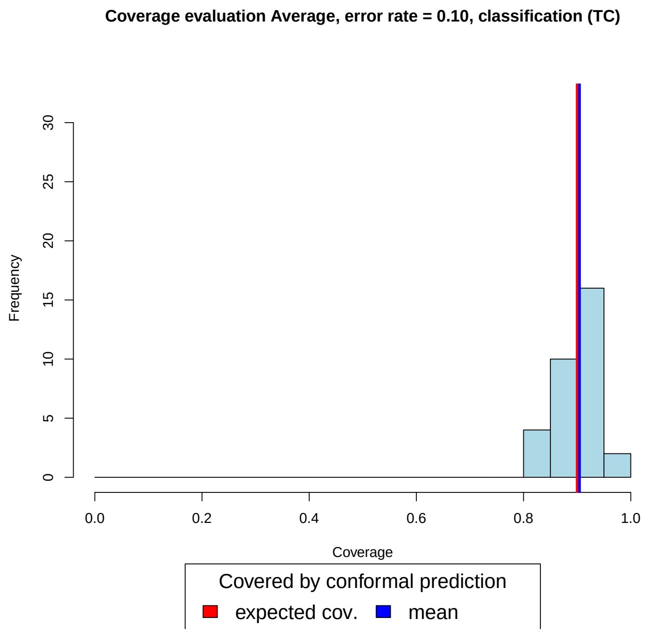


(c) Summation score

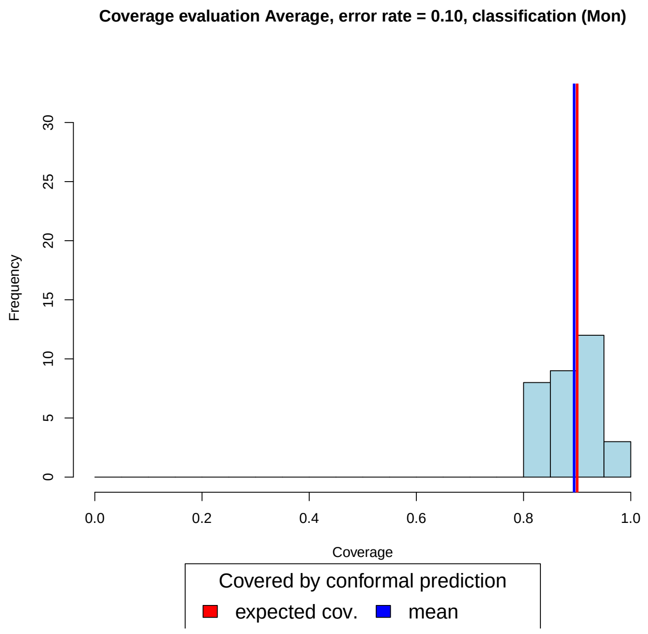


(d) Quantile score

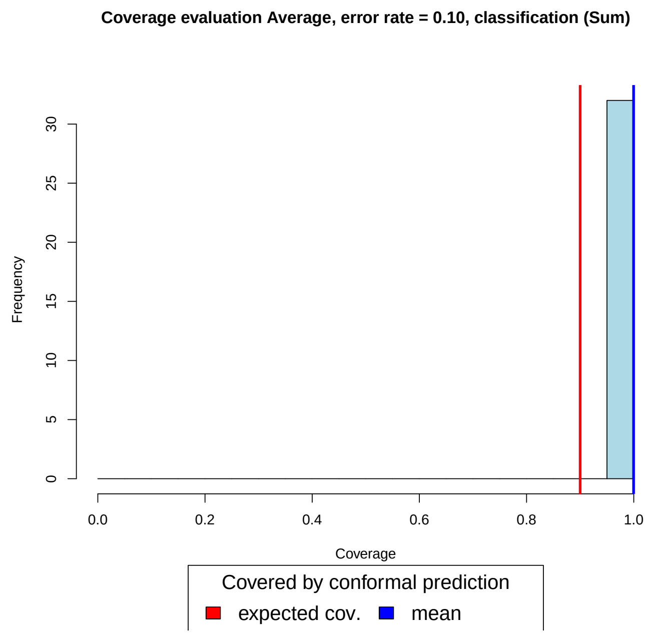
Figure E.41: Coverage evaluation for CP models of 32 drugs from the GDSC2 database trained using IC50 values and a two-class classification setting. This figure depicts histograms of the coverage property across CP models for 32 drugs obtained from the GDSC2 database. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.



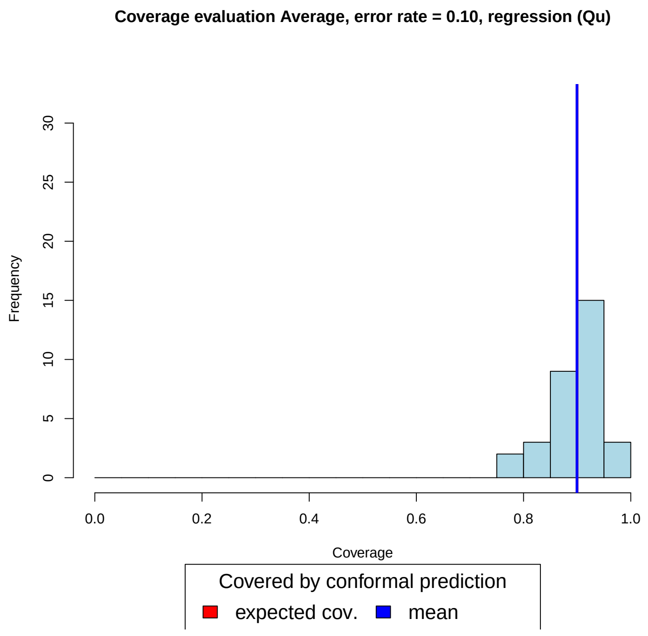
(a) True-class score



(b) Mondrian Score

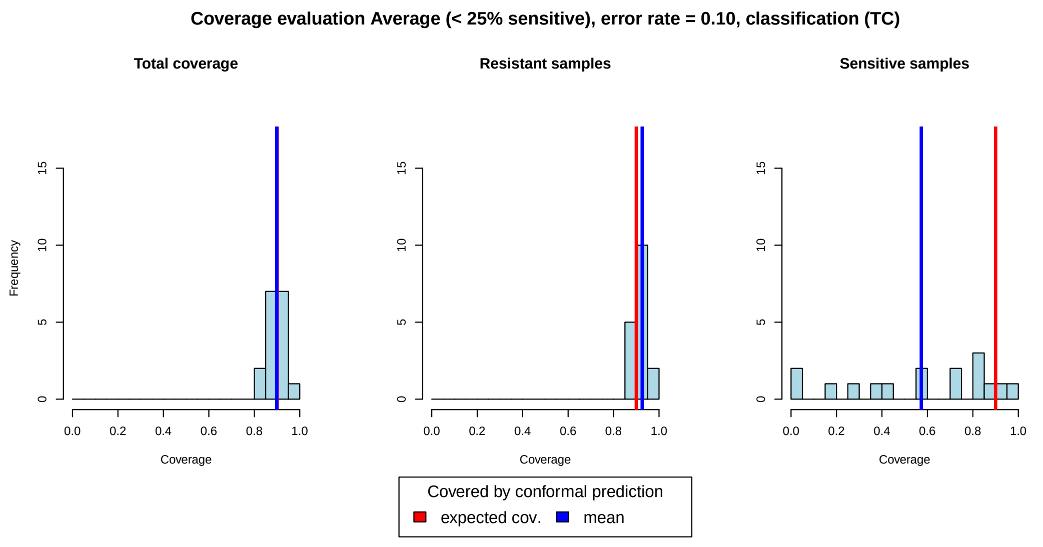


(c) Summation score

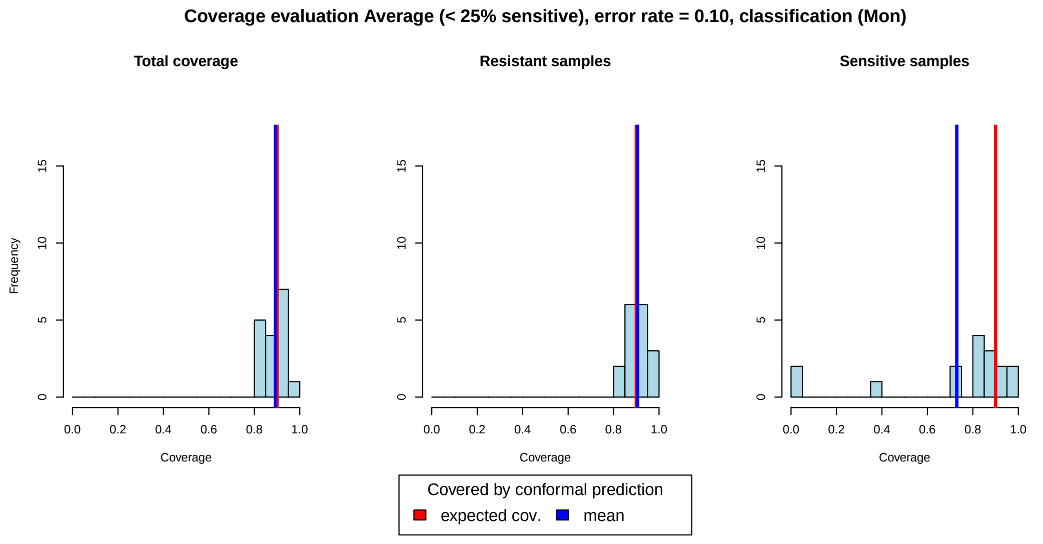


(d) Quantile score

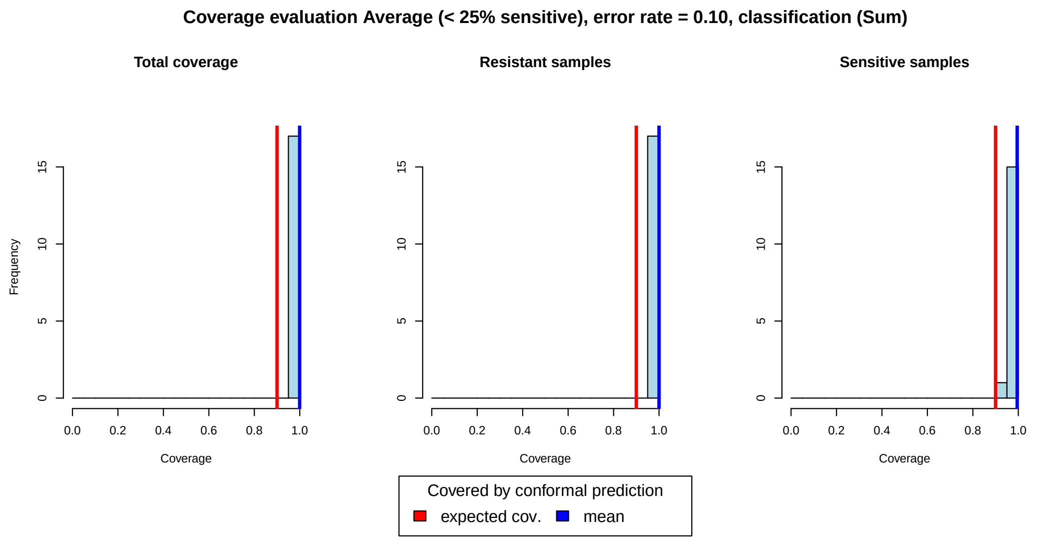
Figure E.42: Coverage evaluation for CP models of 32 drugs from the GDSC2 database trained using CMax viability values and a two-class classification setting. This figure depicts histograms of the coverage property across CP models for 32 drugs obtained from the GDSC2 database. The coverage is computed as the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.



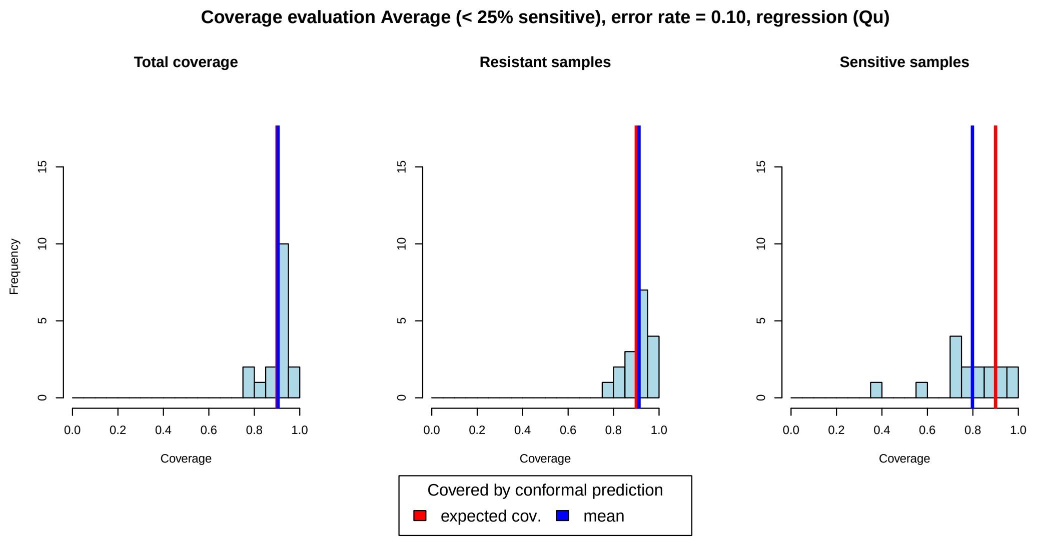
(a) True-class score



(b) Mondrian score

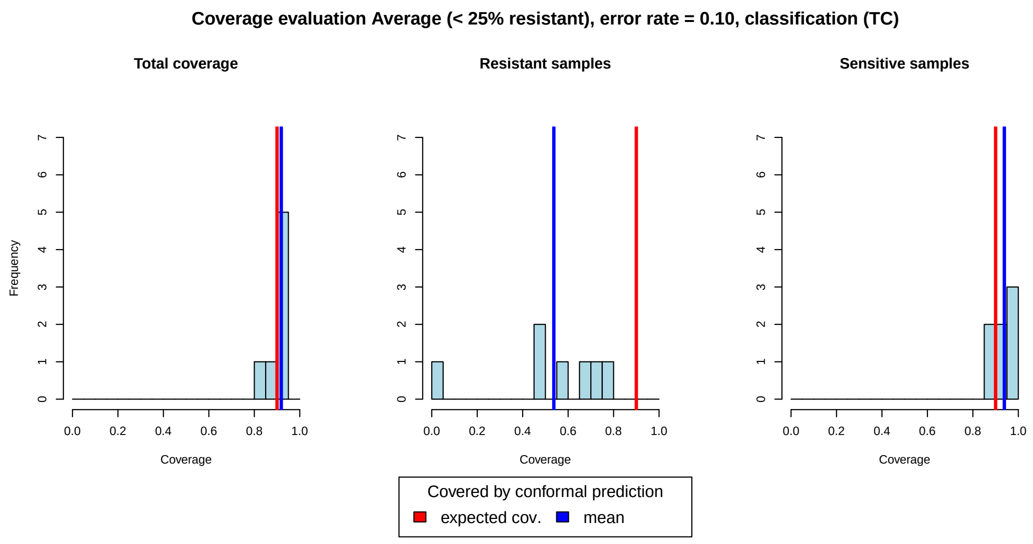


(c) Summation score

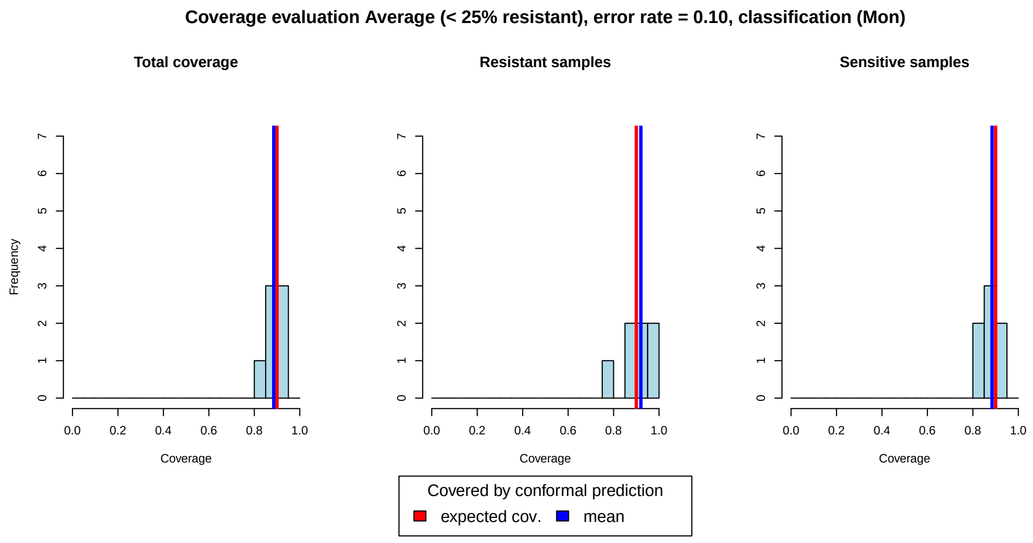


(d) Quantile score

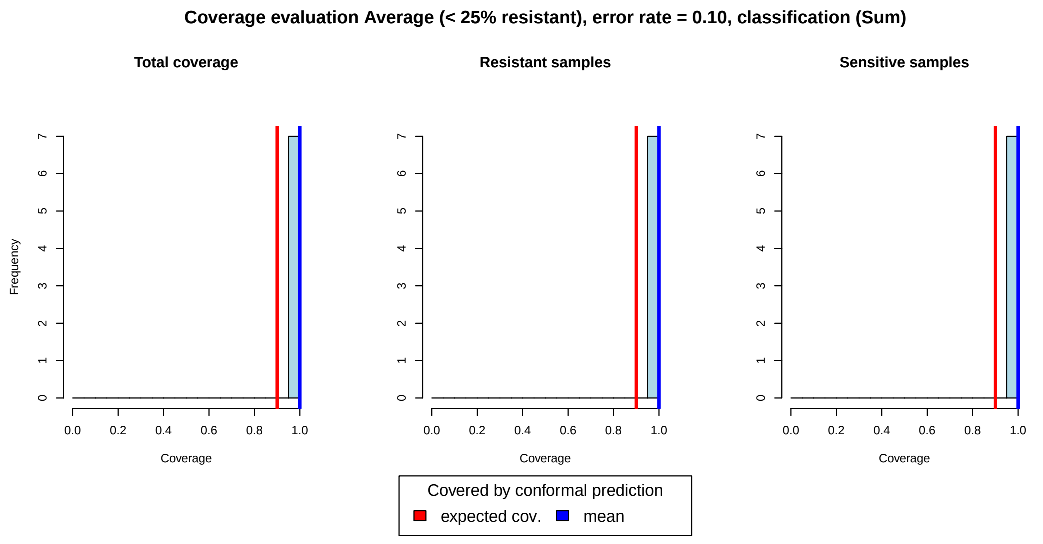
Figure E.43: Coverage evaluation for CP models of 17 drugs from the GDSC2 database trained using CMax viability values and a two-class classification setting with underrepresentation of the sensitive class. This figure depicts histograms of the coverage property across CP models for 17 drugs obtained from the GDSC2 database, for which the number of sensitive cell lines was less than 25%. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.



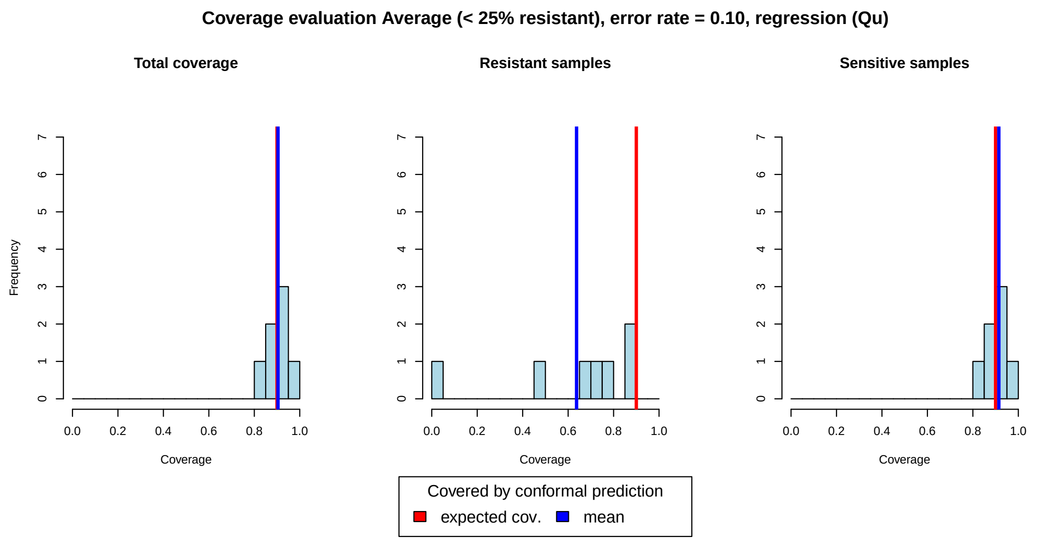
(a) True-class score



(b) Mondrian score



(c) Summation score



(d) Quantile score

Figure E.44: Coverage evaluation for CP models of seven drugs from the GDSC2 database trained using CMax viability values and a two-class classification setting with underrepresentation of the resistant class. This figure depicts histograms of the coverage property across CP models for seven drugs obtained from the GDSC2 database, for which the number of resistant cell lines was less than 25%. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.

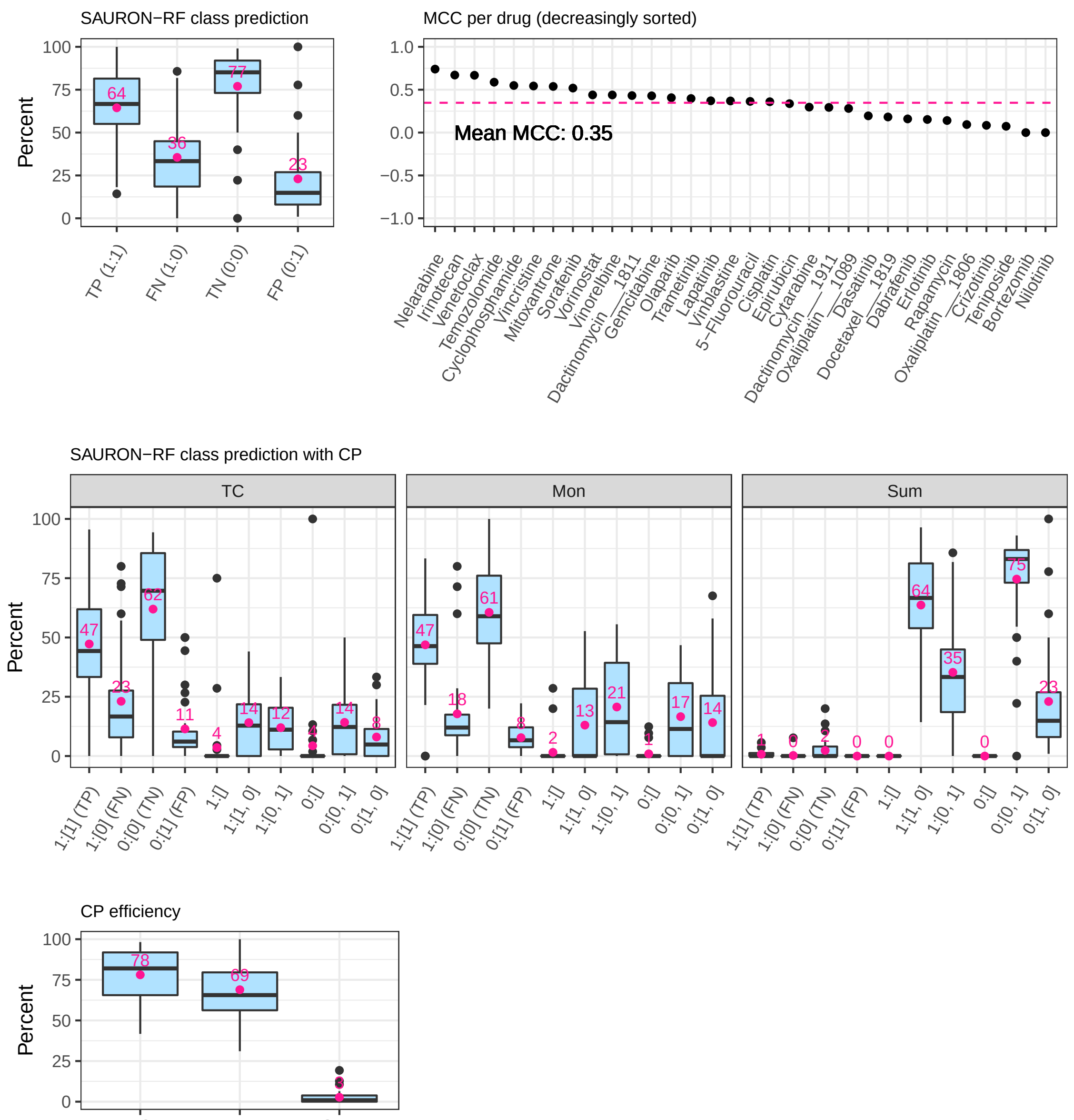


Figure E.45: Classification test set performance for 32 drugs from the GDSC2 database trained using CMax viability values and a two-class classification setting. The upper row of this figure depicts the classification performance of SAURON-RF across the different drugs. The middle row shows the effects of CP on the performance in terms of true positive/negative predictions. In the lower row, the CP efficiency is presented.

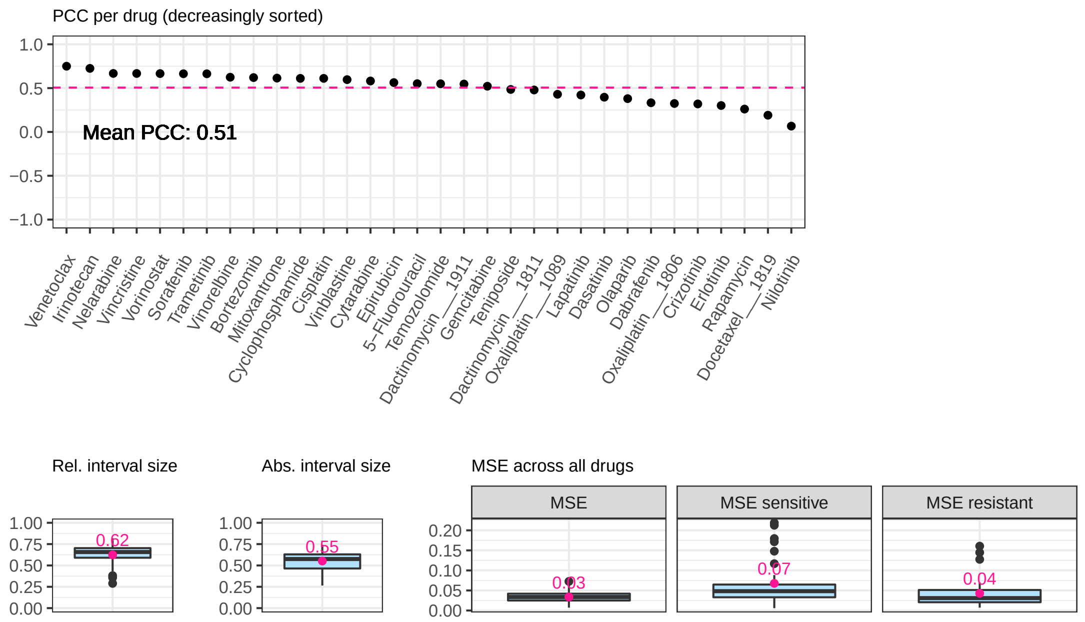
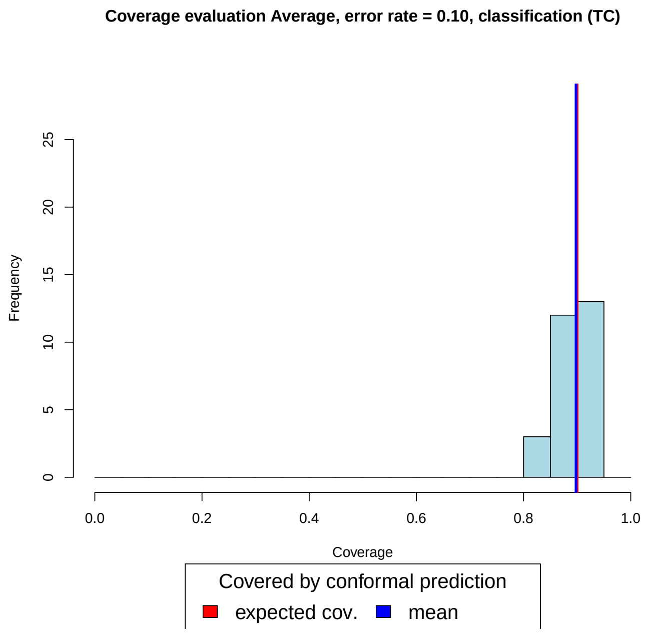
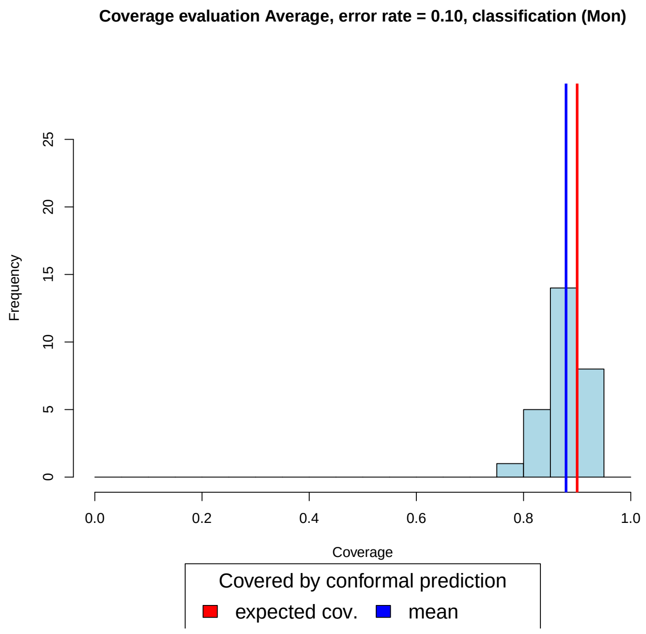


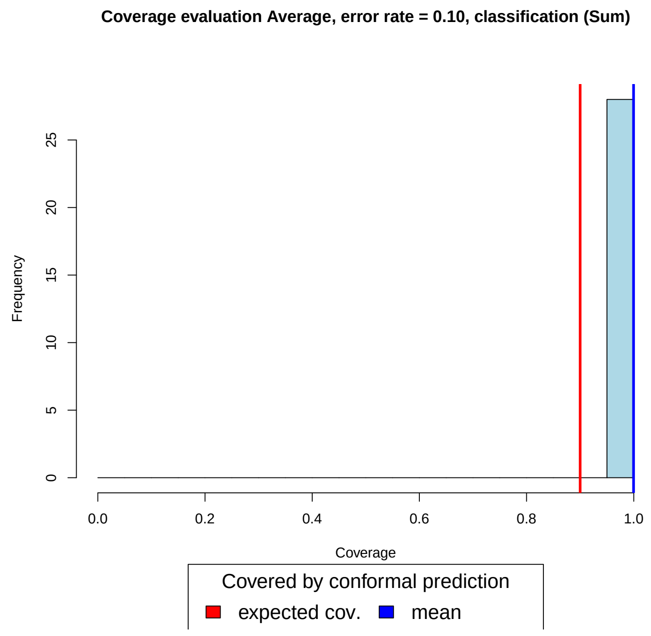
Figure E.46: Regression test set performance for 32 drugs from the GDSC2 database trained using CMax viability values and a two-class classification setting. The upper row of this figure depicts the Pearson correlation coefficient between the actual continuous response values and the predicted continuous response values for all drugs. The lower row shows the mean-squared error (MSE) and the interval width of the CP Quantile regression score relative to the spanned training ranges of the drugs.



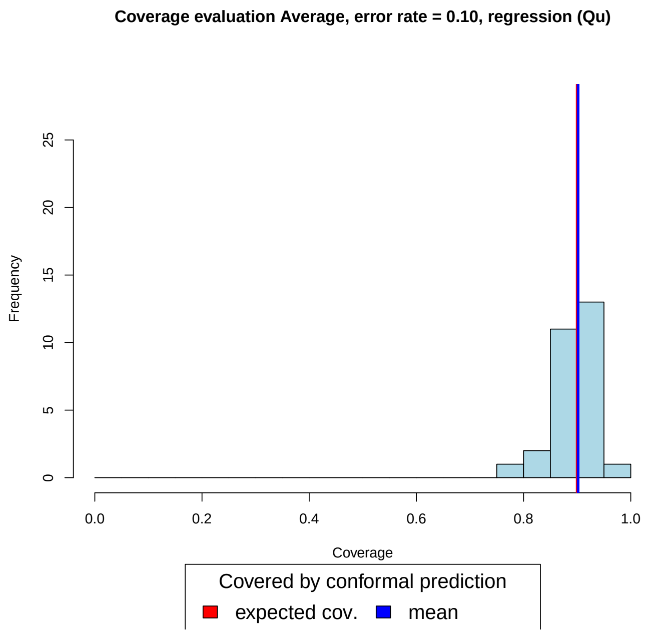
(a) True-class score



(b) Mondrian score



(c) Summation score



(d) Quantile score

Figure E.47: Coverage evaluation for CP models of 28 drugs from the GDSC2 database trained using CMax viability values and a three-class classification setting. This figure depicts histograms of the coverage property across CP models for 28 drugs obtained from the GDSC2 database. The coverage is computed as the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.

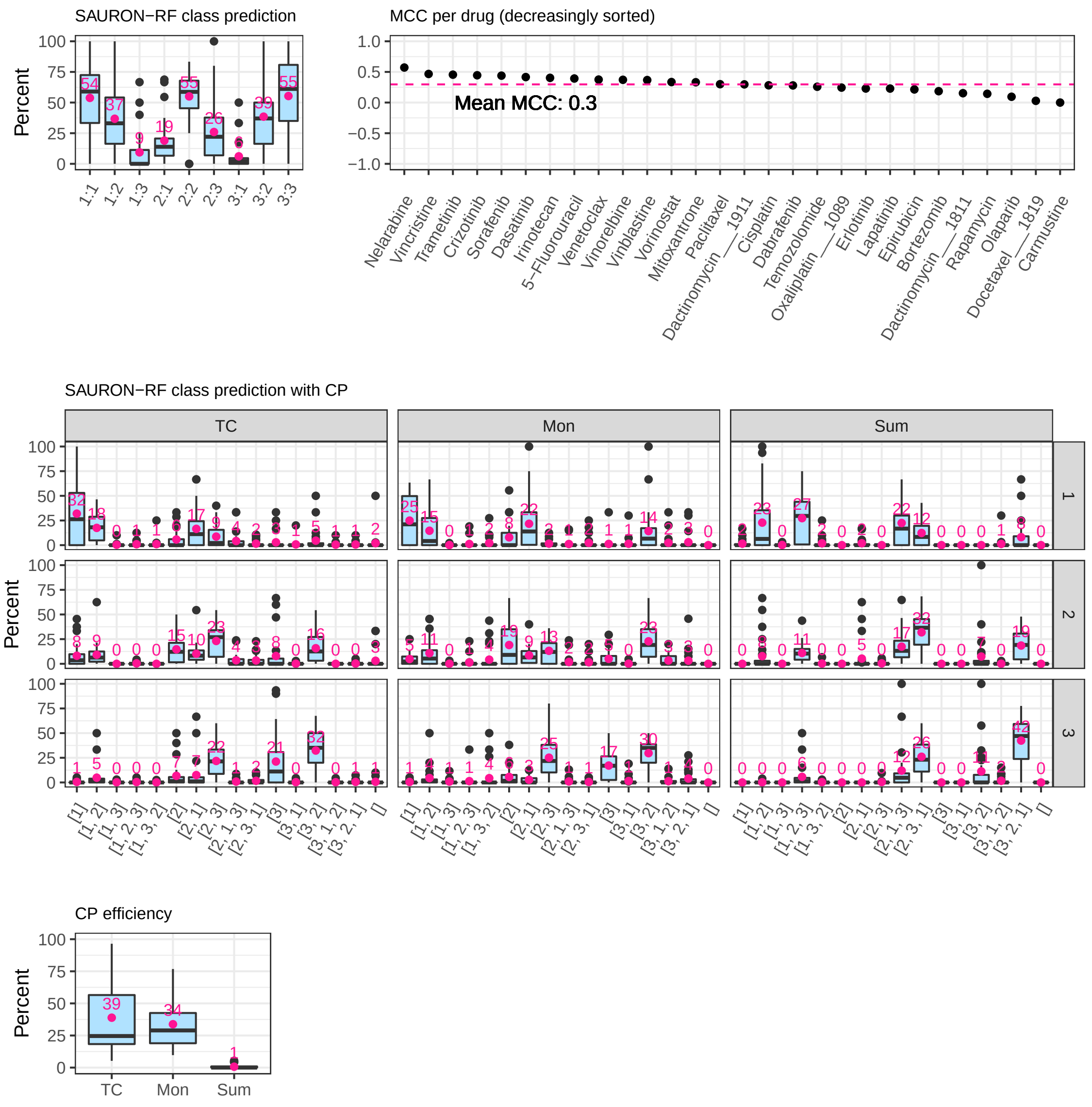


Figure E.48: Classification test set performance for 28 drugs from the GDSC2 database trained using CMax viability values and a three-class classification setting. The upper row of this figure depicts the classification performance of SAURON-RF across the different drugs. The middle row shows the effects of CP on the performance in terms of confusions between classes. In the lower row, the CP efficiency is presented.

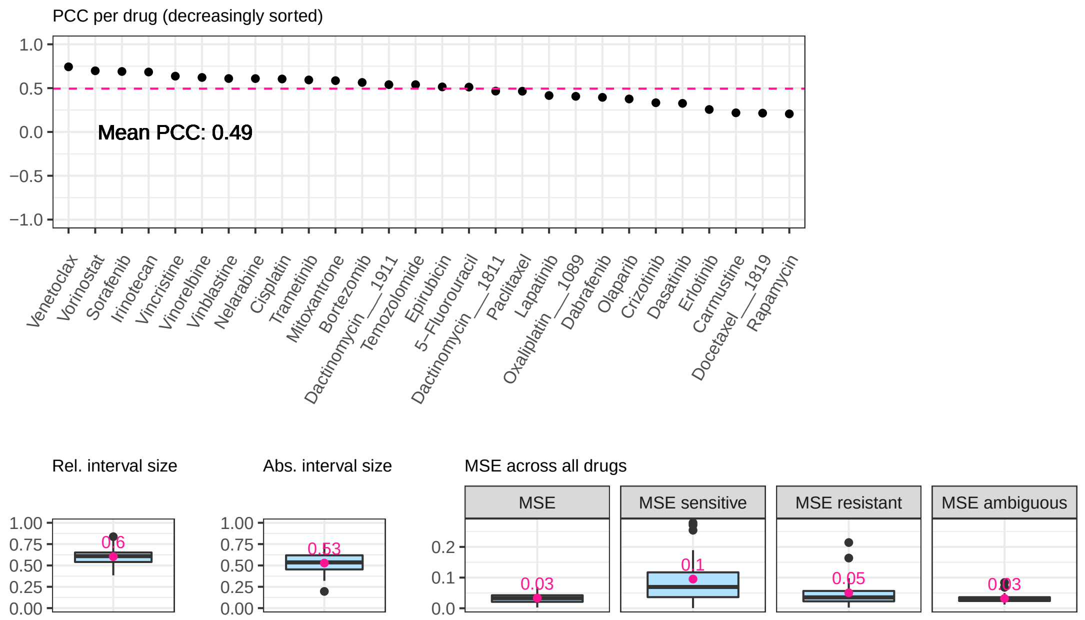


Figure E.49: Regression test set performance for 28 drugs from the GDSC2 database trained using CMax viability values and a three-class classification setting. The upper row of this figure depicts the Pearson correlation coefficient between the actual continuous response values and the predicted continuous response values for all drugs. The lower row shows the mean-squared error (MSE) and the interval width of the CP Quantile regression score relative to the spanned training ranges of the drugs.

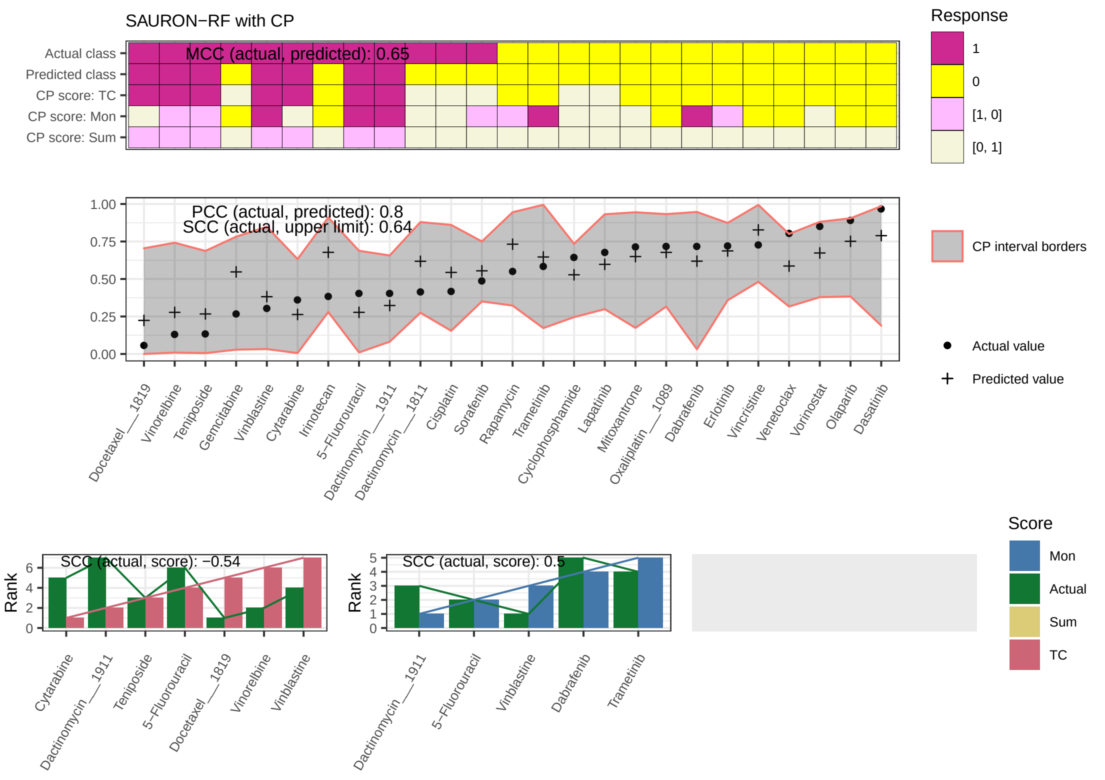


Figure E.50: Prioritization example GDSC2. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 724863) from the test set of the GDSC2 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

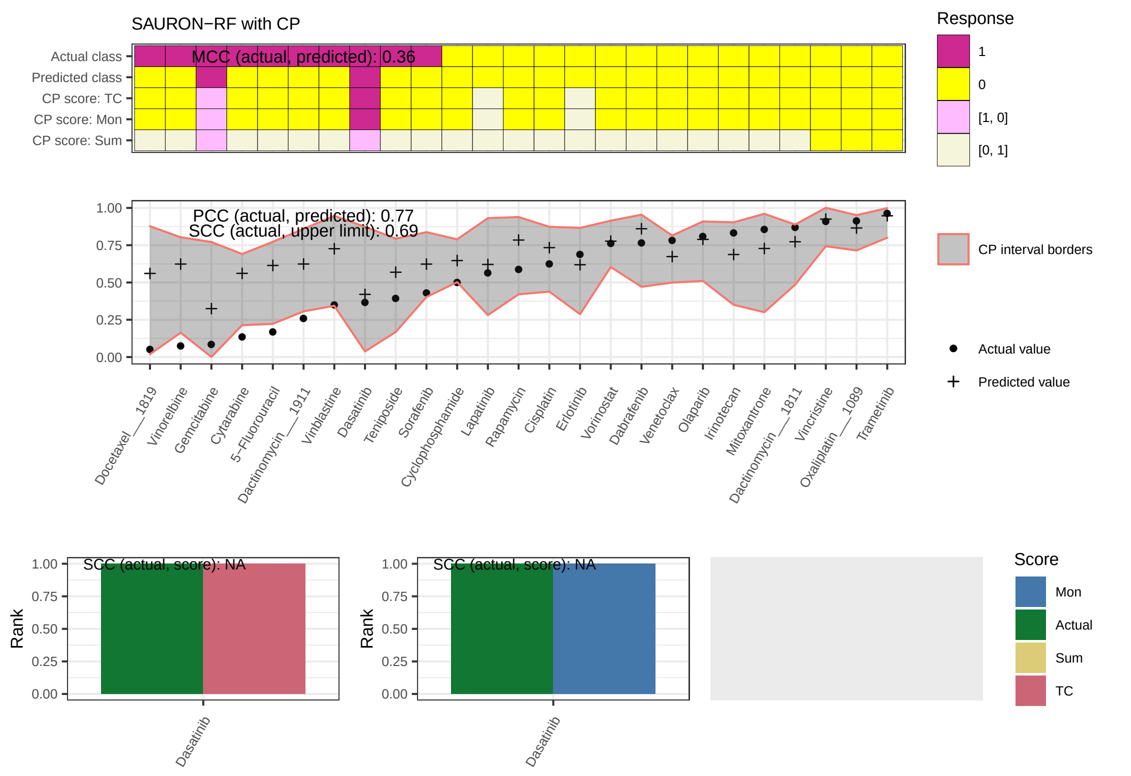


Figure E.51: Prioritization example GDSC2. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 906861) from the test set of the GDSC2 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

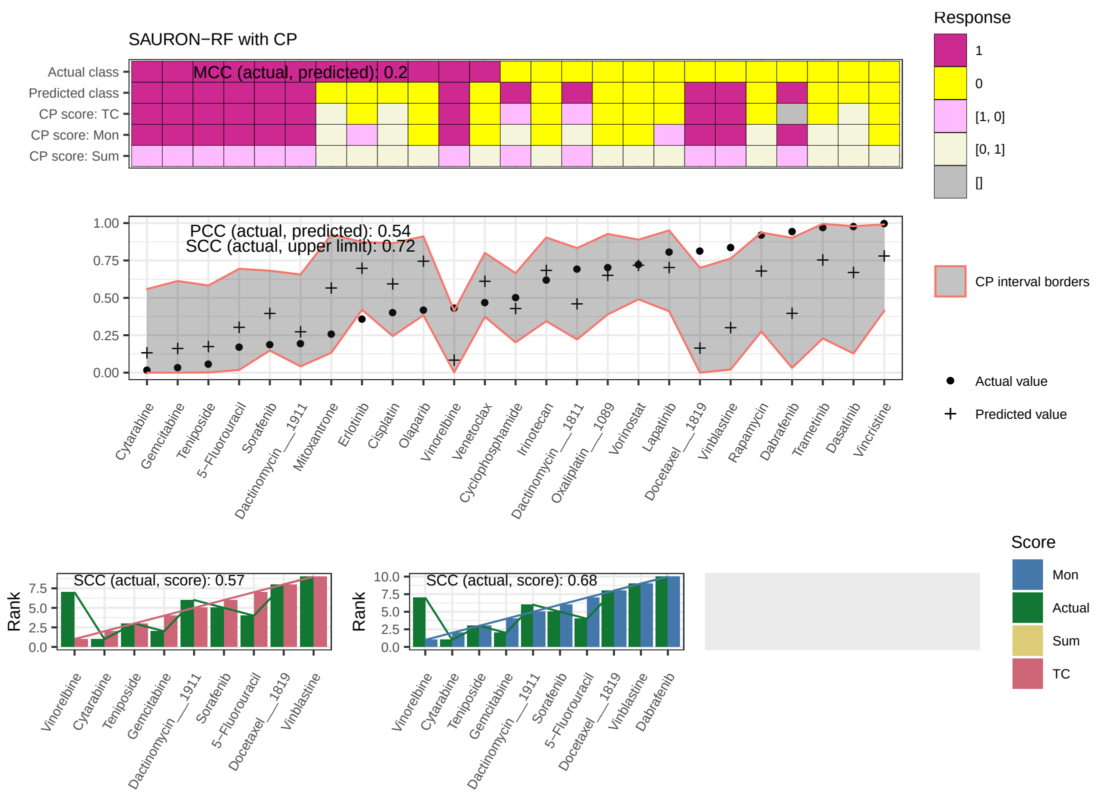


Figure E.52: Prioritization example GDSC2. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 908444) from the test set of the GDSC2 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

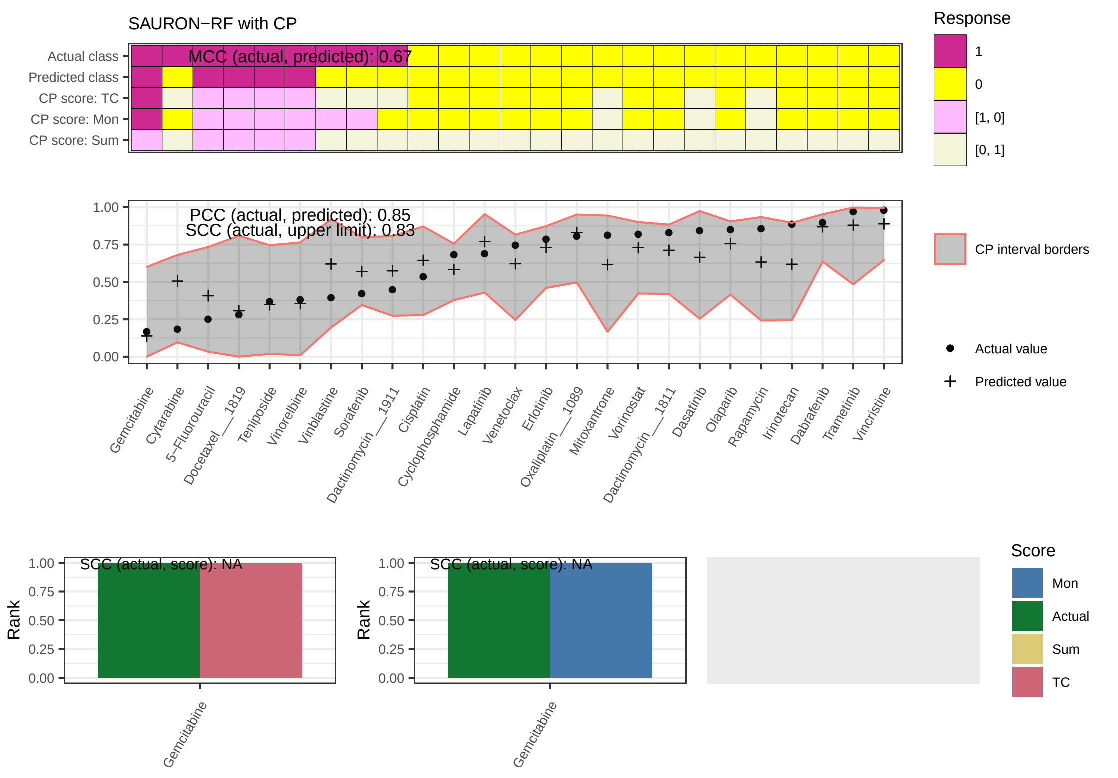
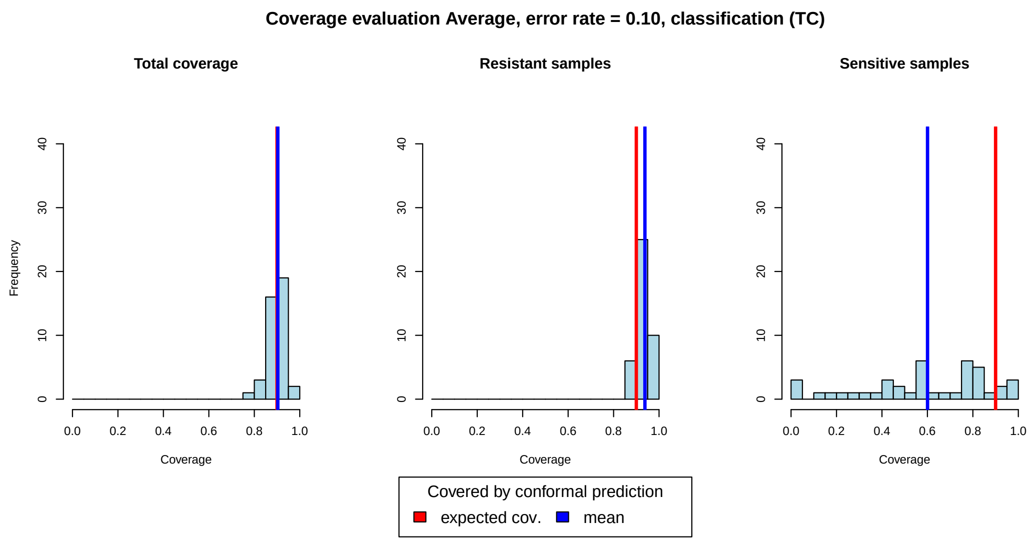
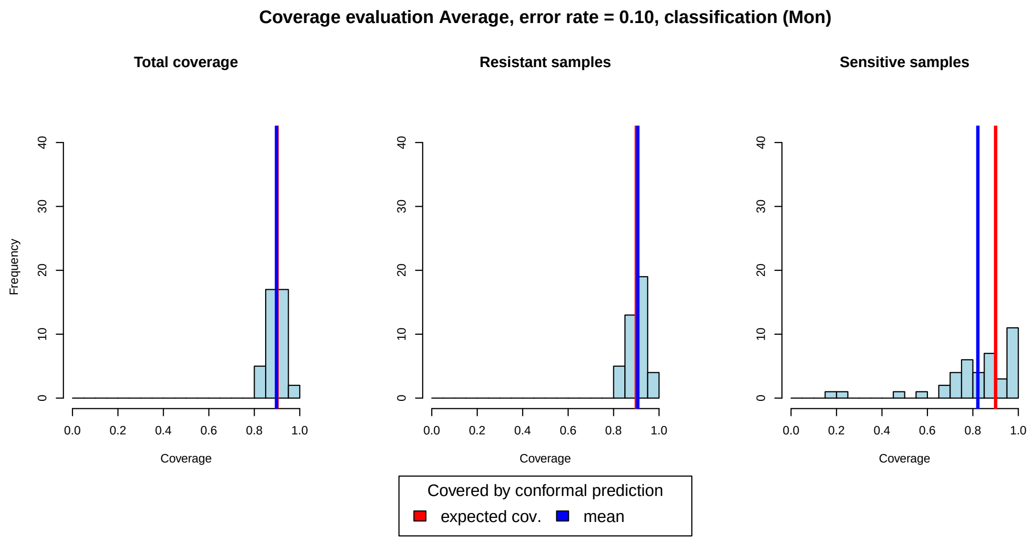


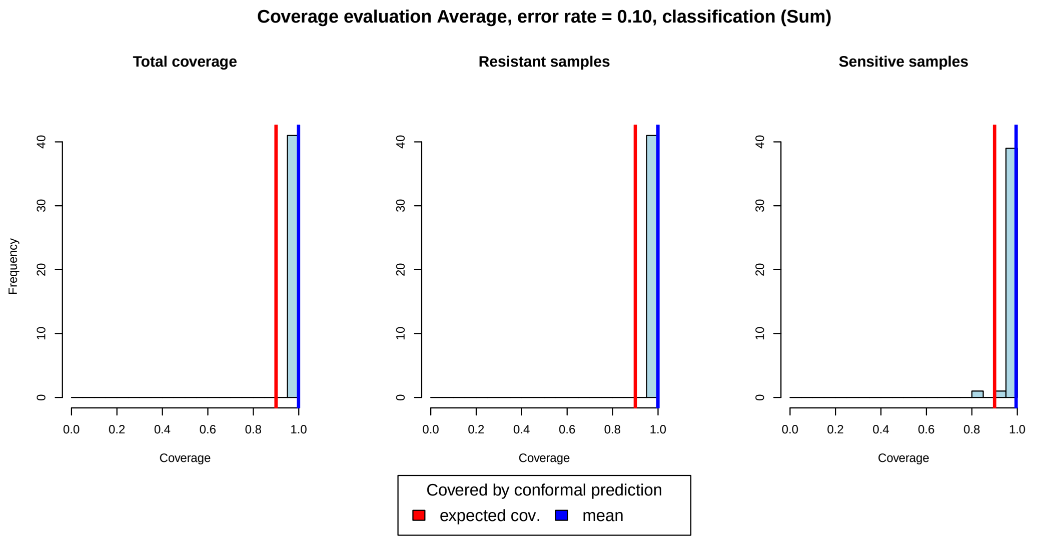
Figure E.53: Prioritization example GDSC2. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 1298216) from the test set of the GDSC2 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.



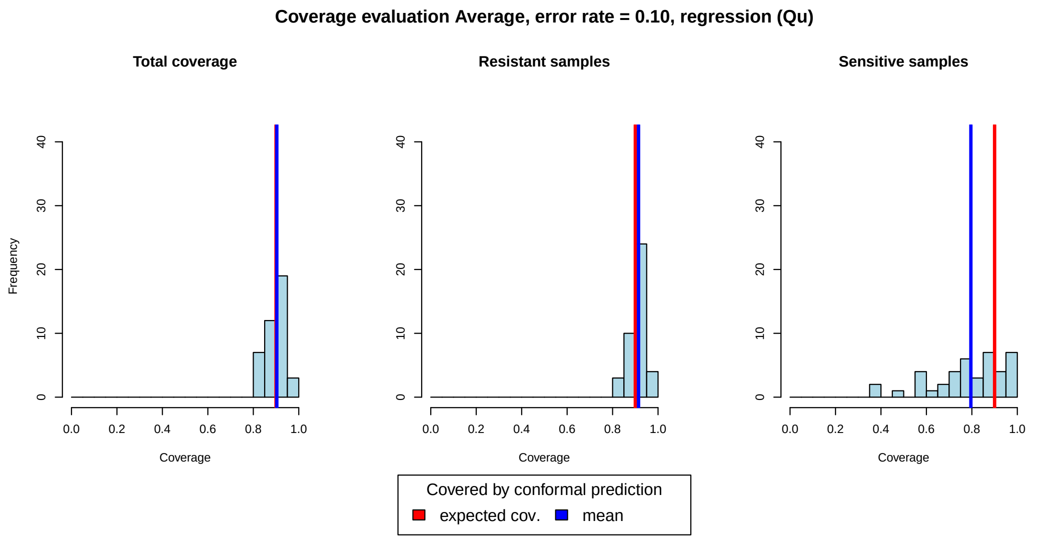
(a) True-class score



(b) Mondrian score



(c) Summation score



(d) Quantile score

Figure E.54: Coverage evaluation for CP models of 41 drugs from the GDSC1 database trained using IC50 values and a two-class classification setting. This figure depicts histograms of the coverage property across CP models for 41 drugs obtained from the GDSC1 database. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.

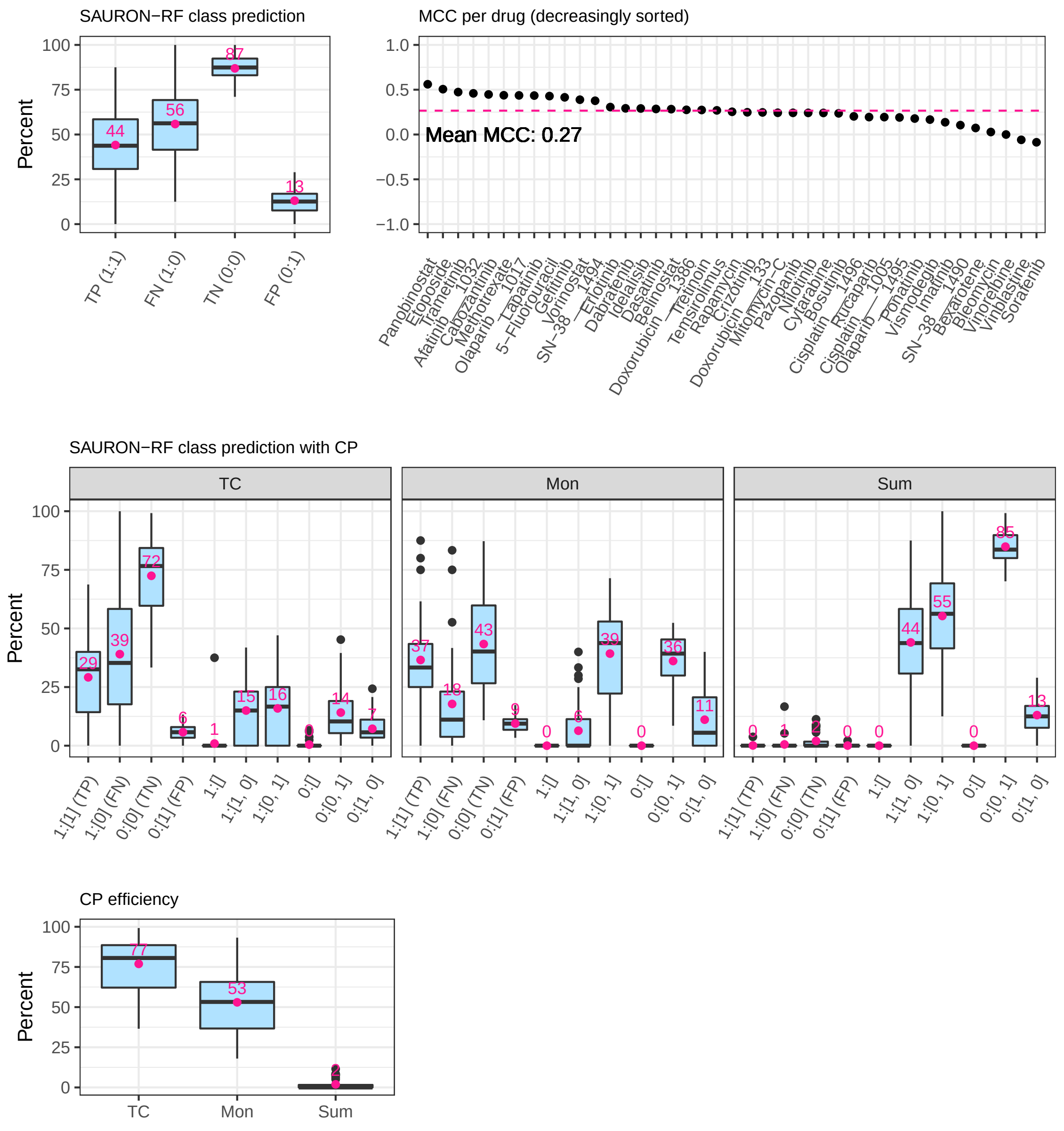


Figure E.55: Classification test set performance for 41 drugs from the GDSC1 database trained using IC50 values and a two-class classification setting. The upper row of this figure depicts the classification performance of SAURON-RF across the different drugs. The middle row shows the effects of CP on the performance in terms of true positive/negative predictions. In the lower row, the CP efficiency is presented.

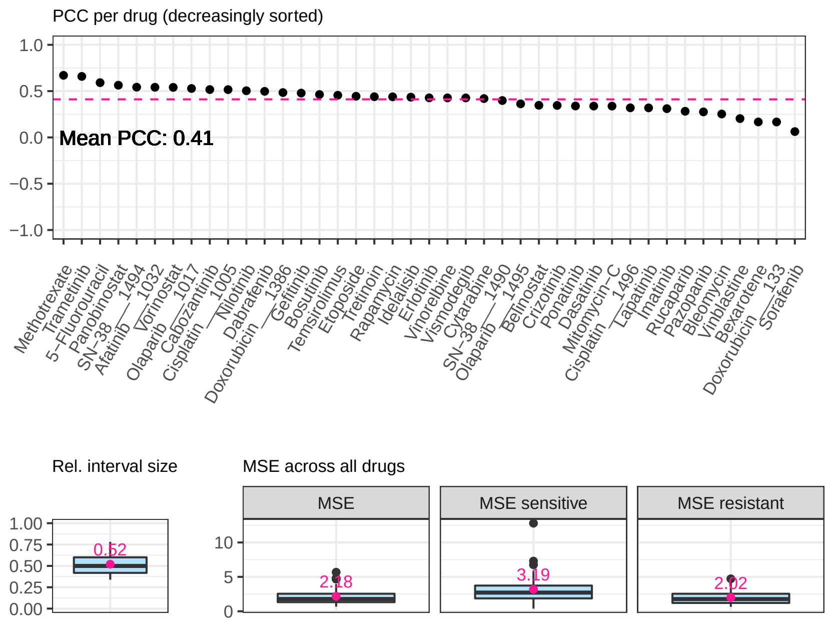
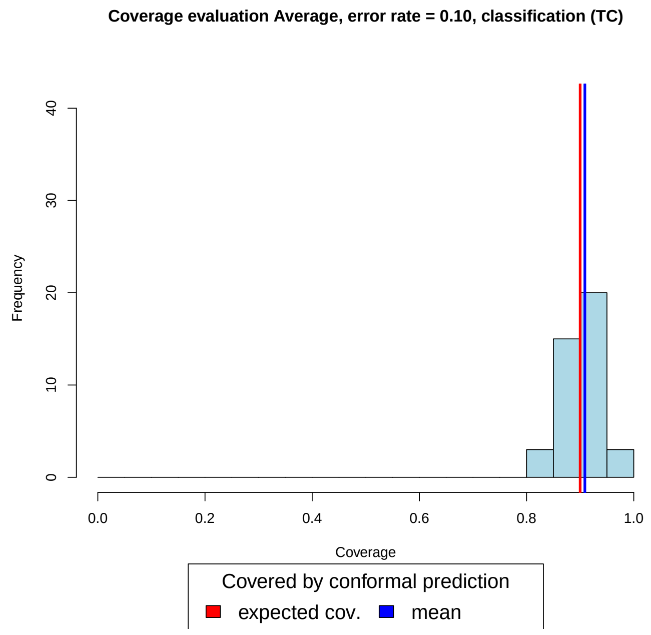
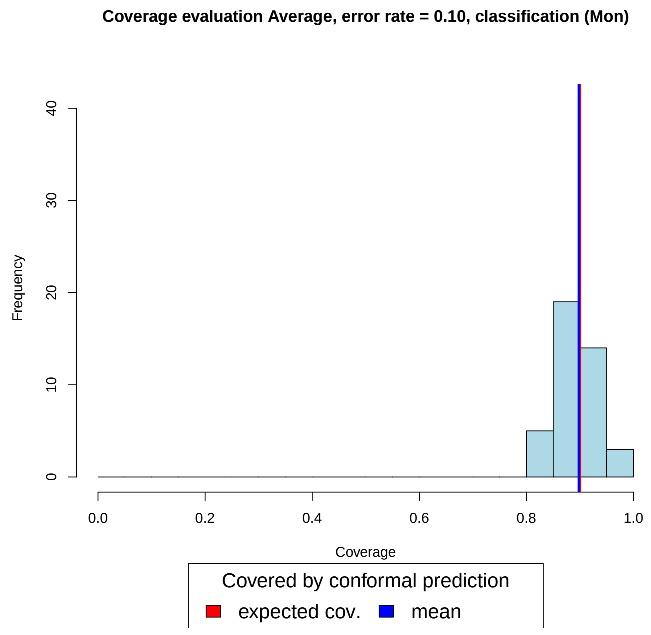


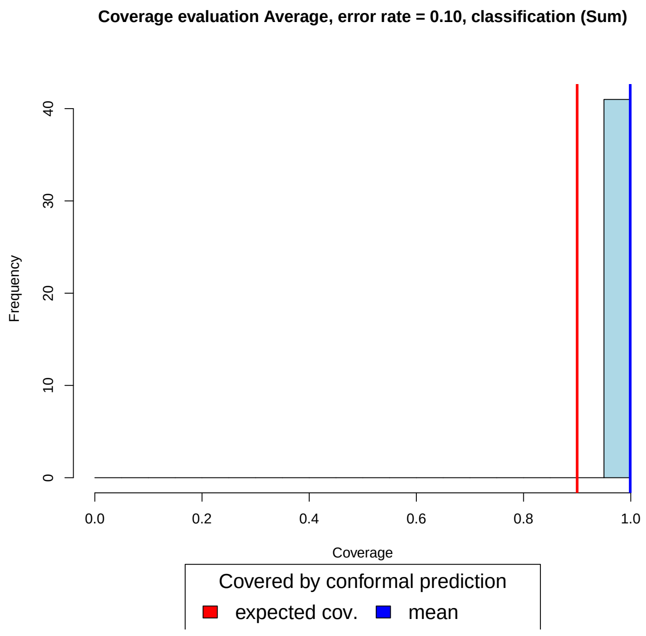
Figure E.56: Regression test set performance for 41 drugs from the GDSC1 database trained using IC50 values and a two-class classification setting. The upper row of this figure depicts the Pearson correlation coefficient between the actual continuous response values and the predicted continuous response values for all drugs. The lower row shows the mean-squared error (MSE) and the interval width of the CP Quantile regression score relative to the spanned training ranges of the drugs.



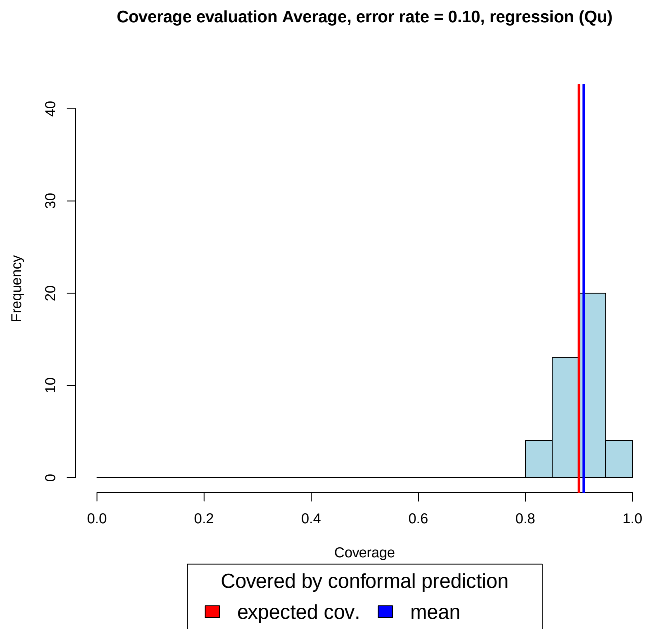
(a) True-class score



(b) Mondrian Score

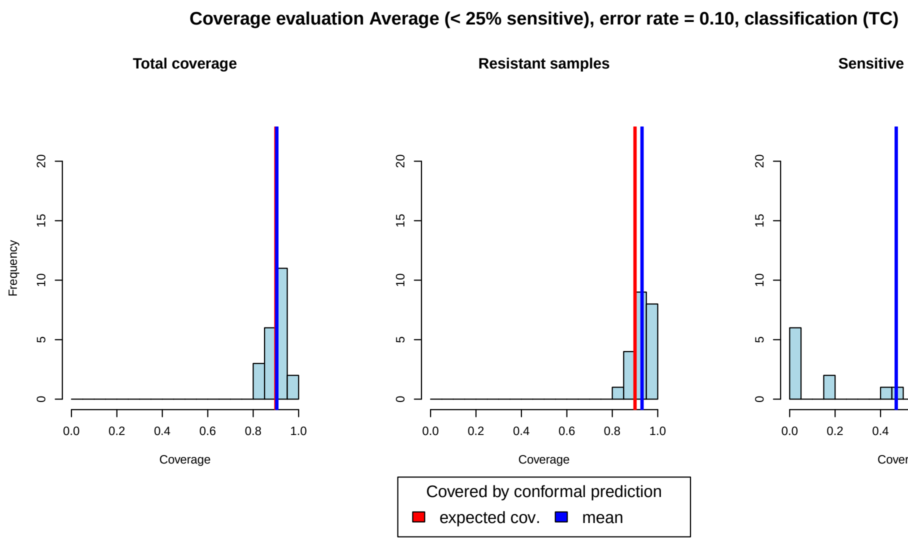


(c) Summation score

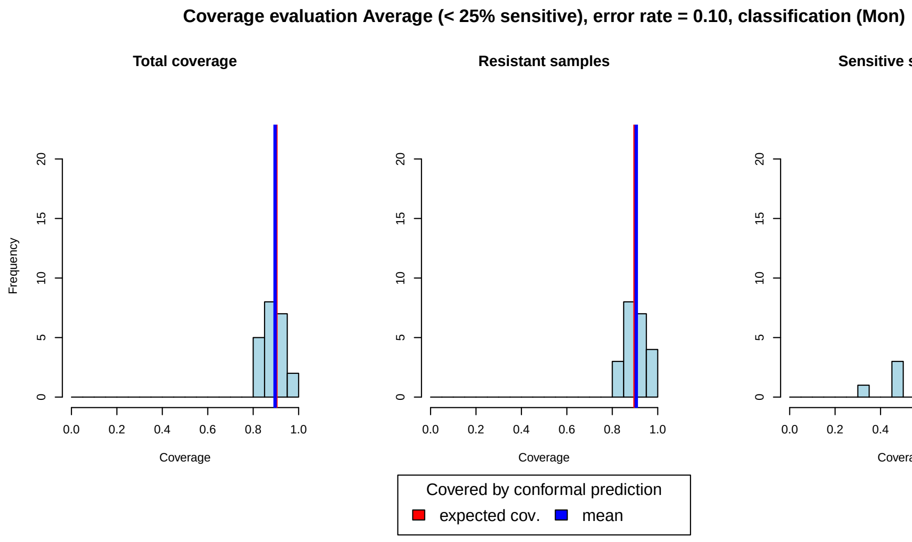


(d) Quantile score

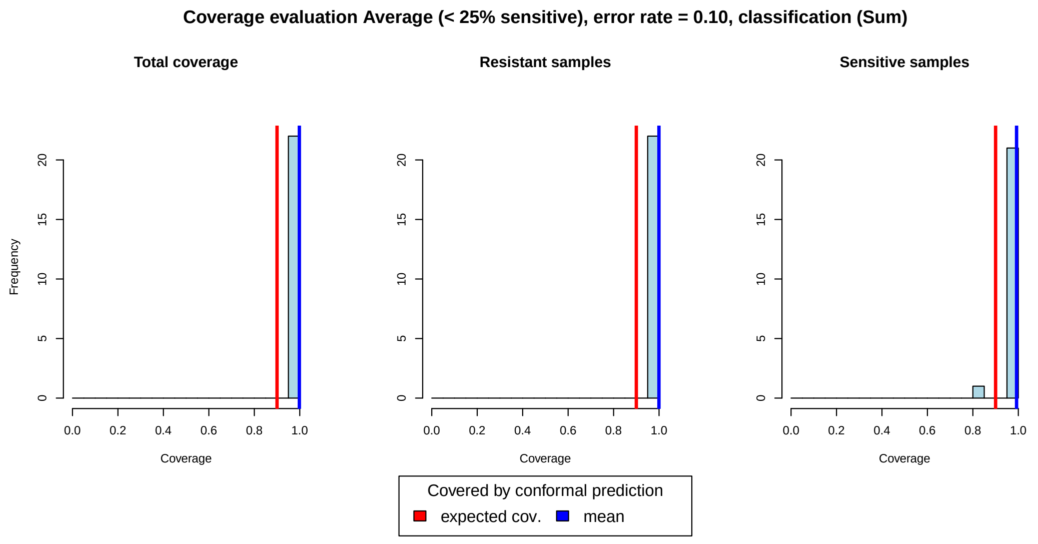
Figure E.57: Coverage evaluation for CP models of 41 drugs from the GDSC1 database trained using CMax viability values and a two-class classification setting. This figure depicts histograms of the coverage property across CP models for 41 drugs obtained from the GDSC1 database. The coverage is computed as the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.



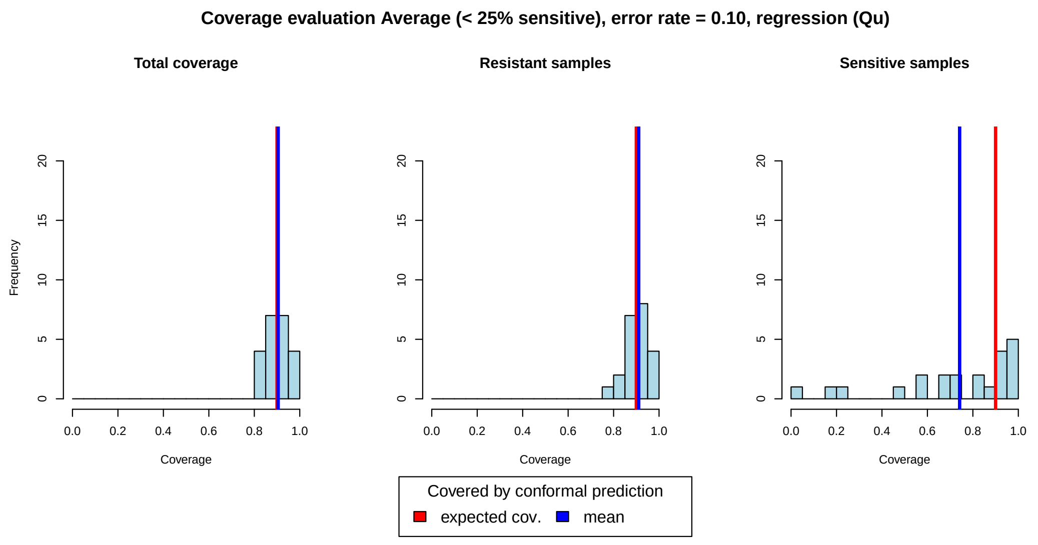
(a) True-class score



(b) Mondrian score

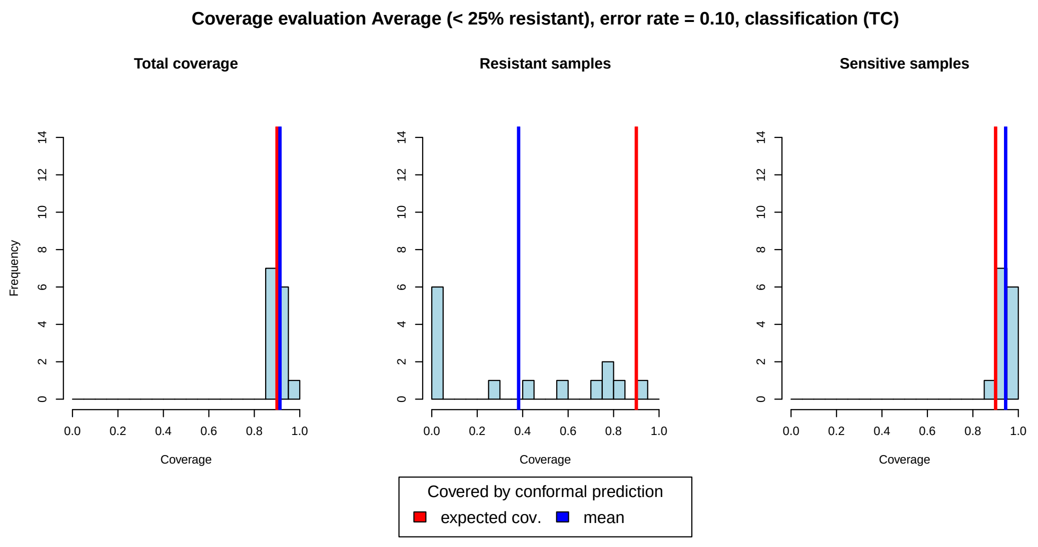


(c) Summation score

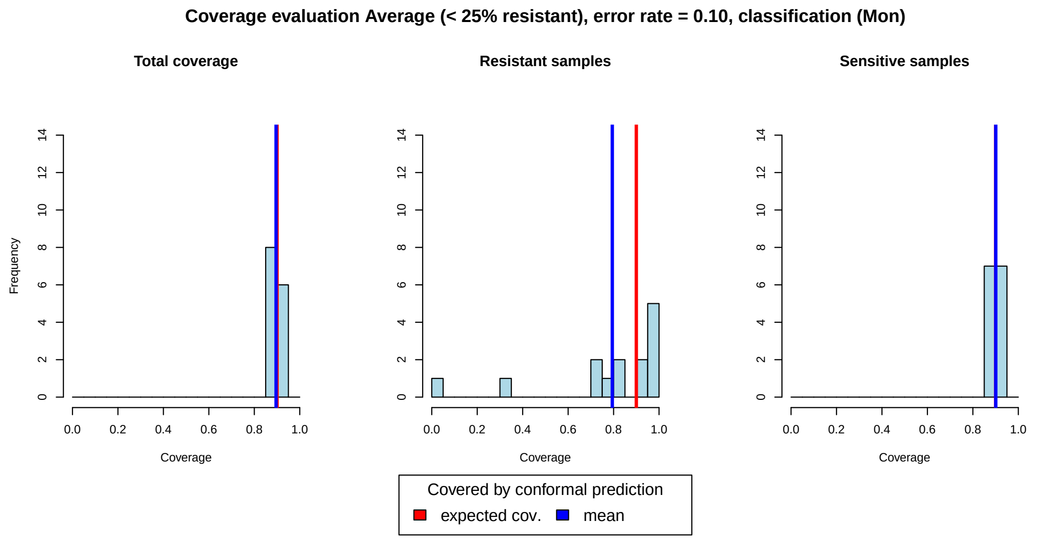


(d) Quantile score

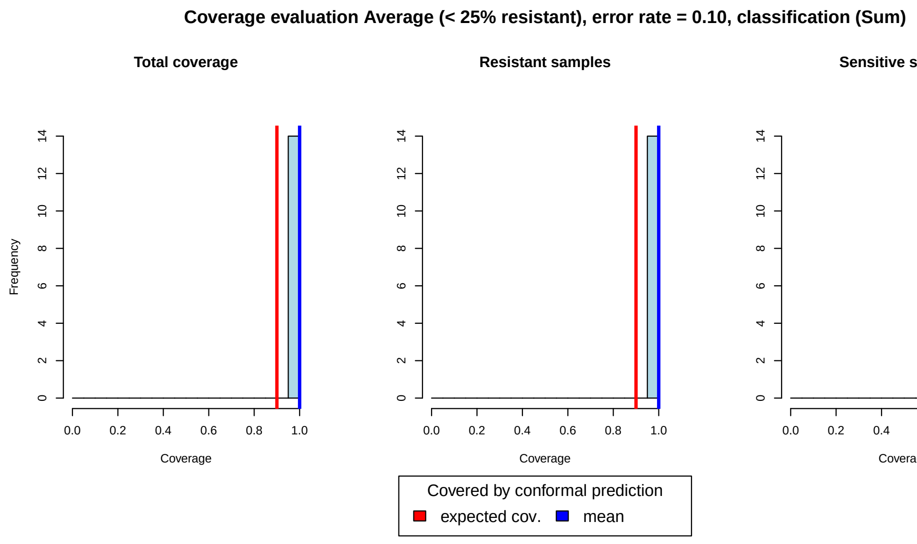
Figure E.58: Coverage evaluation for CP models of 22 drugs from the GDSC1 database trained using CMax viability values and a two-class classification setting with underrepresentation of the sensitive class. This figure depicts histograms of the coverage property across CP models for 22 drugs obtained from the GDSC1 database, for which the number of sensitive cell lines was less than 25%. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.



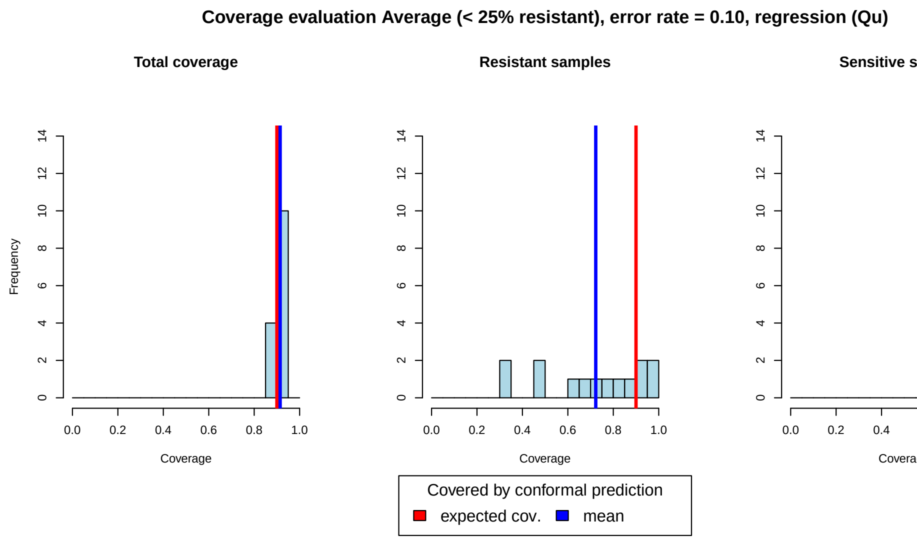
(a) True-class score



(b) Mondrian score



(c) Summation score



(d) Quantile score

Figure E.59: Coverage evaluation for CP models of 14 drugs from the GDSC1 database trained using CMax viability values and a two-class classification setting with underrepresentation of the resistant class. This figure depicts histograms of the coverage property across CP models for 14 drugs obtained from the GDSC1 database, for which the number of resistant cell lines was less than 25%. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.

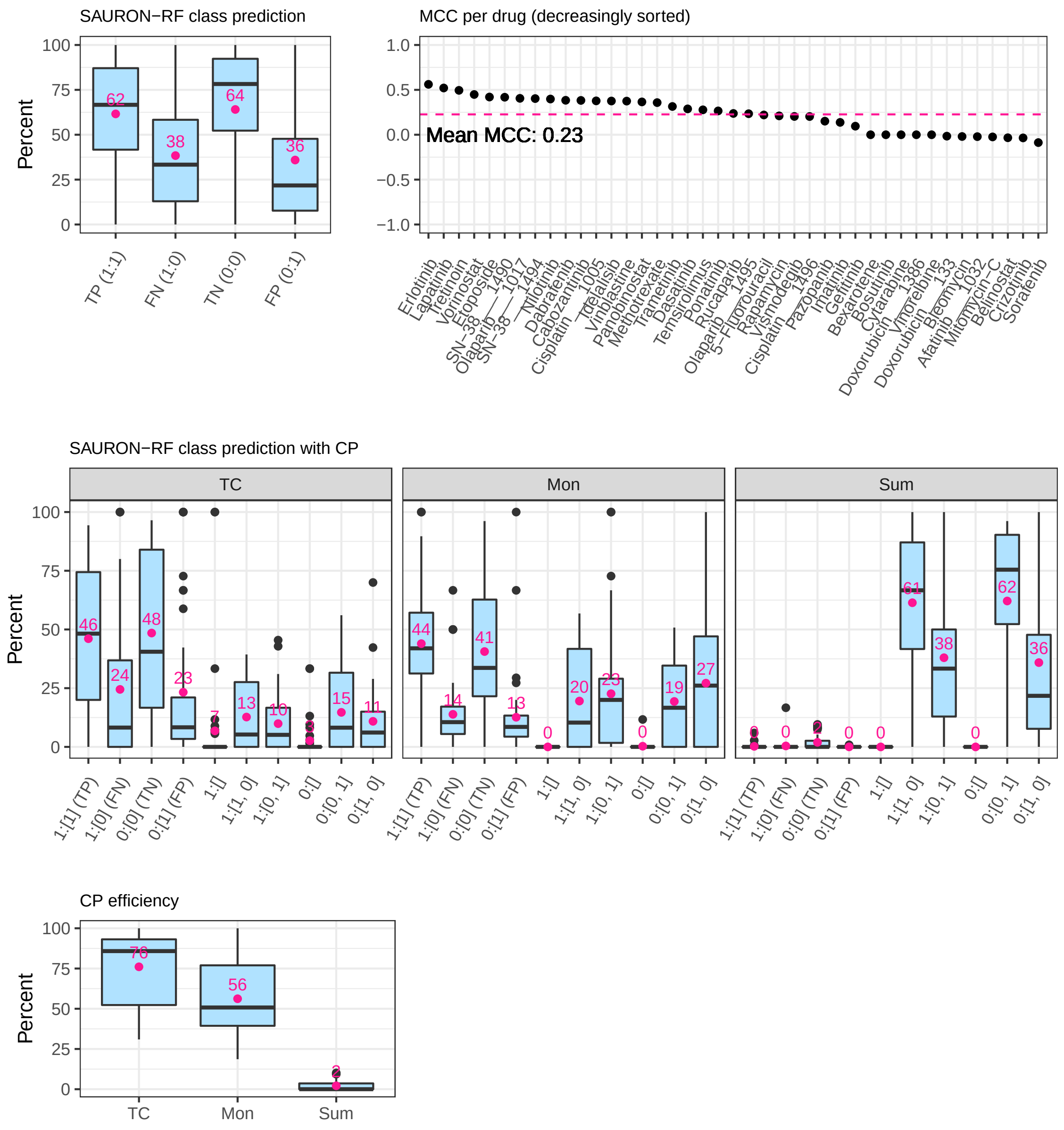


Figure E.60: Classification test set performance for 41 drugs from the GDSC1 database trained using CMax viability values and a two-class classification setting. The upper row of this figure depicts the classification performance of SAURON-RF across the different drugs. The middle row shows the effects of CP on the performance in terms of true positive/negative predictions. In the lower row, the CP efficiency is presented.

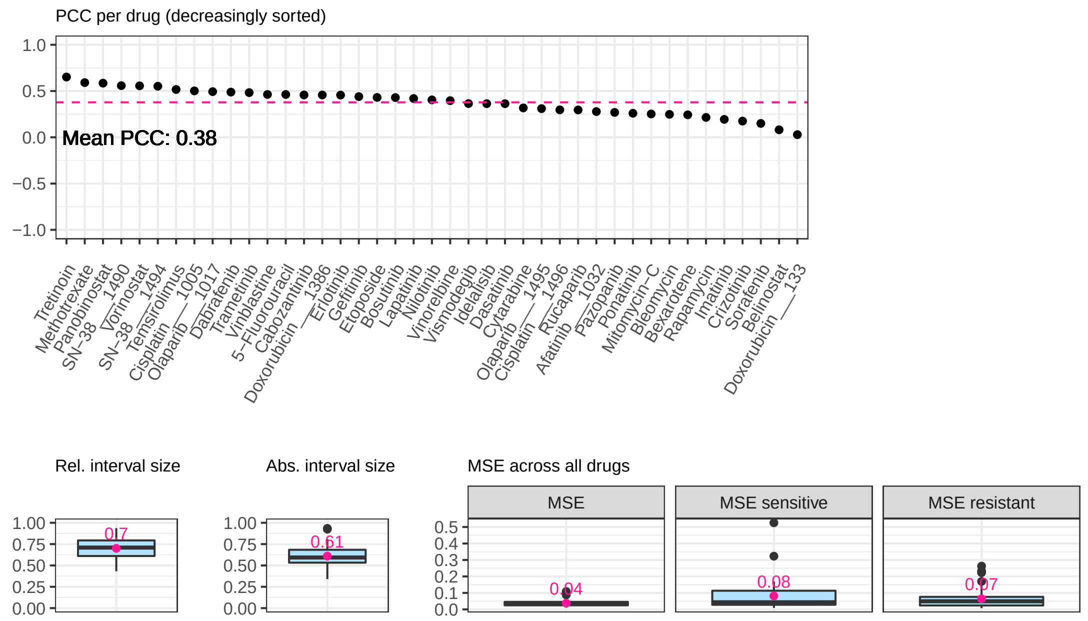
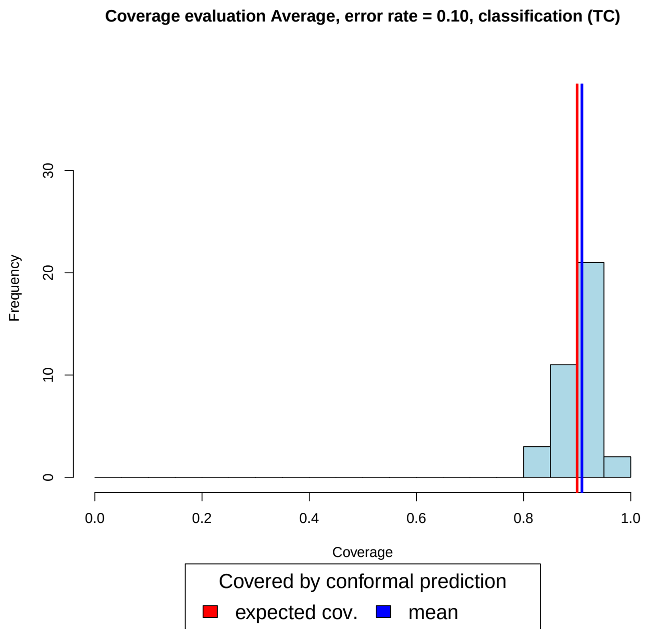
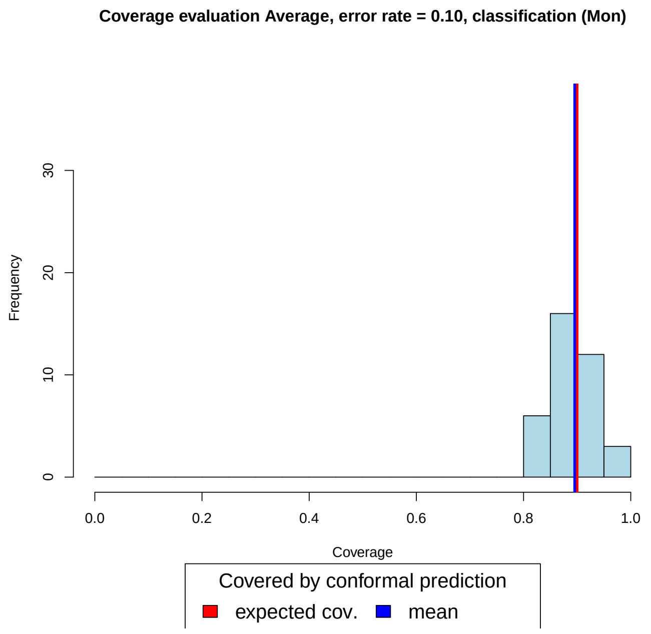


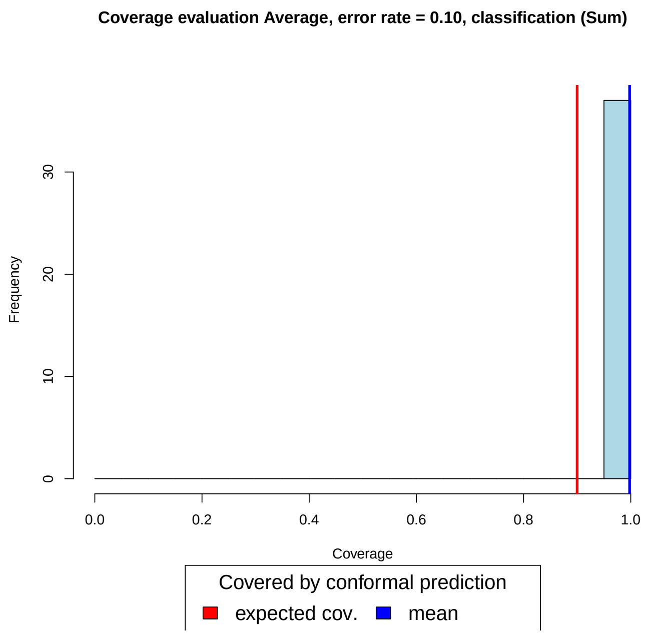
Figure E.61: Regression test set performance for 41 drugs from the GDSC1 database trained using CMax viability values and a two-class classification setting. The upper row of this figure depicts the Pearson correlation coefficient between the actual continuous response values and the predicted continuous response values for all drugs. The lower row shows the mean-squared error (MSE) and the interval width of the CP Quantile regression score relative to the spanned training ranges of the drugs.



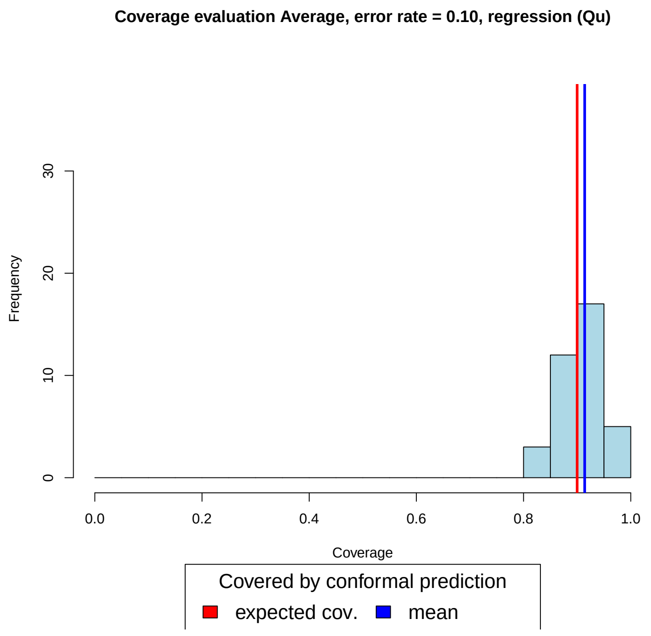
(a) True-class score



(b) Mondrian score



(c) Summation score



(d) Quantile score

Figure E.62: Coverage evaluation for CP models of 37 drugs from the GDSC1 database trained using CMax viability values and a three-class classification setting. This figure depicts histograms of the coverage property across CP models for 37 drugs obtained from the GDSC1 database. The coverage is computed as the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.

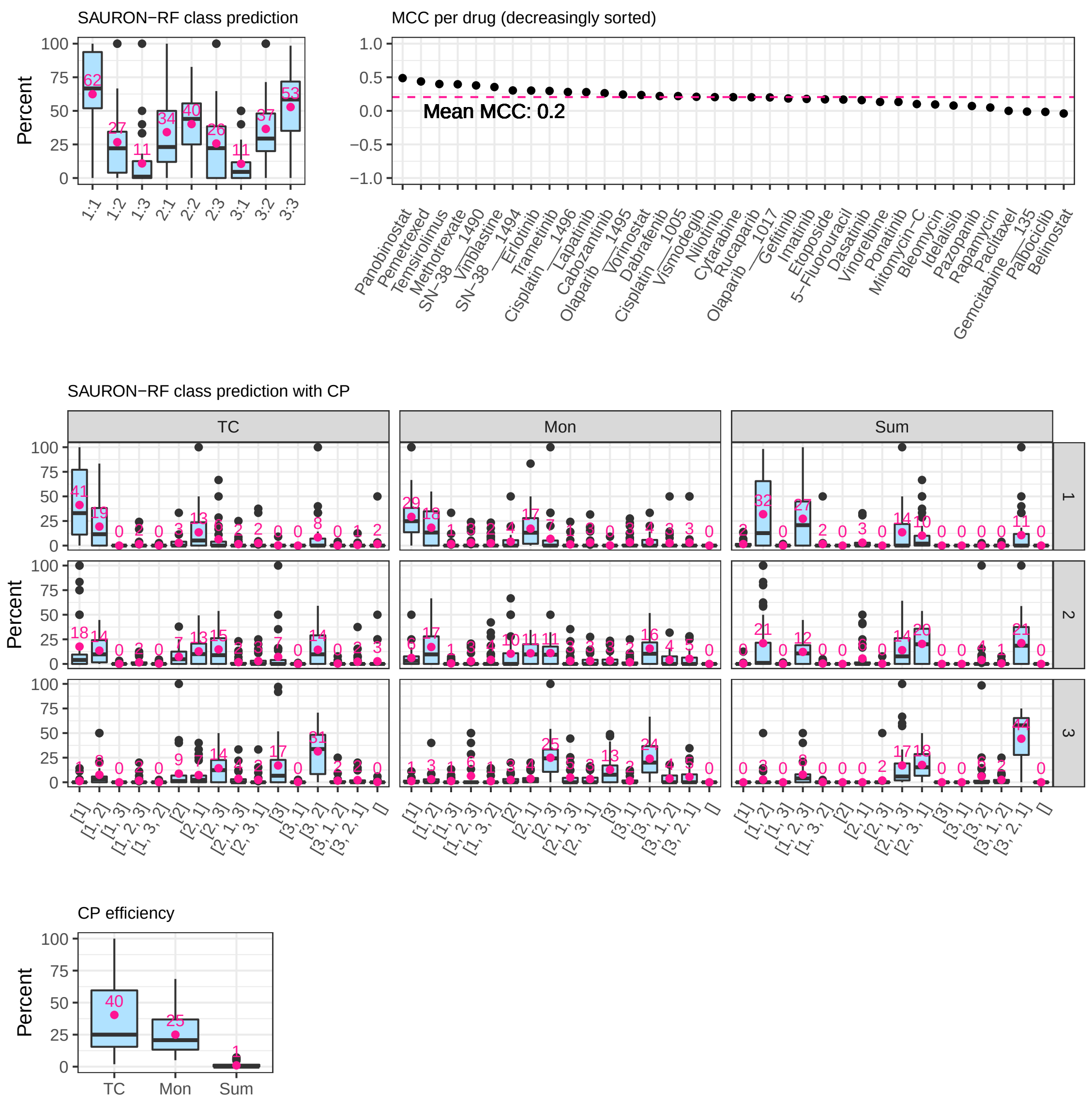


Figure E.63: Classification test set performance for 37 drugs from the GDSC1 database trained using CMax viability values and a three-class classification setting. The upper row of this figure depicts the classification performance of SAURON-RF across the different drugs from GDSC1. The middle row shows the effects of CP on the performance in terms of confusions between classes. In the lower row, the CP efficiency is presented.

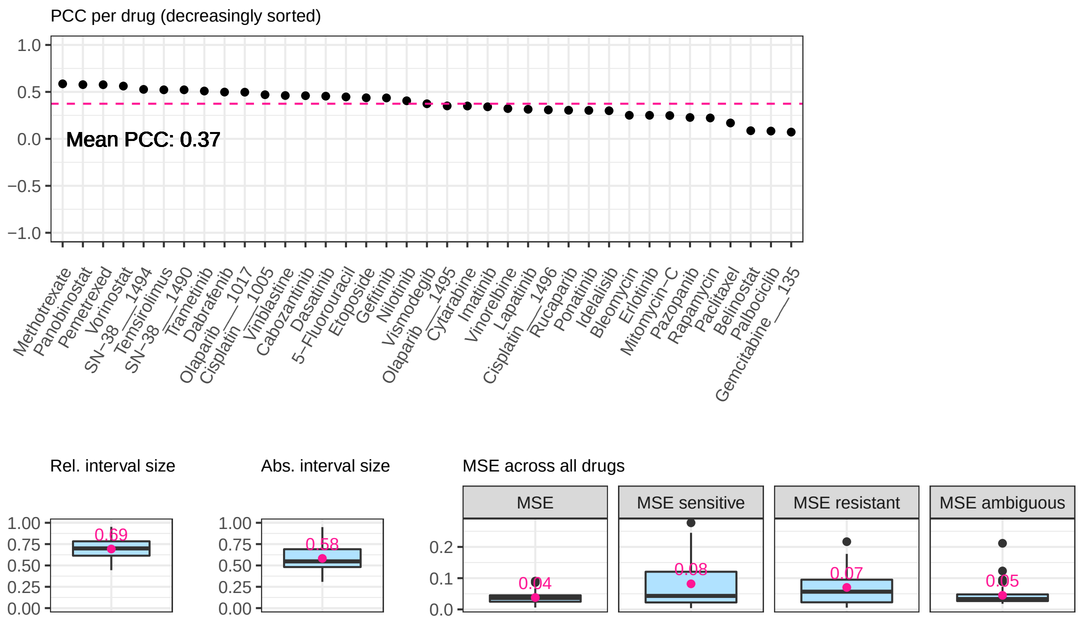
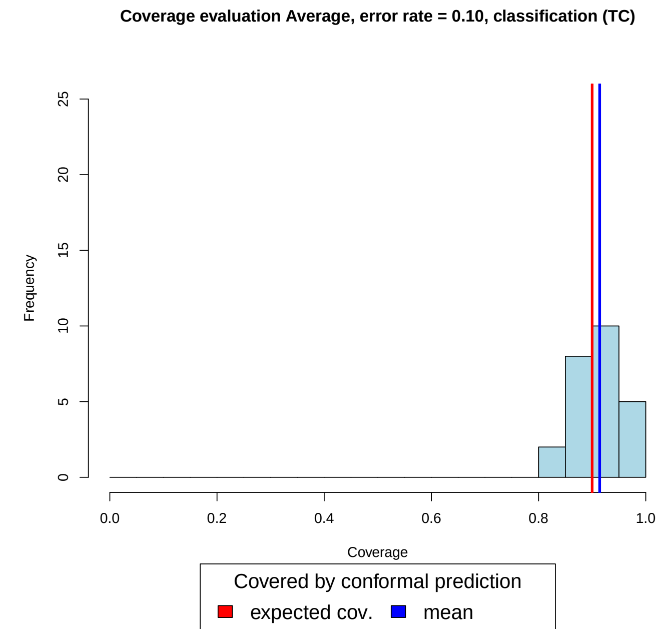
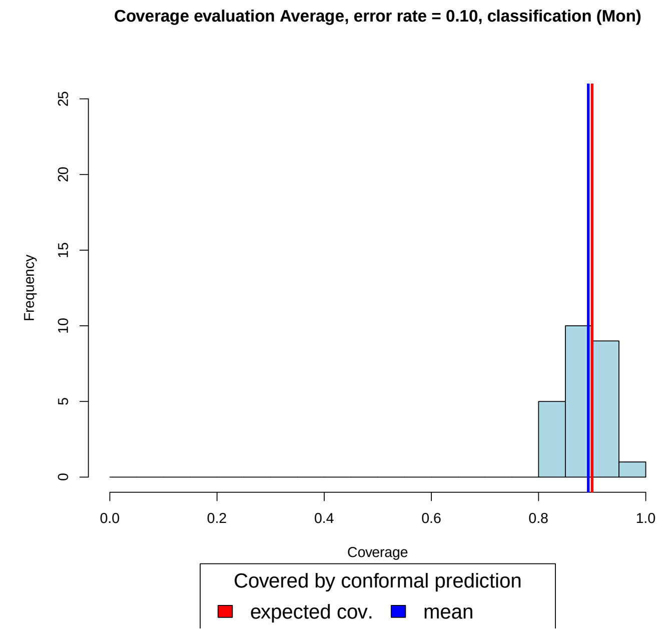


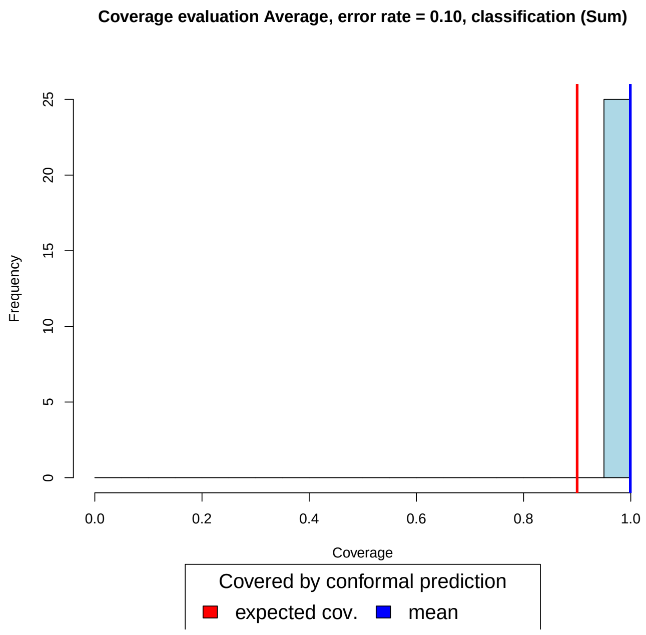
Figure E.64: Regression test set performance for 37 drugs from the GDSC1 database trained using CMax viability values and a three-class classification setting. The upper row of this figure depicts the Pearson correlation coefficient between the actual continuous response values and the predicted continuous response values for all drugs. The lower row shows the mean-squared error (MSE) and the interval width of the CP Quantile regression score relative to the spanned training ranges of the drugs.



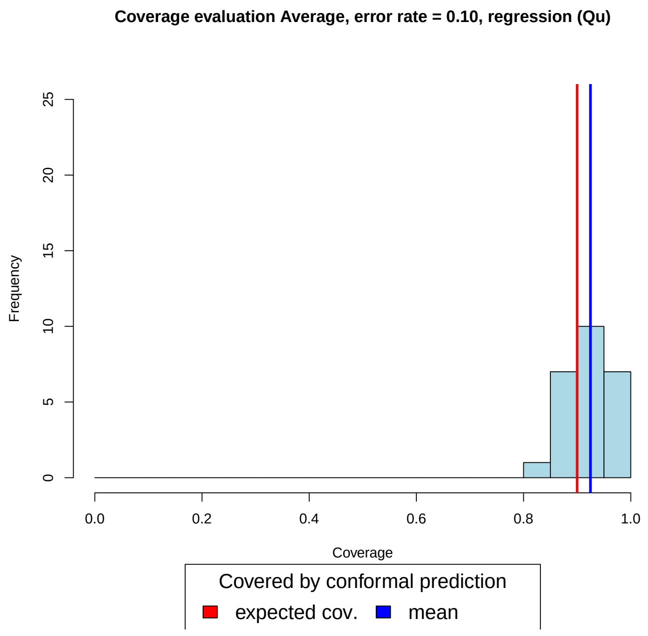
(a) True-class score



(b) Mondrian score

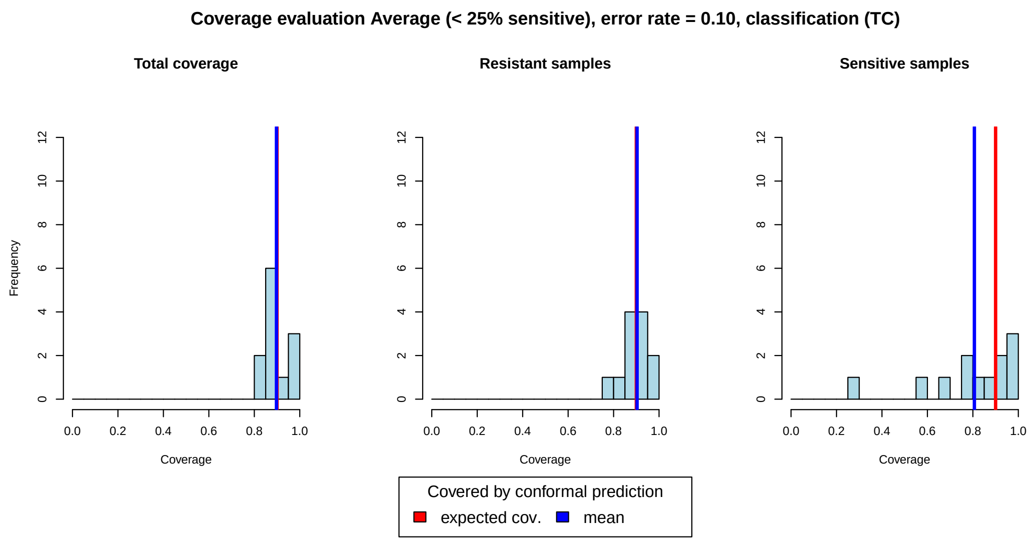


(c) Summation score

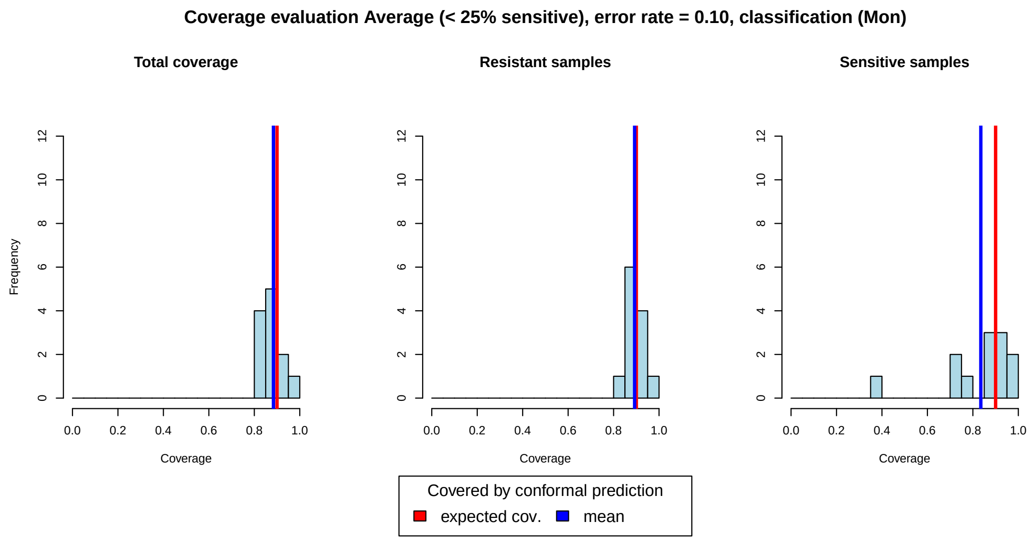


(d) Quantile score

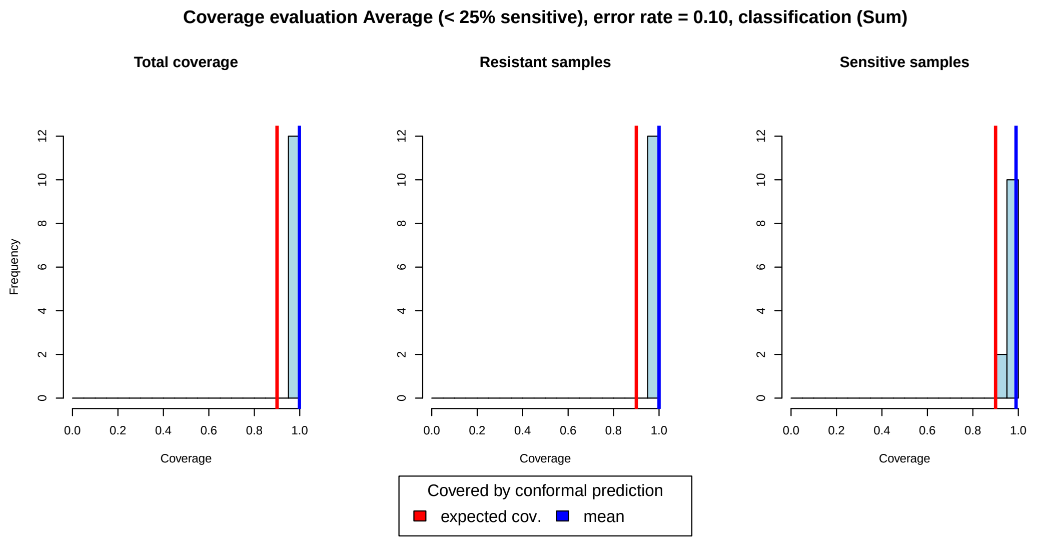
Figure E.65: Coverage evaluation for CP models of 25 drugs from the GDSC1 database trained using CMax viability values and a two-class prioritization setting. This figure depicts histograms of the coverage property across CP models for 25 drugs obtained from the GDSC1 database, which were used to conduct the drug prioritization analyses. The coverage is computed as the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.



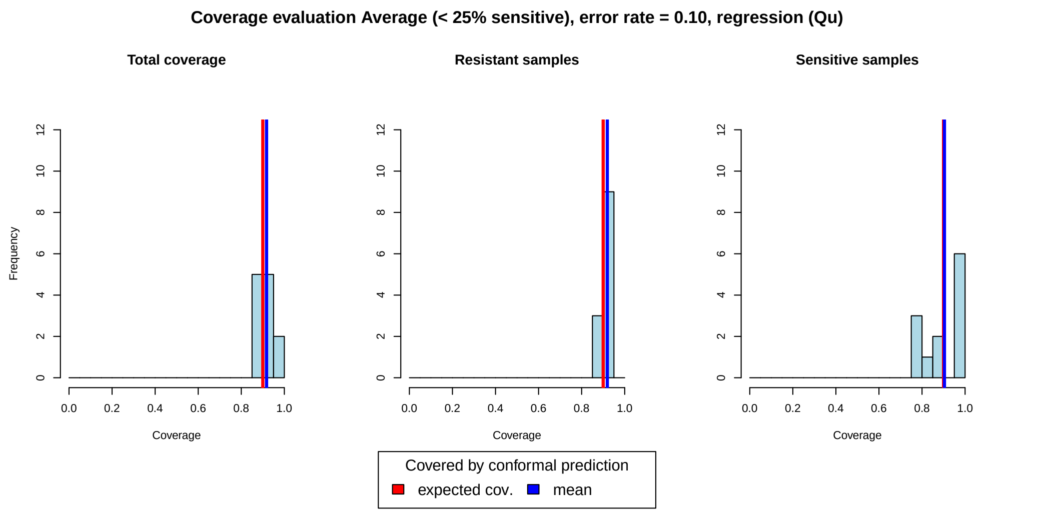
(a) True-class score



(b) Mondrian score

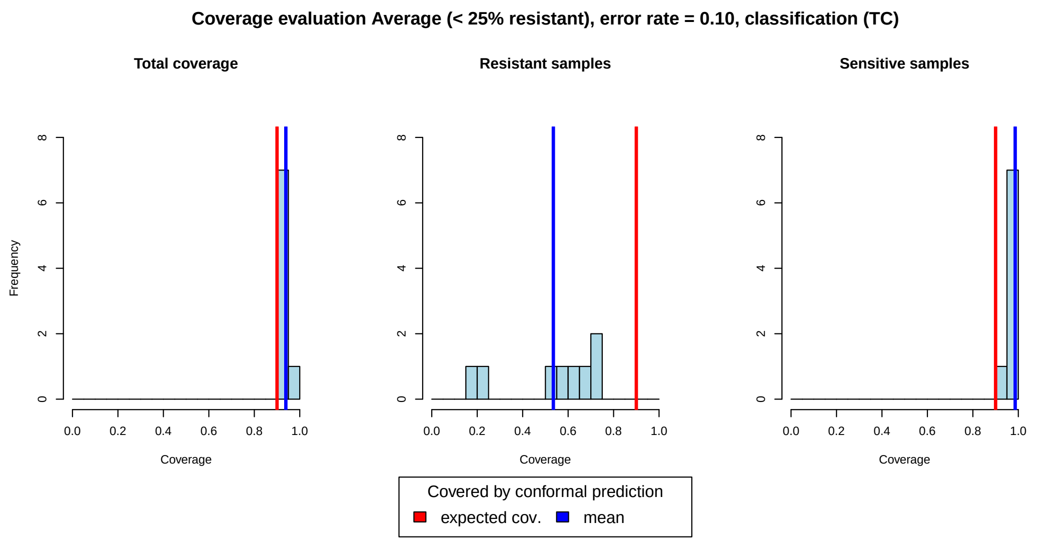


(c) Summation score

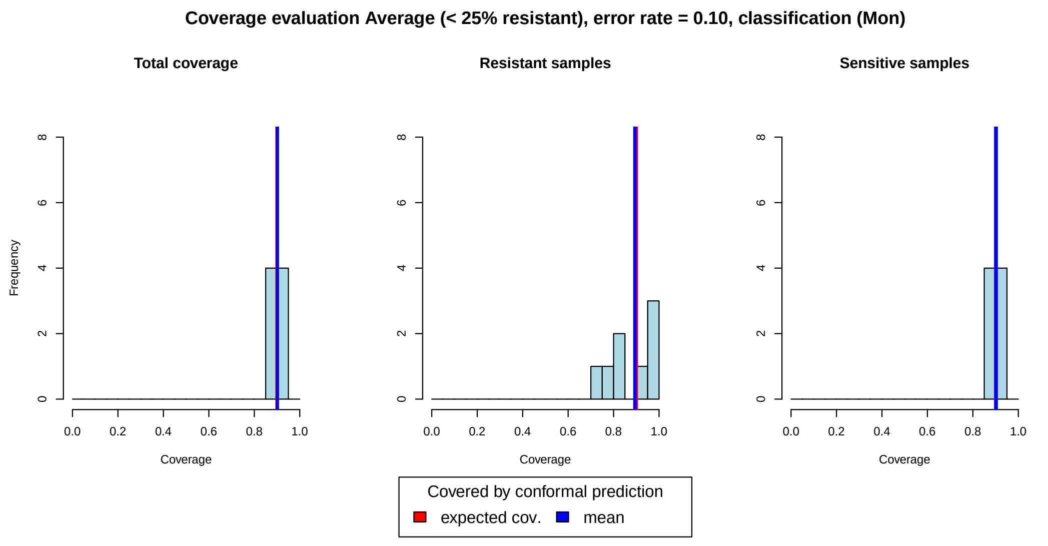


(d) Quantile score

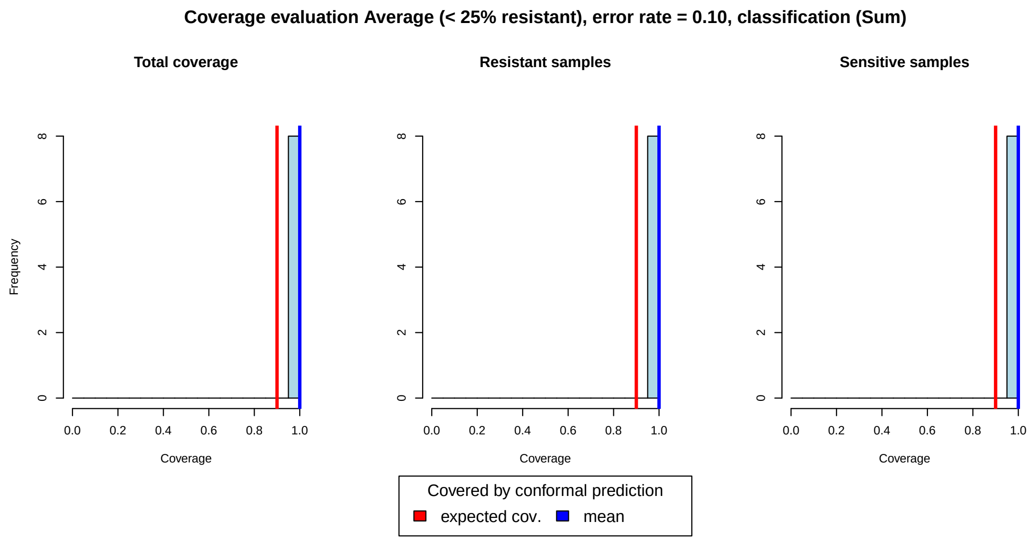
Figure E.66: Coverage evaluation for CP models of twelve drugs from the GDSC1 database trained using CMax viability values and a two-class prioritization setting with underrepresentation of the sensitive class. This figure depicts histograms of the coverage property across CP models for twelve drugs obtained from the GDSC1 database, for which the number of sensitive cell lines was less than 25%. These drugs were used to conduct the drug prioritization analyses. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.



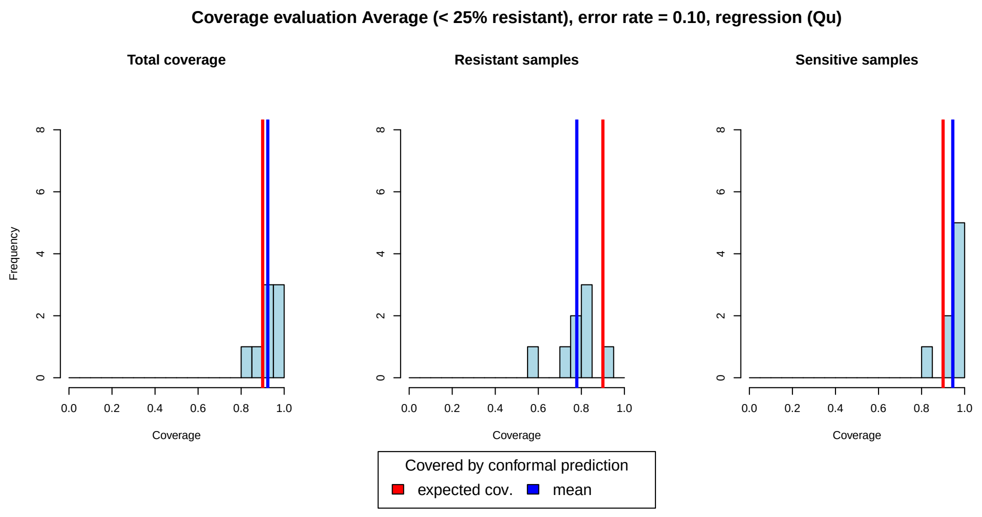
(a) True-class score



(b) Mondrian score



(c) Summation score



(d) Quantile score

Figure E.67: Coverage evaluation for CP models of eight drugs from the GDSC1 database trained using CMax viability values and a two-class prioritization setting with underrepresentation of the resistant class. This figure depicts histograms of the coverage property across CP models for eight drugs obtained from the GDSC1 database, for which the number of resistant cell lines was less than 25%. These drugs were used to conduct the drug prioritization analyses. The left plot in each sub-figure depicts the total coverage, i.e. the fraction of cell lines from the test set of each drug, for which the true response was part of the predicted set / interval. The middle and right plots show the coverage for subsets of resistant and sensitive cell lines for each drug, respectively. The expected coverage of 0.9 for the employed error rate of $\alpha = 0.1$ is shown in red, the actual mean coverage over all investigated drugs is shown in blue. Sub-Figures (a), (b) and (c) depict histograms for the classification setting using the True-class (TC), Mondrian (Mon) and Summation (Sum) scoring functions, respectively. Sub-Figure (d) shows histograms for the regressions setting, where CP was performed using the Quantile (Qu) scoring function.

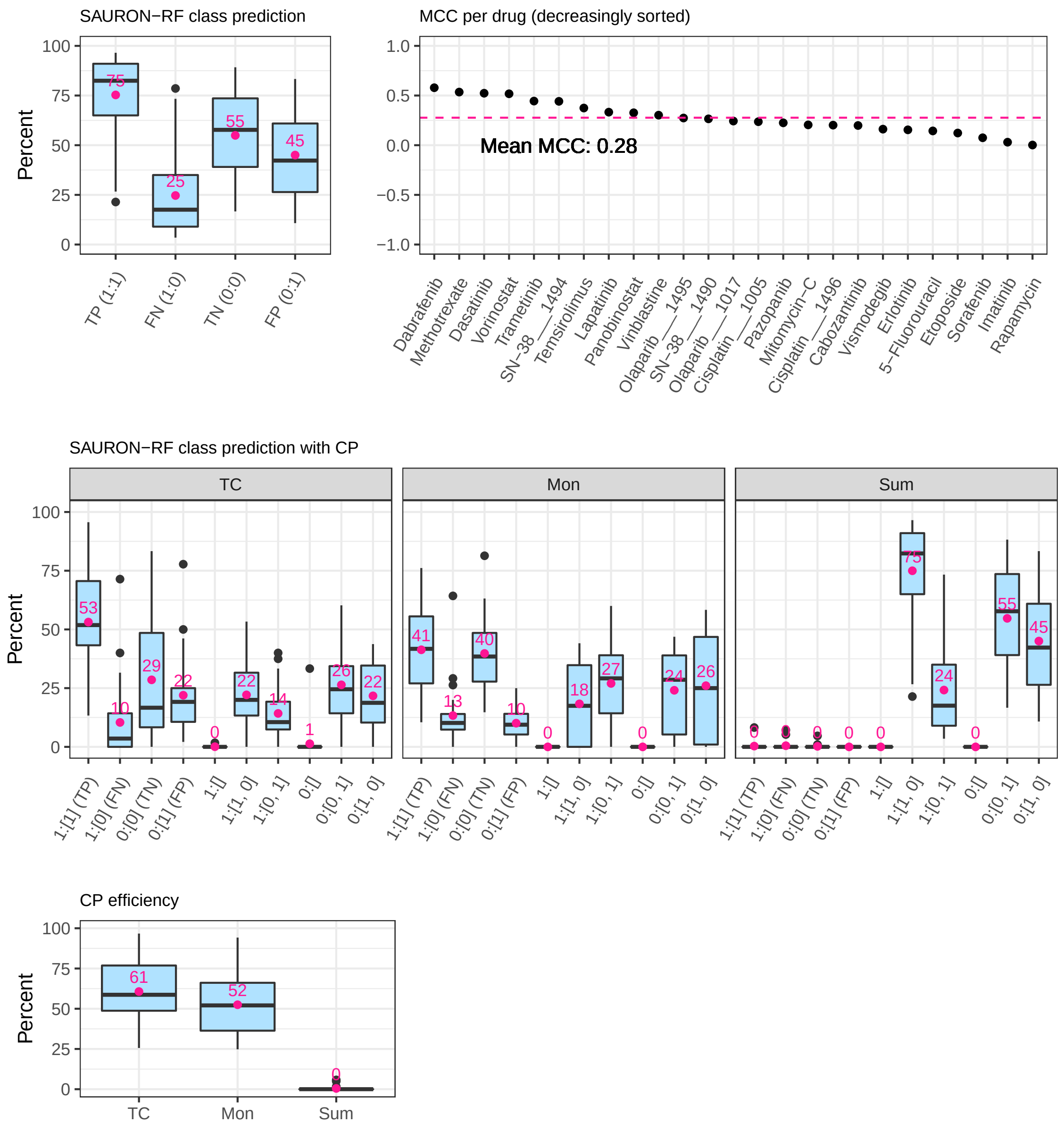


Figure E.68: Classification test set performance for 25 drugs from the GDSC1 database trained using CMax viability values and a two-class prioritization setting. The upper row of this figure depicts the classification performance of SAURON-RF across for all GDSC1 drugs, which were used to conduct the drug prioritization analyses. The middle row shows the effects of CP on the performance in terms of true positive/negative predictions. In the lower row, the CP efficiency is presented.

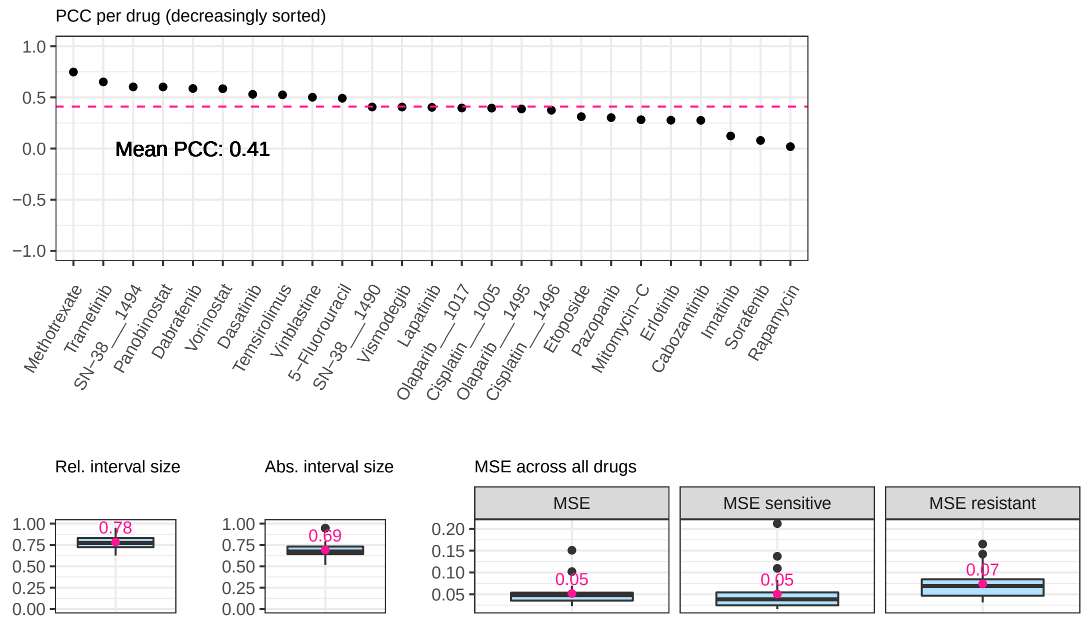


Figure E.69: Regression test set performance for 25 drugs from the GDSC1 database trained using CMax viability values and a two-class prioritization setting. The upper row of this figure depicts the Pearson correlation coefficient between the actual continuous response values and the predicted continuous response values for all GDSC1 drugs, which were used to conduct the drug prioritization analyses. The lower row shows the mean-squared error (MSE) and the interval width of the CP Quantile regression score relative to the spanned training ranges of the drugs.

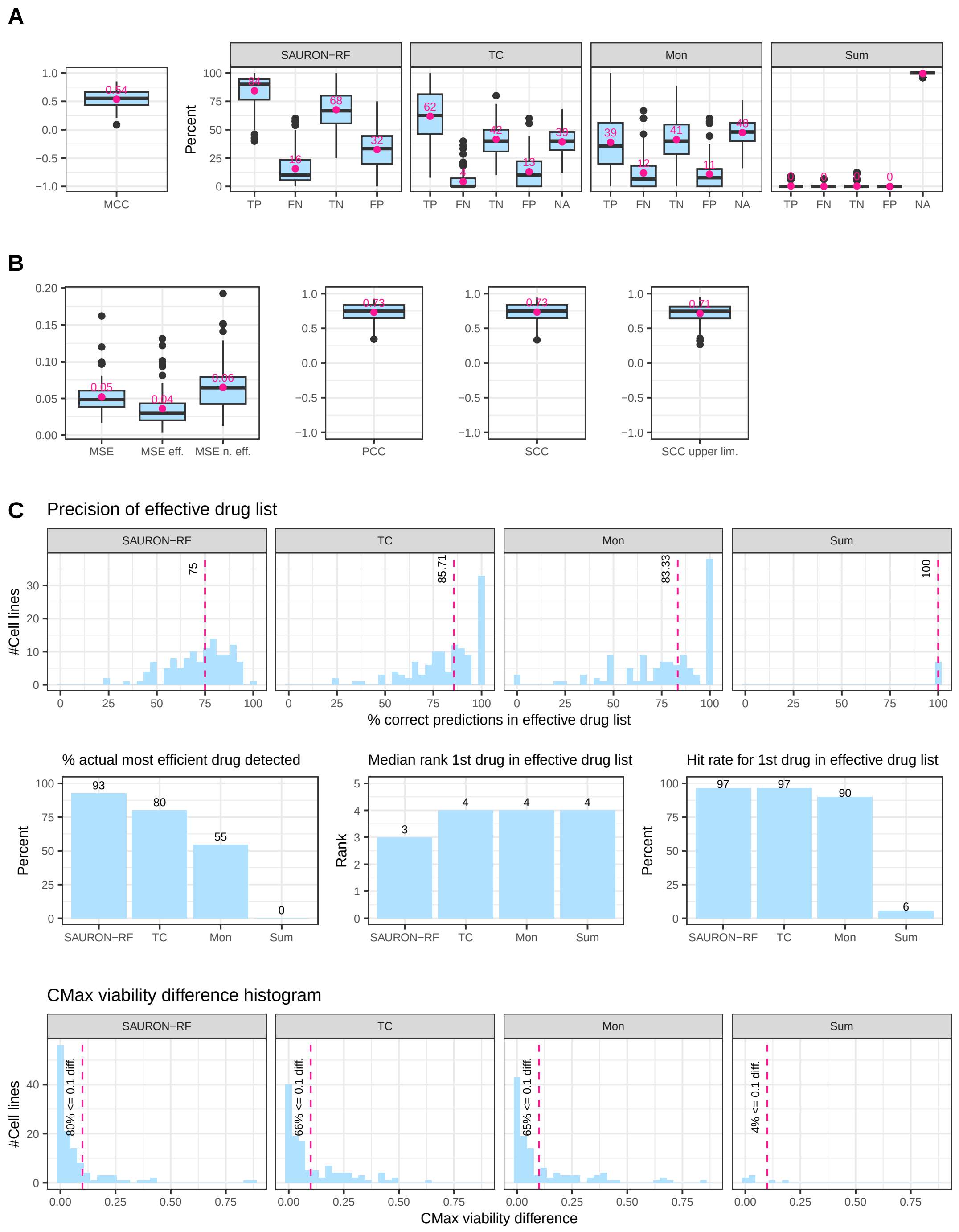


Figure E.70: Prioritization results across all test cell lines of GDSC1. n A, we show the classification performance of SAURON-RF with and without CP. B depicts the regression performance in terms of MSE, PCC and SCC. Here, the MSE is given for the effective drugs, the ineffective drugs, and all drugs. We provide the SCC for the predicted values using SAURON-RF only (SCC) and the upper limit of the CP interval (SCC upper lim.). In C, the upper row depicts the precision of SAURON-RF without (SAURON-RF) and with CP (TC + upper limit, Mon + upper limit, Sum + upper limit). In the middle row, we show the percentage of cell lines for which the most efficient drug was detected, the median rank of the first drug in our effective drug list and the percentage of cell lines for which this prediction was a TP. The CMax viability difference between our first drug and the actual first drug is shown in the lower row.

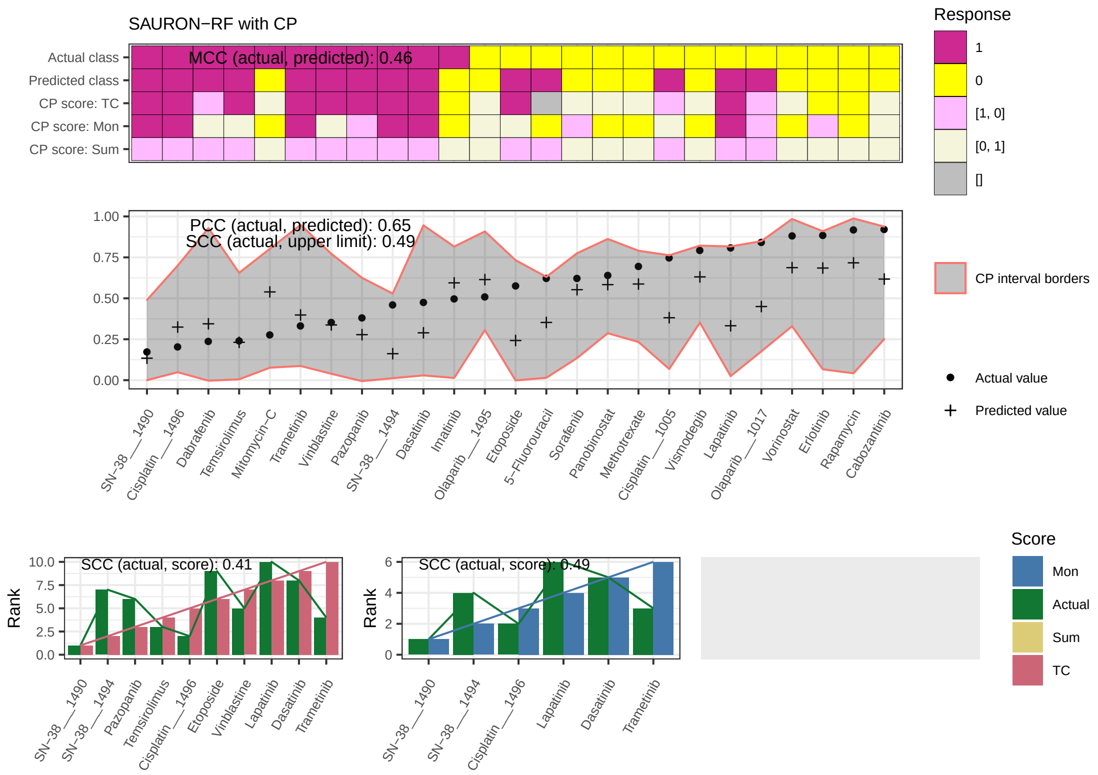


Figure E.71: Prioritization example GDSC1. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 753532) from the test set of the GDSC1 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

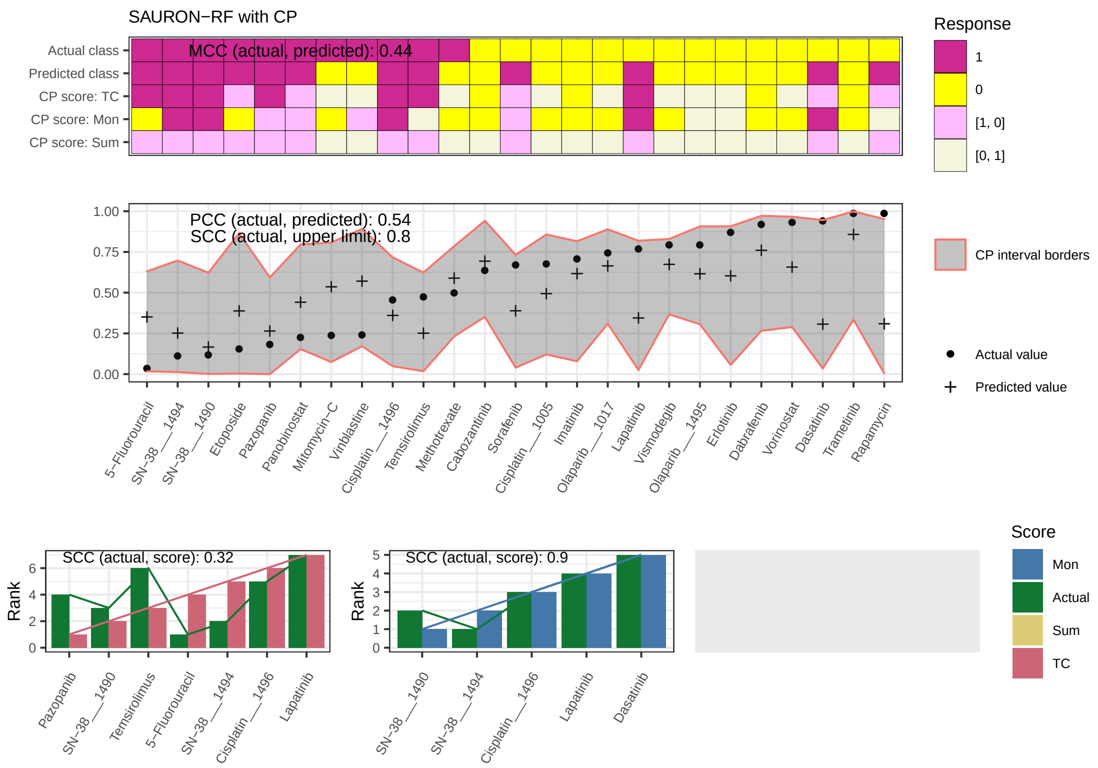


Figure E.72: Prioritization example GDSC1. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 905941) from the test set of the GDSC1 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

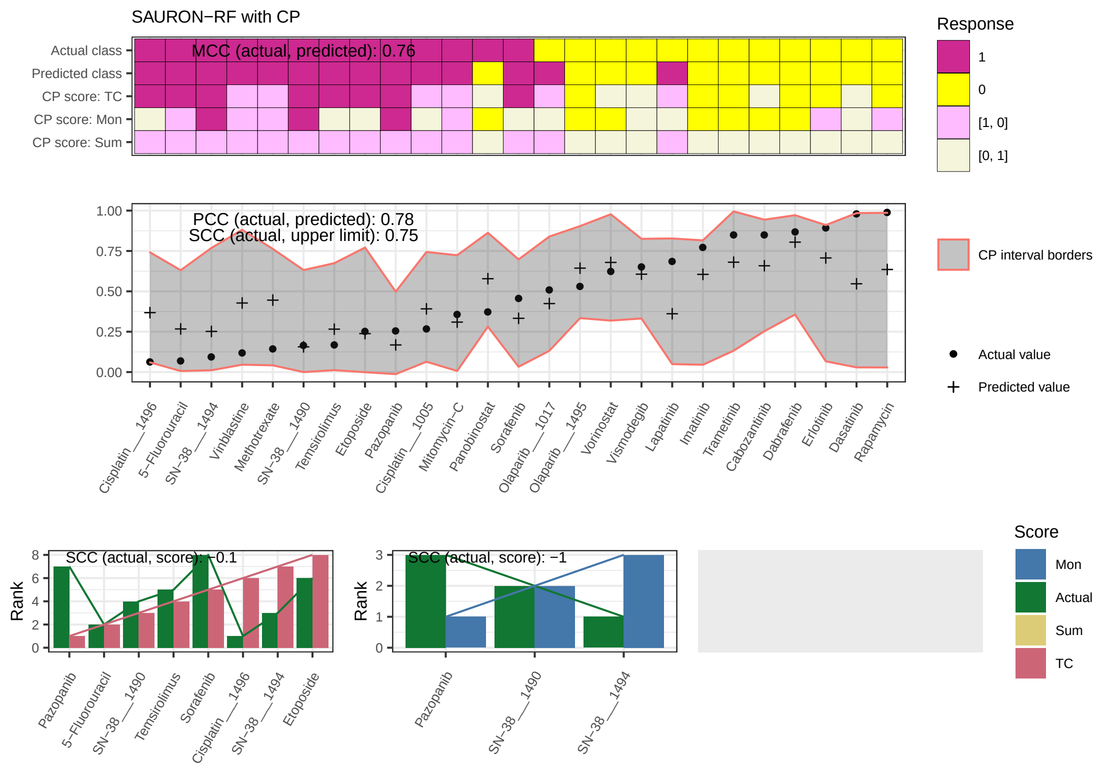


Figure E.73: Prioritization example GDSC1. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 905942) from the test set of the GDSC1 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

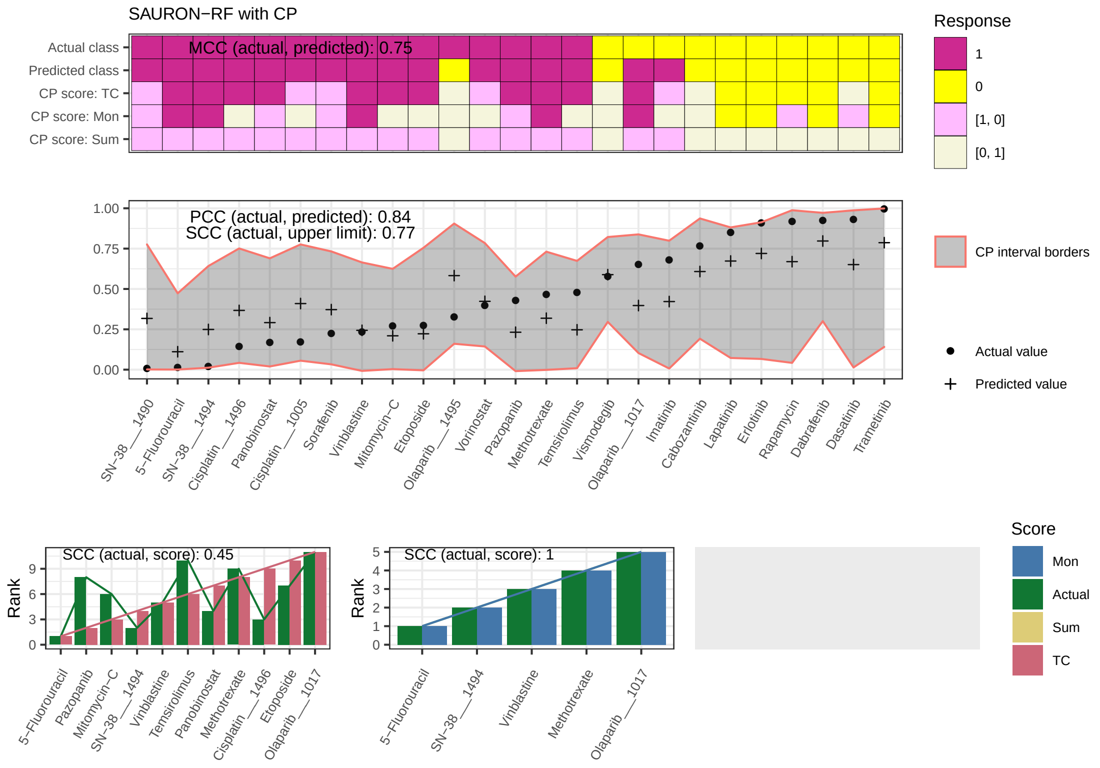


Figure E.74: Prioritization example GDSC1. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 906875) from the test set of the GDSC1 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.

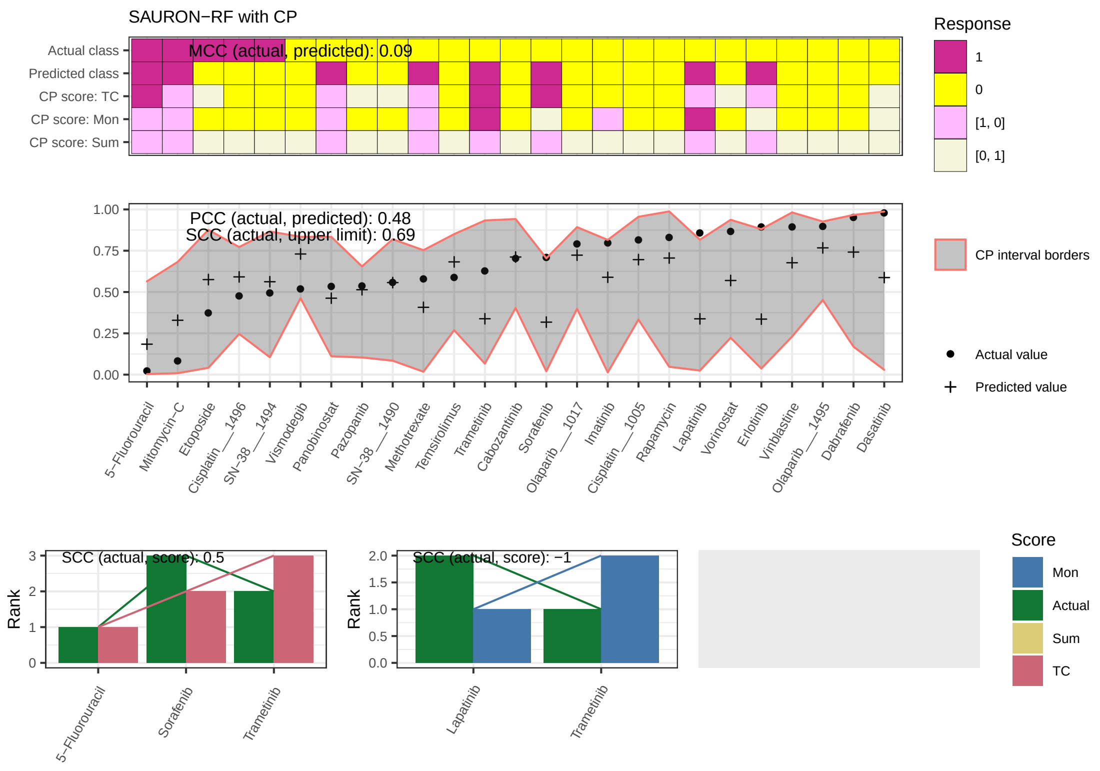


Figure E.75: Prioritization example GDSC1. This figure exemplifies the performance of our prioritization pipeline when applied to one particular cell line (COSMIC ID 917486) from the test set of the GDSC1 data set. The upper plot visualizes the classification performance with and without CP for all analyzed drugs. The middle plot depicts the regression result for all drugs, including the 90% CP interval, and the lower plot shows the resulting prioritized drug lists with the drugs ascendingly sorted by their upper CP limit prediction.