WARSAW UNIVERSITY OF TECHNOLOGY

DISCIPLINE OF SCIENCE: INFORMATION AND COMMUNICATION TECHNOLOGY

FIELD OF SCIENCE: ENGINEERING AND TECHNOLOGY

Ph.D. Thesis

Katarzyna Woźnica, M.Sc.

Towards Trustworthy Automated Data Science

Supervisor:

Prof. dr hab. inż. Przemysław Biecek

WARSAW 2024

Acknowledgments

Along my scientific path, I have received great support from many people. It is difficult for me to list them all here, but I would like to thank my family, friends, colleagues and students of the MINI PW department with whom I had the pleasure to work.

Most of all, I would like to thank my supervisor, Professor Przemysław Biecek, for showing me what scientific work looks like and for creating the conditions for my scientific and personal development in the MI2 group. Thank you for our meetings and discussions and for sharing your knowledge.

Many thanks to all the members of the MI2 group for the interesting discussions during the seminar, the opportunity to learn from each other, and all the pleasant moments over coffee and tea. Special thanks go to Anna Kozak, who gave me a lot of support and help, especially during the last phase of my studies.

I would like to thank my parents, who constantly encourage me to explore new areas and are always there for me. I would also like to express my gratitude to "509"friends, especially Kamila, Milena, Luiza, and Pasza. Their support helped me keep my balance and not lose myself in my academic work.

Abstract

With the growing demand for predictive models, the need for automation in traditional, resource-intensive machine-learning tasks has increased. Automated Data Science (AutoDS) aims to streamline this process, making advanced algorithms more accessible and assisting experts with complex aspects of pipeline generation. However, AutoDS faces significant challenges in developing innovative methods for Data Exploration, Data Engineering, Model Building, and Exploitation. Beyond creating new algorithms or frameworks, there is a crucial necessity to build user trust in the automation system.

This thesis addresses trust improvement in AutoDS systems from two viewpoints. The first involves clarifying and expanding the evaluation of pipeline building steps, while the second includes incorporating domain knowledge in model building. Both issues are frequently raised by AutoDS users.

In the first part of the thesis, we tackle the challenge of broadening the evaluation of AutoDS stages, specifically in Data Engineering, Model Building, and Exploitation. We introduce a unique contribution - benchmarking imputation techniques. Additionally, we provide a methodology for employing eXplainable Artificial Intelligence (XAI) to improve the validation of meta-features in meta-models for hyperparameter optimization. The presented methods offer deeper insight into the impact of the imputation technique and the chosen meta-measure representation on model performance. Furthermore, we integrate the Elo-based predictive power (EPP) meta-score into the model performance benchmark for the OpenML repository. EPP provides a probabilistic interpretation of model performance differences, facilitating model evaluation for non-experts.

The second part of the thesis introduces methodologies for incorporating domain knowledge into Model Building and Data Exploration. The consolidated learning methodology demonstrates how domain-specific hyperparameters can enhance model performance compared to generic ones. Conversely, Semantic Feature Net (SeFNet) proposes enriching understanding and interactions with domain experts by integrating ontologies into Data Exploration.

The presented results and methods advance the AutoDS field comprehensively, addressing all four subfields. This thesis is of significant research importance as user trust building in AutoDS systems has been largely overlooked, with the focus primarily on well-defined problems such as model selection. This dissertation aligns with the new

paradigm of human-centered AutoDS.

Keywords: automation, machine learning, Automated Data Science (AutoDS), human-centered AutoDS, trustworthy AutoDS

Streszczenie

Wraz z rosnącym zapotrzebowaniem na modele predykcyjne, wzrosła potrzeba automatyzacji tradycyjnych, zasobochłonnych zadań uczenia maszynowego. Automated Data Science (AutoDS) ma na celu usprawnienie tego procesu, czyniąc zaawansowane algorytmy bardziej dostępnymi i pomagając ekspertom w złożonych aspektach generowania procesów. AutoDS stoi jednak przed poważnymi wyzwaniami w zakresie opracowywania innowacyjnych metod eksploracji danych, inżynierii danych, tworzenia modeli i eksploatacji. Oprócz tworzenia nowych algorytmów lub frameworków, istnieje kluczowa konieczność budowania zaufania użytkowników do systemu automatyzacji.

Niniejsza rozprawa dotyczy zwiększenia zaufania do systemów AutoDS z dwóch perspektyw. Pierwsza z nich obejmuje wyjaśnienie i rozszerzenie oceny etapów tworzenia pipeline'u, podczas gdy druga obejmuje włączenie wiedzy dziedzinowej do tworzenia modeli. Obie kwestie są często poruszane przez użytkowników AutoDS.

W pierwszej części rozprawy podejmujemy wyzwanie rozszerzenia oceny etapów AutoDS, w szczególności w dziedzinie inżynierii danych, budowania modeli i eksploatacji. Przedstawiliśmy w niej analizę porównawczą technik imputacji. Dodatkowo zaprezentowano metodologię wykorzystania technik eXplainable Artificial Intelligence (XAI) w celu poprawy walidacji meta-funkcji w meta-modelach do optymalizacji hiperparametrów. Przedstawione metody oferują głębszy wgląd w wpływ techniki imputacji i wybranej reprezentacji meta-miary na wydajność modelu. Ponadto, zintegrowaliśmy meta-miarę Elo-based predictive power (EPP) w benchmark wydajności modeli dla repozytorium OpenML. EPP zapewnia probabilistyczną interpretację różnic w wydajności modelu, ułatwiając ocenę modelu osobom niebędącym ekspertami.

Druga część rozprawy wprowadza metodologie włączania wiedzy dziedzinowej do budowania modeli i eksploracji danych. Metodologia consolidated learning pokazuje, w jaki sposób hiperparametry specyficzne dla domeny mogą poprawić wydajność modelu w porównaniu z hiperparametrami ogólnymi. Z kolei Semantic Feature Net (SeFNet) proponuje wzbogacenie zrozumienia i interakcji z ekspertami dziedzinowymi poprzez integrację ontologii z eksploracją danych.

Przedstawione wyniki i metody kompleksowo rozwijają dziedzinę AutoDS, odnosząc się do wszystkich czterech poddziedzin. Ma to duże znaczenie badawcze, ponieważ

budowanie zaufania użytkowników w systemach AutoDS zostało w dużej mierze pominięte, a skupiano się głównie na dobrze zdefiniowanych problemach, takich jak wybór modelu. Niniejsza rozprawa wpisuje się w nowy paradygmat AutoDS skoncentrowanego na człowieku.

Słowa kluczowe: automatyzacja, uczenie maszynowe, Automated Data Science (AutoDS), Human-centered AutoDS, AutoDS godne zaufania

Contents

Chapter 1

Introduction

The purpose of this thesis is to present challenges and selected methods to address the issue of trust in Automated Data Science (AutoDS). In the following sections, we show the motivation behind automatization and then move to the challenges in the field and the research hypothesis.

1.1 Motivation behind automatization

The growth of data-driven applications and the advancement of algorithms has led to an increased demand for predictive machine learning (ML) models. A significant hurdle is the uncovered need for machine learning experts to keep pace with the rapidly evolving developments in machine learning (ML). Furthermore, despite the complexity of tasks in building an ML pipeline, a significant portion of the work tends to be repetitive. It can consume a considerable amount of time and effort of ML experts, limiting their resources for exploring alternative approaches [224]. Consequently, valuable resources may be underutilized, impeding the discovery of enhanced solutions or deeper insights from available data.

In light of these considerations, there has been a shift towards creating automated methods for constructing machine learning pipelines. This direction is referred to as Automated Machine Learning (AutoML, [98]). The tools developed in this field are designed to empower individuals and organizations, regardless of their technical background, by providing greater accessibility to advanced algorithms and addressing critical aspects of modeling with minimal human intervention. The development of AutoML also benefits the community of machine learning experts since automated tools would permit a re-

duction in the time spent on these repetitive steps, thereby enabling a greater focus on those that require more expert knowledge and human intervention. AutoML frameworks assist experts in fast prototyping initial models, which can serve as baselines for further development.

AutoML frameworks primarily concentrate on training models, encompassing tasks like selecting ML algorithms and hyperparameters optimization. Various packages adopt diverse strategies for constructing pipelines [60, 63, 157, 57]. The initial focus was primarily on generating pipelines for tabular data and the supervised machine learning problem. One of the first frameworks, AutoWEKA [208], utilizes the extensive array of ML algorithms in WEKA [83]. The entire collection of diverse approaches could be observed in the ChaLearn Challenge, which was conducted in two editions, 2015-2016 and 2018. The organizers summarized the results and potential of the various methods in [82], and the winning solution evolved in an Auto-sklearn package [60, 63]. This framework is built upon the Python library scikit-learn [161] and uses the same syntax. Auto-sklearn employs Bayesian optimization for algorithm selection and hyperparameter tuning, supplemented by meta-learning for pipeline optimization. The next popular framework is AutoGluon [57], created by AWS. It adds new quality to generated pipelines, developing ensembles of ML models such as multi-layer stacking. Also, other companies such as H2O [128] and Google [76] propose commercial solutions to AutoML. The initial focus was primarily on generating pipelines for tabular data and the supervised machine learning problem. Over time, the development of frameworks dedicated specifically to Neural Architecture Search (NAS, [56]) also began, which have a slightly different structure than frameworks just for tabular data, focusing on classic ML models like Extreme Gradient Boosting (XGBoost, [36]) or random forest [26]. In the case of neural networks, we can easily parameterize the internal structure of the model by specifying the number of layers and neurons. The main libraries for accelerating the optimization of neural-based models by using pretrained models are AutoKeras [108] and Auto-PyTorch [241]. The multitude of available frameworks, each with different strengths, makes it difficult to make an optimal choice [71].

However, comparing the existing AutoML frameworks with the CRISP-ML cycle [203], we see that they primarily emphasize model building and hyperparameter optimization. The remaining steps are not a key part of these frameworks, so there is still a place to

improve them. Model selection and HPO optimization only account for 18% of the data scientists' time, as data engineering and development are the most time-consuming [6]. Similar conclusions are drawn from the user studies where interviewees appreciate the support from AutoML tools but also point that data mining and domain knowledge specific to the prediction problem is also important [72, 51, 45]. Participants expressed a desire to delve deeper into data exploration and seek guidance from domain experts. They believed this approach could have led to the development of improved models and enhanced confidence in the generated model. One participant stated that *"I should consult with domain experts or medical professionals to grasp the data and its characteristics... I must also consider ways to devise new features for enhancing the model"* [51]. This shows that many needs have not yet been addressed by the field of AutoML, and it is necessary to broaden the perspective beyond model training. This is how the concept of Automated Data Science is established.

1.2 Automated Data Science

Automated Data Science (AutoDS) involves not only the task of efficiently constructing models but also delves into the complexity of Data Engineering, Data Exploration, and Exploitation [47]. These subfields differ in the degree to which the problem is formalized and defined. Some of them are more open problems and require human intervention. AutoDS operations also differ in universality, which is understood as using algorithms independent of the domain in which they are applied. The main characterizations and challenges facing each of these subfields are presented below.

Data Engineering involves collecting, processing, and pre-transforming data to ensure it is of high quality and in a format suitable for modeling operations. There are numerous challenges associated with this subfield. One challenge associated with data collection is the issue of data heterogeneity and compatibility [127]. For instance, integrating data from different sources and timespans necessitates monitoring of format consistency [205]. Another important aspect is the multitude of potential data transformation operations. Some introduce new information into the problem by creating new variables, but some are necessary for the following ML algorithms to work properly. Examples of the latter include encoding of categorical variables, imputation of missing data, feature selection, and dimension reduction. From this perspective, Data Engineering sig-

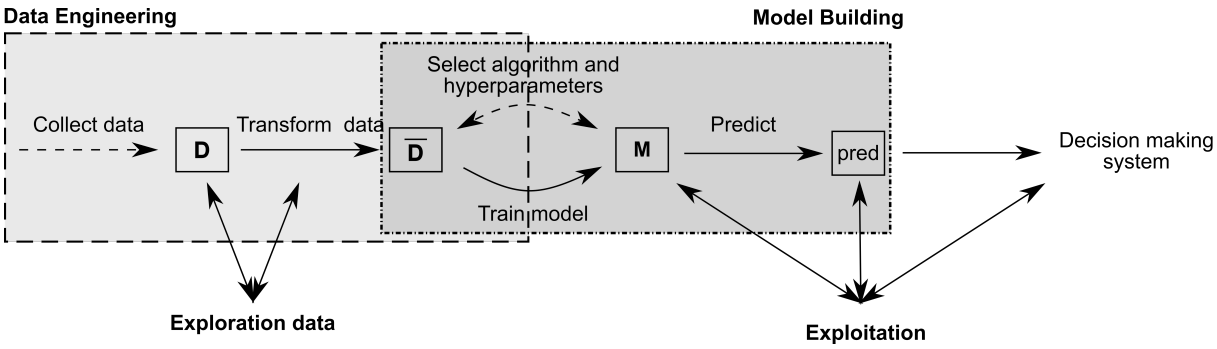

Figure 1.1: Diagram of the AutoDS model pipeline with four subfields marked: Data Engineering, Data Exploration, Model Building, and Exploitation. These subfields are not disjoint; some operations can be included in two areas. Data Engineering and Model Building are more technical and involve applying a broad class of methods. Data Exploration and Exploitation, on the other hand, require more user intervention. Therefore, the schema indicates that these stages of pipeline construction go beyond the sequential nature of AutoDS.

nificantly increases the potential search space of possible operations in the pipeline, as we have to consider preprocessing techniques in addition to the algorithms considered in Model Building [63]. Efficient exploration of these algorithms is a significant challenge. All stages in Model Building and Data Engineering are closely interconnected and are often performed in a loop; once the feedback is received, it is necessary to modify the previous steps. In addition, Data Engineering, despite being more technical, is closely related to Data Exploration.

Data Exploration aims at understanding data features and interpreting their interactions, especially between explanatory variables and outcomes. The foundations of the field are often considered to have been laid by Tukey et al. [210], who emphasized the role of visualization in exploring relationships in data. Data Exploration is a very open-ended problem, and we can refer to any insight coming from data exploration as a pattern [215]. The basic problem is to define what aspect of the designed system is to explore and how it should interact with the human [184, 225]. For example, whether the human should suggest directions for exploration or the system should do it. The next challenge is an assessment of the pattern's interestingness and how to inject this information into the AutoDS system. Moreover, the Data Exploration subfield is very vulnerable to problem specification, and navigating within the dataset may require external domain knowledge, especially in highly specialized applications, such as medicine [2].

Model Building ranges from selecting and training suitable algorithms to optimizing hyperparameters and evaluating their performance. It mainly involves balancing model

complexity and generalization for optimal performance. The greatest progress has been made in supervised learning for tabular datasets [157, 63, 57]. In that field, Model Building has already been automated to an advanced degree, as AutoML represents the beginning of AutoDS development. The research goes beyond classical algorithms and also addresses the problem of building neural network architectures [146, 241]. Other data types, such as time series data and unsupervised learning data, still require further development. Given the high performance of AutoML in numerous applications, future development should also focus on utilizing these systems as sources of information regarding the potential of models. This approach is of particular importance to researchers in the field of machine learning.

Exploitation includes reporting, monitoring, and extracting insights from model behavior and predictions. This is a very open problem, and the AutoDS system can be analyzed at different levels. Firstly, Explainable machine learning techniques give us insight into the rules and patterns of the created decision-making system, and we can interpret its performance and robustness. This is where domain-specific requirements are often crucial. Another aspect is also monitoring the models and their performance. For the results to be relevant to users, we need to remember that we need to communicate these complex results to non-technical audiences.

Figure 1.1 shows a model workflow of preparing a decision-making system with four areas marked. These areas overlap because some of the operations are of a mixed nature. The first stage is Data Engineering, which includes data collection and transformation. Some data transformations may result from the Data Exploration step. Preprocessing Algorithm Selection in Data Engineering can also be seen as part of Algorithm Selection included in Model Building. Analysis of trained models is based on the evaluation of their performance. In Exploitation, it is also assessed, but often, this evaluation is deepened by the more detailed behavior of models and their stability.

Each area mentioned above in AutoDS is a vast topic and requires many technical and algorithmic advances to create comprehensive systems to support users. However, all this work should be aimed at meeting the requirements of users who need to have trust in AutoDS systems [51, 224].

1.2.1 Two challenges of AutoDS to improve trust

This thesis focuses on two aspects significantly affecting user trust: extending the evaluation into the AutoDS system and incorporating domain knowledge.

Evaluating a given operation in the pipeline is closely related to the interpretability of AutoDS processing. Users report that they want to know more details about why operations are chosen [51]. If the problem is well-defined and close-ended, we can quantify the desired effect of a given process. Then, we can perform optimization and choose the most favorable solution. Defining the objective function and conducting the optimization is a desirable property because it allows for relatively easy pipeline construction. This is the case in the subfield of Model Building. If quantification is not possible, the selection process may not be objective and more difficult to implement in AutoDS. In this case, one must focus more on helping the user evaluate different process aspects. Such a need exists in Data Engineering. In both cases, it is worth leaning into the explanation and its accessibility to users to increase their trust in model performance. Providing a broader perspective during evaluation also makes it possible to increase the utility of AutoDS tools as a source of knowledge about ML methods.

The second aspect is the need to take into account domain knowledge. This has been challenging, leading to the predominance of domain-independent methods in AutoML processes, as expert knowledge typically requires interaction with domain experts. While the high universality of these operations is beneficial, users express a desire to consult models with experts to meet specific requirements [72]. AutoDS users also anticipate that integrating domain knowledge will enhance model performance. The solution lies in incorporating domain knowledge and providing more support for interaction with domain experts. Domain knowledge is valuable from the problem-understanding stage through to reporting, ensuring the decision-making system fulfills domain-specific requirements.

It remains an open challenge to combine these two perspectives and provide additional insight alongside AutoDS. However, it affects the willingness to use AutoDS tools and should be addressed in the near future.

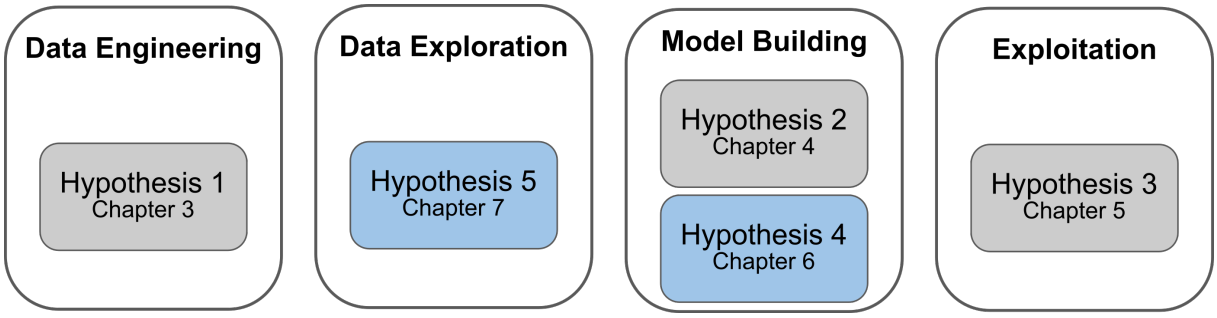

Figure 1.2: The contributions presented in the following chapters are shown in the figure. Each was assigned to one of the four AutoDS sub-domains. All of the research hypotheses can be categorized under one of the two challenges facing AutoDS, namely increasing perspective in the evaluation of the various stages of AutoDS and applying domain knowledge to the AutoDS process. This assignment is indicated in color on the figure. Grey boxes indicate hypotheses assigned to the evaluation part, while blue boxes indicate hypotheses related to the application of domain knowledge.

1.3 Research hypotheses

The objective of this thesis is to better address the need for trust in the AutoDS process. In Figure 1.2, we present five contributions to all four subfields of AutoDS. These contributions extend the perspective of evaluation and introduce additional information about the domain into the system.

1.3.1 Evaluation of the individual stages of the AutoDS

The first part of the thesis focuses mainly on evaluating the components that should be included in the final pipeline created through AutoDS tools. Three papers are included in this part. They touch on data engineering and reporting, as well as the process of selecting algorithms and hyperparameters used in model building. In this section, we formulate three research hypotheses related to each of them.

Hypothesis 1. Validation of preprocessing techniques can simplify the exploration of potential operations within Data Engineering, enhancing the efficiency of data processing workflows.

This hypothesis is addressed in Chapter 3, corresponding to the article "Does imputation matter? Benchmark for predictive models" presented at the *Workshop on the Art of Learning with Missing Values (Artemiss) at the International Conference of Machine Learning 2020* by Katarzyna Woźnica and Przemysław Biecek.

The original solution to the scientific problem is to benchmark data imputation techniques in terms of their effect on the performance of different machine learning models and to conclude that many techniques do not yield high returns compared to the baseline, which is mean imputation.

Hypothesis 2. Leveraging explanatory machine learning techniques extends the validation process for the quality of meta-models and meta-features employed in hyperparameter optimization for tabular data, enabling a deeper understanding of meta-feature importance and its impact on model performance.

This hypothesis is addressed in Chapter 4, corresponding to the article "Towards explainable meta-learning" presented at the *International Workshop and Tutorial on eXplainable Knowledge Discovery in Data Mining (XKDD Workshop) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Database 2021* by Katarzyna Woźnica and Przemysław Biecek.

The original solution to the scientific problem is a methodology for studying the impact of meta-features on the quality of the meta-model using explanatory machine learning techniques.

Hypothesis 3. Integrating the meta-measure with probabilistic interpretation of model performance enhances benchmark validation and provides easily comprehensible insights into model quality, facilitating model monitoring by non-experts in the field.

This hypothesis is addressed in Chapter 5, corresponding to the article "Interpretable Meta-Score for Model Performance" published in *Nature Machine Intelligence* in 2022 by Alicja Gosiewska, Katarzyna Woźnica, and Przemysław Biecek.

The original solution to this scientific problem is the application of an Elo-based Predictive Power (EPP) meta-score that provides a probabilistic interpretation of model performance differences into an OpenML benchmark.

1.3.2 Integrating domain knowledge into the AutoDS

The second part is dedicated to incorporating domain knowledge into frameworks and providing resources to develop new methods addressing this need.

Hypothesis 4. Considering the specificity of the new prediction problem may lead to a greater transfer of hyperparameters compared to transferring from generic datasets like

those in OpenML databases, as tailored hyperparameters can better address the unique characteristics of the problem domain.

This hypothesis is addressed in Chapter 6, corresponding to the article "Consolidated learning: a domain-specific model-free optimization strategy with validation on metaMIMIC benchmarks" published in *Machine Learning* in 2023 by Katarzyna Woźnica, Mateusz Grzyb, Zuzanna Trafas, and Przemysław Biecek.

The original solution to this scientific problem is to propose a consolidated learning methodology for constructing a meta-train set. In consolidated learning, we take advantage of the similar structure of the datasets that are included in the meta-learning for hyperparameter optimization.

Hypothesis 5. Incorporating ontologies into Data Exploration enables the injection of domain-specific information and modeling knowledge from diverse prediction problems, enriching the understanding of data.

This hypothesis is addressed in Chapter 7, corresponding to the article "SeFNet: Linking Tabular Datasets with Semantic Feature Nets" submitted to *Knowledge-based Systems* in 2024 by Katarzyna Woźnica, Piotr Wilczyński, and Przemysław Biecek.

The original solution to this scientific problem is to propose the Semantic Feature Net (SeFNet) methodology, which structures tabular datasets based on the semantic similarity of variables included in the dataset. Semantic similarity is incorporated into the system using domain ontology.

1.4 Structure of the thesis

This thesis is composed of eight chapters. In Chapter 1, we provide background information and motivation behind Automated Data Science. Chapter 2 presents an overview of methods used in AutoDS and applied in the following chapters, presenting original contributions to the topic. Original contributions are discussed in Chapters 3-7 and are divided into two parts. Part I focuses on the evaluation challenges and consists of three chapters. Part II, considering injecting domain knowledge in the decision system, includes two chapters. Chapter 8 concludes the thesis and points out future work directions.

Chapters 3-7 include publications published in journals or presented at a conference.

The relation between the presented article and the corresponding hypothesis from Section 1.3 is outlined in the preface to every chapter.

1.5 Acknowledgments

The work on this thesis is financially supported by the NCN Opus grant 2017/27/B/ST6/0130, NCN Sonata Bis-9 grant 2019/34/E/ST6/00052 and NCBiR grant INFOSTRATEG-I/0022/2021-00. The research was carried out on devices co-funded by the Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) program.

1.6 Additional publications

In addition to the articles presented in this thesis, PhD candidate collaborated with the Polish Lung Cancer Group and the Jagiellonian University Medical College during doctoral studies. This collaboration resulted in 15 publications. Through these publications, the PhD candidate understood the medical domain and the needs reported by those working in this field.

Chapter 2

Formalization and methods in AutoDS

This chapter presents the formalization of specific components of AutoDS and describes the methods applied in the original contributions of this thesis.

AutoDS is difficult to define, but the concept becomes clearer when we outline the goals set for the field. AutoDS aims to support humans in building decision-making systems using machine learning models. Such a system should ultimately operate automatically without human intervention. However, the construction process itself most often requires this intervention; a human must oversee the selection of operations, their effects, and their conclusions. Some stages of the process can be automated, mainly technical work linking successive operations and building the decision-making system as an information system. Other stages of building the system are more difficult to automate because they require taking into account domain knowledge and domain-specific requirements. They are often not expressed in mathematical form, allowing easy integration of these asperities into the system.

2.1 Model Building in the AutoDS

In this section, we focus on formalizing the Model Building subfield. This is the closely defined part of the AutoDS.

2.1.1 Formalization of Model Building

The critical concept in Model Building is the dataset providing information about the considered prediction problems. Because of the desired universality, we should consider

the whole family of multivariate distributions from which datasets can be generated.

Let us indicate as $\boldsymbol{\mathcal{D}}$ family of distributions of tabular datasets. Single dataset $D = (X, Y)$ is a finite sample from the joint distribution $\mathcal{D} = (\mathcal{X}, \mathcal{Y}) \in \boldsymbol{\mathcal{D}}$, where $\mathcal{X} \subset \mathbb{R}^p$ is the p-dimensional feature space and $\mathcal{Y}$ is the target variable space, categorical or numerical. The family of distributions $\boldsymbol{\mathcal{D}}$ is an abstract concept because, in reality, we have no such knowledge and only observe the realization of a specific subset of data distributions. Let $\mathbf{D} = \{D_1, D_2, \ldots, D_m\}$ denote a finite set of datasets, where dataset D_i is the realization of a sampling from the distribution $\mathcal{D}_i$.

Let us denote by $\boldsymbol{\mathcal{A}}$ the family of all algorithms creating machine learning pipelines and single algorithm as $\mathcal{A} \in \boldsymbol{\mathcal{A}}$. The algorithm family is very extensive and includes variety of ML algorithms such as random forest [26], Extreme Gradient Boosting (XGBoost, [36]), Generalized Linear Models (GLM, [143]), Support Vector Machines (SVM, [43]) and neural networks. In machine learning, algorithm $\mathcal{A}$ has an encoded set of rules for finding data patterns and returns the prediction. Finding patterns according to $\mathcal{A}$ is called training and should be performed on training data, but predictions should generalize on unseen data.

Every algorithm $\mathcal{A}$ is dependent on hyperparameter configuration represented as vector $\overline{\lambda} = (\lambda^1, \ldots, \lambda^d) \in \boldsymbol{\Lambda}^{\mathcal{A}} \subset \mathbb{R}^d$, where every λ^i hyperparameter can be discrete or numerical. Hyperparameters have an impact on internal processing and adaptable parameters. Examples of hyperparameters are the number of trees in a random forest or the number of layers in a neural network. The parameters of the ML model are, for instance, weights in neural networks and coefficients in GLM. The user configures hyperparameters but does not directly configure parameters.

In general AutoDS definition, family $\boldsymbol{\mathcal{A}}$ can be seen as product of spaces of all preprocessing methods and ML algorithms $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}_{preproc}} \times \boldsymbol{\mathcal{A}_{ML}}$. In this thesis, we assume that preprocessing methods $\mathcal{A}_{preproc} \in \boldsymbol{\mathcal{A}_{preproc}}$ are trained on the dataset, and they return a modified version of the dataset. A machine learning algorithm $\mathcal{A}_{ML}$ takes as input any dataset and searches for patterns in it according to the given algorithm. For simplicity, we assume that algorithm $\mathcal{A}$ is a superposition of the preprocessing method and ML algorithm for given hyperparameter vector $\overline{\lambda}$, and we denote by $\mathcal{A}(\overline{\lambda}, \cdot)$.

Predictive model trained along with $\mathcal{A}$ algorithm and hyperparameter configuration $\overline{\lambda}$ on dataset sample D we denote as $\mathcal{A}(\overline{\lambda}, D)$. A predictive model is called an already

trained algorithm. Thus, an algorithm trained on different data may result in a different model. Then, the function predicts $\mathcal{P}$ for a given trained model $\mathcal{A}(\overline{\lambda}, D)$ and another subsample for distribution $\mathcal{D}$ return predictions. We denote this as $\mathcal{P}\left(\mathcal{A}(\overline{\lambda}, D), X^{new}\right)$, where X^{new} is a sample of feature space from new subset $D^{new} = (Y^{new}, X^{new}) \sim \mathcal{D}$.

To evaluate the quality of the trained model, we use the quality function $L\left(\mathcal{P}\left(\mathcal{A}(\overline{\lambda}, D), X^{new}\right), Y^{new}\right)$, mapping trained model and dataset to model performance. Evaluation metrics are based on the comparison of $\mathcal{P}(\mathcal{A}(\overline{\lambda}, D), X^{new})$ and actual labels Y^{new}.

Model performance can be expressed as one or more dimensional real value vector. The most common metrics for binary classification are accuracy, F1, or area under curve (AUC). For regression, the most common is Mean Square Error (MSE). Depending on the type of measure, we may be interested in the highest possible value (Accuracy, AUC) or the lowest possible value (RMSE, Error Rate). However, in the following discussion, we assume we want to achieve the highest possible measure. In the case of decreasing measures, the negative value of the measure can be maximized.

For every algorithm, hyperparameter configuration and dataset, we attempt to estimate the expected value of the performance for a random sample of observations given as

$$G(\mathcal{D}, \mathcal{A}, \overline{\lambda}) = \mathbb{E}_{D^{train}, D^{new} \sim \mathcal{D}}\, L\left(\mathcal{P}(\mathcal{A}(\overline{\lambda}, D^{train}), X^{new}), Y^{new}\right). \tag{2.1}$$

Two samples $D^{train}, D^{new} = (Y^{new}, X^{new})$ are drawn independently from the same distribution $\mathcal{D}$. However, in practice, we observe only a finite sample from $D \sim \mathcal{D}$, so we estimate Equation 2.1 using k-fold cross-validation as

$$\tilde{G}(\mathcal{D}, \mathcal{A}, \overline{\lambda}) = \frac{1}{k} \sum_{i=1,\ldots,k} L\left(\mathcal{P}(\mathcal{A}(\overline{\lambda}, D^{train(i)}), X^{test(i)}), Y^{test(i)}\right), \tag{2.2}$$

where $D^{train(i)}$ and $D^{test(i)} = (Y^{test(i)}, X^{test(i)})$ are train and test sample in i fold. Implementation details about cross-validation may be found in [202].

The main objective of hyperparameter optimization for a prediction problem $\mathcal{D}$, is to find λ^* configuration, optimal concerning the expected value of model performance

$$\overline{\lambda}^* = \underset{\overline{\lambda} \in \mathbf{\Lambda}^{\mathcal{A}}}{\arg\max}\ G(\mathcal{D}, \mathcal{A}, \overline{\lambda}). \tag{2.3}$$

So far, we consider one algorithm $\mathcal{A}$ and maximize its performance. However, it is

possible to generalize this concept to Combined Algorithm and Hyperparameter Selection (CASH, [208]), and then we optimize not only the hyperparameters but also the choice of algorithm, i.e.

$$\mathcal{A}^*, \overline{\lambda}^*_{\mathcal{A}^*} = \underset{\substack{\mathcal{A}\in\boldsymbol{\mathcal{A}} \\ \overline{\lambda}\in\Lambda^{\mathcal{A}}}}{\arg\max}\ G(\mathcal{D}, \mathcal{A}, \overline{\lambda}). \tag{2.4}$$

The CASH problem is a double loop, where the outer loop explores the discrete space of algorithms, and the inner loop solves the Hyperparameter Optimization problem. CASH can also be approached as a regular optimization problem where the choice of a classification algorithm is modeled as a categorical variable, and the algorithm hyperparameters are modeled as conditional hyperparameters [208].

The hyperparameter optimization problem presented in this section is a black-box optimization problem and most often reduces to iterative testing of successive configurations of hyperparameters. Hyperparameter optimization methods differ in the way in which successive points are proposed. We denote the finite set of T configurations for the algorithm $\mathcal{A}$ as

$$\Lambda^{\mathcal{A}}_T = (\overline{\lambda}_1, \ldots, \overline{\lambda}_T),$$

where $\overline{\lambda}_i \in \boldsymbol{\Lambda}^{\mathcal{A}}$ for every $i = 1, \ldots, T$. Various methods for selecting hyperparameters $\Lambda^{\mathcal{A}}_T$ are presented in Section 2.1.3.

In the case of numerical hyperparameters, the search space can be infinite, while we are able to test a finite number of configurations.

So, we may not find the best algorithm globally, but we want to find the best possible solution after a small number of iterations. When interrupted, hyperparameter optimization methods should return the best configuration found so far to ensure anytime performance [105].

2.1.2 Average Distance to Maximum

In this section, we define the Average Distance to Maximum to assess the quality of a given hyperparameter optimization strategy S.

Let $\mathbf{D}_{meta-test} = \{D_1, D_2, \ldots, D_m\}$ denote a finite set of datasets, where the dataset D_i is understood as the realization of a sampling from the distribution $\mathcal{D}_i$. Some hyperparameter selection strategies are themselves algorithms using meta-learning described

in more detail in Section 2.1.4. They should be tested on an independent collection of datasets to check the generalizability of these strategies. Hence, we call this dataset a *meta-test*.

We assume that strategy S returns a finite set of hyperparameters for each dataset D_i dataset from $\mathbf{D}_{meta-test}$ and denote as $S(\mathcal{A}, D_i) = \Lambda^{\mathcal{S},\mathcal{A},i}$. Some strategies, such as Bayesian optimization, may provide different collections for every dataset D_i.

Then, the Average Distance to Maximum (ADTM) in k iteration is defined

$$ADTM(\mathbf{D}_{meta-test}, S, \mathcal{A}, k) = \frac{1}{|\mathbf{D}_{meta-test}|} \sum_{D_i \in \mathbf{D}_{meta-test}} \min_{\overline{\lambda} \in \Lambda^{\mathcal{S},\mathcal{A},i}_{1:k}} \frac{l^i_{max} - l^i(\overline{\lambda})}{l^i_{max} - l^i_{min}}, \tag{2.5}$$

where $l^i(\overline{\lambda}) = L\left(\mathcal{P}(\mathcal{A}(\overline{\lambda}, D^{train}_i), X^{test}_i), Y^{test}_i\right)$ standing for the model performance for test sample of D_i dataset is scaled with the difference between $l^{max}_i = \max_{\lambda \in \Lambda_{\mathcal{A}}} L\left(\mathcal{P}(\mathcal{A}(\lambda, D^{train}_i), X^{test}_i), Y^{test}_i\right)$ and l^{min}_i defined analogously. Moreover, $|\mathbf{D}_{meta-test}|$ indicates the cardinality of meta-test set, and $\Lambda^{\mathcal{S},\mathcal{A},i}_{1:k}$ returns k-first elements of $\Lambda^{\mathcal{S},\mathcal{A},i}$.

ADTM determines how far we are from the optimal solution for a given algorithm after running k iterations of optimization for the entire set of datasets. Since the results for different datasets may be on a different scale, we use appropriate scaling.

Most often, ADTM is used not for a specific optimization step but for the whole series of, for example, the first 100 iterations. The results are mainly compared on plots.

2.1.3 Hyperparameter optimization methods

This section presents various hyperparameter optimization (HPO) methods for a specific machine learning algorithm, denoted as $\mathcal{A}$.

For dataset D, optimization strategies involve conducting trials with a finite sequence of hyperparameter values $\Lambda^{\mathcal{A},D}_T = (\overline{\lambda}_1, \ldots, \overline{\lambda}_T)$, where T is the number of iterations, which depends on the available budget in terms of time or computational resources. This finite set of hyperparameters should provide the best possible performance model according to the Equation 2.2. Hyperparameter optimization methods differ in selecting configuration points $\overline{\lambda}_i$.

Generally, we can divide HPO methods into two groups:

1. Offline methods involve conducting iterations of optimization and testing new solu-

tions independently. The knowledge of previously tested solutions does not influence the selection of subsequent points. This independence makes these methods easily parallelizable. Offline methods are highly versatile and can be easily compared across different experiments since the set of tested points can remain consistent.

2. Online methods are influenced by previous iterations of optimization. The selection of new candidate points is iteratively updated based on the performance of points of earlier iterations. Online methods need to balance exploring the hyperparameter domain and focusing on subspaces yielding good performance.

Offline methods

The most known offline methods are grid search and random search [12]. These methods are based on testing a finite number of hyperparameter configurations selected independently of each other.

Both methods are entirely independent of the dataset, and optimization must start from scratch without knowing which hyperparameters may impact the model performance most. So, many optimization iterations are often needed to find near to optimal solutions. In addition, as offline methods, they do not use the information obtained in the earlier runs, namely which model algorithm settings resulted in a good performance model. However, their advantages are ease of usage and parallelization of computation.

Grid search

Grid search (GS) uses a predefined set of values for every hyperparameter λ_i^k. Then, every combination (the Cartesian product) of predefined hyperparameter values is evaluated, and the best configuration is selected. In this method, users usually define fixed values as a set of discrete values for every hyperparameter. If it is a continuous hyperparameter, the most common way is to use evenly distributed points from a fixed interval. Sometimes hyperparameter value transformations are used to provide better coverage of the search space; for example, if the range of values is extensive, it is worth considering an exponential transformation of the points [170].

This approach suffers from the curse of dimensionality because the number of required function evaluations grows exponentially with the dimension of the configuration space. When the optimization budget can be increased, or a satisfactory solution has not been

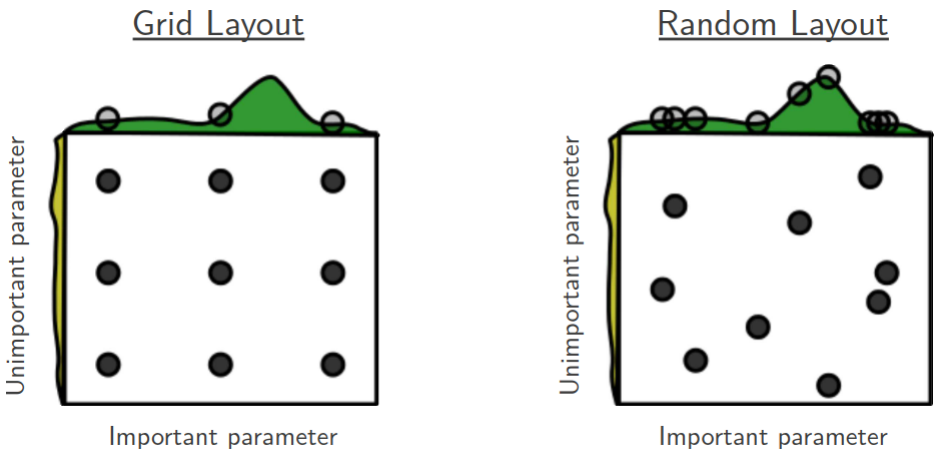

Figure 2.1: Taken from [12]. Comparison of grid search and random search for two-dimensional hyperparameter space. The hyperparameters on the X-axis significantly impact the model's quality, whereas those on the Y-axis have a minor effect. In grid search, using all possible combinations of a fixed set of hyperparameters results in testing only three values of an important hyperparameter. Random search, with the same number of configurations tested, allows for testing up to nine different values of this crucial hyperparameter.

found, increasing the resolution of discretization significantly raises the number of function evaluations needed. Adding only one value for a particular hyperparameter means that hundreds of configurations often have to be tested. What is essential is that grid search must be carried out holistically because interrupting it in progress can cause some of the hyperparameters to go unchecked. The simple solution is to randomize the order in which the values for each hyperparameter are changed to introduce variation for all hyperparameters fairly evenly.

Random search

In random search (RS), we do not define fixed values, but we define the marginal distributions from which values are sampled

$$\lambda_i^k \sim \boldsymbol{\lambda}^k, \tag{2.6}$$

where $\boldsymbol{\lambda}^k$ is a priori defined marginal distribution for k-configuration of hyperparameter. These distributions must be specified by users. The most popular choices are uniform distribution from fixed intervals for numerical hyperparameters or finite collections for categorical hyperparameters. If the numerical hyperparameter is not bounded, the exponential distribution is often chosen [170].

The most significant change from grid search is the independent sampling of each

hyperparameter value in a configuration [12]. This change makes it possible to search the hyperparameter space more efficiently than grid search when some hyperparameters have little impact on the performance of the models. A diagram showing an example of a set of hyperparameters for grid and random search is shown in Figure 2.1. The hyperparameters on the X-axis have a large effect on the quality of the model, while the hyperparameters on the Y-axis have a small effect. In grid search, only three values of a significant hyperparameter are tested by using all possible combinations of a fixed set of hyperparameters. In random search, with the same number of configurations tested (and therefore the same optimization cost), we test as many as nine different values of this hyperparameter.

Furthermore, unlike grid search, random search offers flexible resource allocation. This flexibility allows for adding any random hyperparameter configuration to a random search design while still maintaining the integrity of the random search approach.

Grid search and random search are straightforward to use and have also found wide applications in research. According to [22], most survey respondents answered that these are the techniques they use to find hyperparameters. Random search is also integrated into AutoML frameworks such as H2O AutoML.

Bayesian optimization

Bayesian optimization-based methods are the most known examples of online methods, sometimes called Sequential Model-Based Optimization (SMBO, [96]). In Bayesian optimization, the objective is to benefit from exploring new, previously unexamined regions while sampling from those with a known history and demonstrated efficacy [96, 13, 197].

This Bayesian optimization consists of two functions. One is a surrogate model, the purpose of which is to specify the quality of the model on unseen values of hyperparameters. The surrogate model should also provide an uncertainty of the estimated value. This uncertainty is supposed to be high in unexplored regions but low close to already checked points. The second component is the acquisition function, which balances estimated model performance and its uncertainty coming from the surrogate model to propose a new configuration for validation. Thus, acquisition function trade-off exploration vs. exploitation of hyperparameter space. Most often, the exploration component has more weight at the beginning, but over time, the exploitation of promising subspace

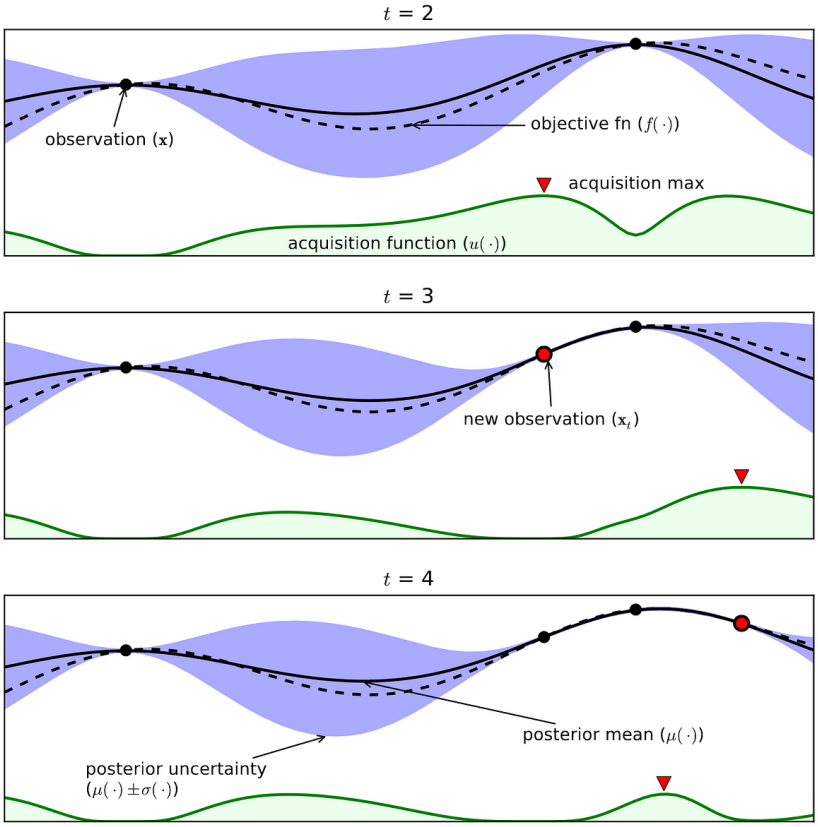

Figure 2.2: A demonstration of Bayesian optimization on a 1-dimensional maximization problem taken from [192]. The figures illustrate a surrogate model, specifically a Gaussian process, across four iterations of sampled objective function values. Each panel's bottom section displays the acquisition function. This function peaks where the surrogate model forecasts a high objective (exploitation) and where the prediction uncertainty is high (exploration). Areas exhibiting both characteristics are prioritized for sampling. The region on the far left remains unsampled because, despite its high uncertainty, it is expected to provide minimal improvement over the highest observed value.

becomes more prevalent.

The next point to be checked is selected as the point maximizing the acquisition function. After checking the actual value of the objective function at this point, it is included in the recorded history, based on which the surrogate model is trained. So, in Bayesian-based optimization methods, the set Λ_T is selected at runtime, and configuration λ_i is determined by an acquisition function which is based on the model performance for the preceding values $\overline{\lambda}_1, \ldots, \overline{\lambda}_{i-1}$.

The performance of Bayesian optimization is mainly affected by choice of surrogate model and the acquisition function [134]. In the case of the surrogate model, it is essential that it estimates the uncertainty of the estimated values. The best-known model that meets such a condition is the Gaussian Process (GP, [174]). However, Gaussian Processes are only suitable for continuous hyperparameters. In addition, they scale poorly, which is

a major practical limitation because many ML algorithms have a multidimensional space of hyperparameters. Therefore, another surrogate model is the Tree Parzen Estimator (TPE, [13]). This model addresses the main weakness of GP because it is adapted to discrete hyperparameters and scales to high-dimensional spaces. TPE implementation is available in the most known package for Bayesian optimization – SMAC [135], which is used in Auto-sklearn and AutoWEKA.

Also, the acquisition function can be specified in different ways. The Probability of Improvement (PI) is the simplest and measures the likelihood that a new point will improve upon the current best observation. Expected Improvement (EI, [113]) extends this approach and quantifies the expected gain from sampling at a particular point. Alternatives are Upper Confidence Bound (UCB) and Thompson Sampling [34]. Each acquisition function has unique characteristics and applications: PI tends to be more exploitative, EI balances exploration and exploitation effectively, UCB allows for a clear trade-off between the two, and Thompson Sampling is advantageous in high-dimensional spaces.

Optimization efficiency is also affected by initial conditions such as the number and values of starting points and the distribution from which hyperparameters are drawn. Hvarfner et al. [99] show that it is possible to modify the distribution to express users; belief in the dependence of model performance on the value of the hyperparameters. In later iterations, this distribution is updated and improved depending on the results obtained in the optimization.

2.1.4 Meta-learning

The optimization methods presented so far work only on the basis of information from a single dataset. It is natural to use the knowledge from previous experiments to focus on hyperparameters that have the greatest impact. Using information from previous experiments to improve the pipeline-building process is the essence of meta-learning [217]. In this section, we focus on using meta-learning in meta-task of optimizing hyperparameters for ML algorithms operating for tabular datasets.

For tabular data, most meta-learning methods have a two-step structure [111]. In Figure 2.3, we show the schema of the procedure. The first step is finding the expressive representations of each dataset $\mathcal{D}_i$, and we refer to this meta-extractor as function $\phi : \boldsymbol{\mathcal{D}} \to \mathbb{R}^k$, where k stands for the size of the representation.

Figure 2.3: Schema of high level intuition behind meta-learning for tabular datasets and HPO. The first step is finding the representations of each dataset by meta-extractor ϕ. The second step is the meta-model θ, which, based on the given representation, aims to provide information about the quality of the hyperparameters.

The second step is the meta-model θ, which, based on the given representation, is to solve the considered meta-task. In the case of the hyperparameter optimization meta-task, the meta-model aims to provide information about the quality of the hyperparameters. This information can take various forms, e.g., a prediction of the performance model for a given configuration or a proposal for a finite set of configurations to be checked. In the first case, the meta-model performs based on the dataset representation and hyperparameter configurations predicting the model performance $\theta : \mathbb{R}^k \times \mathbf{\Lambda}^{\mathcal{A}} \to \mathbb{R}$. In the second, however, based on the representation, the meta-model returns a finite set of hyperparameter configurations $\theta : \mathbb{R}^k \to \{\mathbf{\Lambda}^{\mathcal{A}}\}_{j=1}^{T}$.

Meta-learning most often defines at least two sets of datasets. One of them, $\mathbf{D}_{meta-train}$ is used for meta-model training, while $\mathbf{D}_{meta-test}$ is used to check the quality of the meta-model. A frequently used method to make efficient use of datasets is an analogous approach to leave-one-out cross-validation, namely, leave-one-dataset-out. In addition, sometimes, when the meta-extractor also needs training, we should consider an additional set of datasets.

Meta-features extractor

For tabular data, finding representations of datasets is a non-trivial challenge and is still an open problem. First, meta-extractor representations should correlate with the quality of trained models with given hyperparameters. The meta-extractor should also operate on datasets of different dimensions [111].

The initial approach in finding representation for tabular datasets is the collection of predefined meta-features described by Rivolli et al. [179]. We can highlight the following categories of predefined meta-features:

1. Simple - basic statistics that summarize the dimension of the data. They include

the number of variables, the number of observations, and the number of numeric or categorical variables. Often, attention is also paid to imbalanced data in terms of both the response variable and individual explanatory variables.

2. Statistics - characteristics of distributions of individual numerical variables. For one variable, statistics include primarily moments of distributions such as mean, variance, kurtosis, and skewness, as well as correlation and covariance between any two variables. Statistical meta-features for a given dataset are deterministic. Also included in this group are measures that determine the predictive power of individual variables, e.g., the accuracy of a model based on a given variable.

3. Information-theoretic based - meta-features that summarize the amount of information transmitted in categorical data. The entropy of the variables represents uncertainty associated with them [189]. Mutual information and joint entropy are used to assess the relationship between features and targets.

4. Model-based - meta-features extracted from trained predictive models. The idea behind these meta-features is to better capture the data's complexity. For multi-dimensional data, statistical meta-features may not express variability sufficiently. However, extracting features from predictive models can capture higher-order dependencies and interactions that can be crucial in finding an appropriate representation. It is advisable to use algorithms that are both simple and fast to train, especially when dealing with large datasets. In practice, the majority of model-based meta-features are derived from decision trees. Some examples of such features include the depth of a decision tree without any pruning or the node count of that same tree.

5. Landmarkers - meta-features that are at a higher level of generality than model-based ones. They stand the performance of these simple models such as Naive Bayes or Decision Tree with default hyperparameters. Landmarkers are a kind of baselines that signal the intrinsic difficulty of a given prediction problem.

Predefined meta-features are available for datasets in the OpenML repository. In addition, to ensure reproducibility and comparability, Alcobaça et al. [3] create the MFE package that computes the given set of meta-features.

Statistical and informatics-based meta-features are mostly calculated for each variable separately. To ensure that the meta-extractor will provide representations of datasets in the same dimension space, regardless of the dataset dimension, it is necessary to aggregate the meta-feature values. The most common aggregation is the mean, but sometimes also minimum or maximum are applied.

The aforementioned set of meta-features has been used in many works regarding the optimization of hyperparameters [230, 229, 60]. However, such a broad collection of meta-features turns out to be necessary because different statistics are informative depending on the considered meta-task. Bilalli et al. [17] showed that different sets of meta-tasks are valid for a classification problem and others for regression.

In addition to predefined meta-features, a new approach to finding representations is to use encoder-based meta-extractors. Then, the meta-extractor optimizes the dataset's representation, and the user most often has to define only the size of this representation. In the case of this type of meta-extractor, the greatest difficulty is to ensure schema-agnosticism so that the meta-extractor works on diverse datasets with different structures. The first such model is Dataset2Vec [111], but also FLAT [240] operates on a similar principle. Dataset representations can also be extracted from other models that work for heterogeneous tabular datasets [101].

Meta-models

Based on how these meta-features are utilized, we may define the following types of meta-learning frameworks.

1. **Sequential surrogate meta-models** employ meta-features to assess the similarity between meta-train datasets and new task. Then, meta-models use these similarity measures as weights to configuration transfer. A meta-learner does not use meta-features explicitly; they are compressed to similarity measures, so it is more challenging to understand the meta-features impact.

   Extensions of SMBO are used to propose new hyperparameters with the potential for good model performance on the new dataset. The MI-SMBO procedure [60] uses the best predicted hyperparameters from similar datasets to select the initial points to optimization for novel data. Two definitions of similarity between meta-features vectors are proposed: the p-norm of the difference between the datasets'

meta-features and a metric that reflects how similar the datasets are concerning the performance of different hyperparameter settings. The meta-features modification is utilized in the main optimization step in a surrogate Gaussian Process (GP) for all tasks simultaneously [234]. To measure similarities between instances, they use a squared exponential kernel and the nearest neighbor's kernel. This approach is extended to become more scalable in Wistuba et al. [230, 231]. They fit separate GPs for each task in the meta-data, then aggregate them into one using the Nadaraya Watson kernel.

2. **Black-box surrogate meta-models** directly predict model performance for given hyperparameter settings and a given dataset described with the meta-feature vector. A meta-learner is a regression model, and meta-data has a tabular format – every dataset is represented as a vector with a sequence of meta characteristics [24]. Firstly, this generic approach is proposed by Giraud-Carrier et al. [73]. The authors provide a knn-ranking method to rank candidate models. Vilalta et al. [219] also consider this approach in the meta-model framework and suggest the potential source of meta-features. Reif et al. [176] use the SVM model as a meta-regressor and utilize a broad scope of meta-features: statistical and information theory measures, model-based features, and landmarkers. For every machine learning algorithm, they train the meta-learner independently because of different hyperparameter configurations. Similar meta-features are applied by Davis and Giraud-Carrier [46], but the meta-learner is a Multilayer Perceptron. Probst et al. [170] extend the idea of surrogate benchmarks and train surrogate regression to map hyperparameter configuration to model performance.

Both perspectives on meta-learning models empirically prove that transferring knowledge from independent tasks is beneficial. The sequential approach requires the validation of each step separately, and it is more challenging to explore interactions between meta-features and hyperparameter configurations. On the other hand, the black box meta-model is very similar to the classic machine learning approach. It, therefore, offers excellent opportunities for analogous model exploration methods.

Hyperparameter portfolio

Hyperparameter portfolio is a particular case of meta-learning since at least one portfolio configuration should parameterize a good-quality model for previously performed experiments and should transfer this good performance to a new dataset. We assume that at least one configuration will be promising for new, unknown data. The data repository, based on which we determine the portfolio, is called a meta-train set, and new target prediction problems are called a meta-test.

A predefined, limited set of hyperparameter configurations optimized for a wide range of datasets has been shown to give comparable results to Bayesian optimization [229, 167] and proves to be even better when considering anytime performance. Moreover, the portfolio approach may be seen as an extension of the default hyperparameter values that is easy to share and parallelize. In the first studies introducing this method, all meta-train datasets have the same relevance for the portfolio composition due to their independent weighting. Therefore, to enhance the impact of meta-learning, Feurer et al. [62] use meta-features (i.e., vectors of dataset characteristics) to evaluate the dataset similarity. Subsequently, during portfolio development, a higher weight is given to a good configuration of hyperparameters from meta-train sets more similar to the new data. This approach combines the online and offline procedures since a static portfolio leverages the most effective configuration for similar datasets, assuming their optimized functions have similar learning curves. The use of meta-train datasets reduces the time for the early iterations in Bayesian methods.

Algorithm 1 Average SMFO

Require: Set of feasible hyperparameter configurations Λ, set of datasets $\boldsymbol{D}$, number of maximal tries T
Ensure: Sequence of hyperparameter configurations to evaluate.
$\overline{\lambda} \leftarrow ()$
while $T > 0$ **do**
$\Lambda^* \leftarrow CANE(\Lambda \setminus \lambda, T, \boldsymbol{D})$
$T \leftarrow |\Lambda^*|$
$\overline{\lambda} \leftarrow \overline{\lambda} \cup \Lambda^*$
end while

The most well-known method of portfolio construction is the A-SMFO algorithm proposed in [229] and presented in Algorithm 1. It is based on the principle of composing a portfolio complementarily according to the Algorithm 2 and repeating this operation

until you get a portfolio of the expected size. The model portfolio performance on a set of datasets is defined as the average of the best model quality for each dataset obtained by the considered models. In each step, we check which configuration of hyperparameters most improves the portfolio performance on a fixed set of datasets $\boldsymbol{D}$, and such configuration is added to the previously selected set. Such an improvement function is denoted as L^* in the algorithm.

Algorithm 2 CANE Optimal Sequence

Require: Set of feasible hyperparameter configurations Λ, set of datasets $\boldsymbol{D}$, number of maximal tries T
Ensure: Sequence of hyperparameter configurations to evaluate.
  $\Lambda_0 \leftarrow ()$
  for $t = 1, \ldots, T$ **do**
    $\lambda_t \leftarrow \arg\max_{\lambda \in \Lambda} L^*(\lambda, \Lambda_{t-1}, \boldsymbol{D})$
    $\Lambda_t \leftarrow \Lambda_{t-1} \cup \lambda_t$
    if every dataset has the best hyperparmeter in Λ_t **then**
      return Λ_t
    end if
    return Λ_t
  end for

2.2 Data Engineering

Before we start building a machine learning model, it is necessary to prepare the data, as the data quality is crucial to the entire pipeline. Data Engineering, which involves the technical aspects of data processing, is time-consuming and laborious, making it a natural target for automation. Due to the multitude of aspects involved in data preparation, De Bie et al. [47] have proposed dividing data engineering into three high-level themes: (1) data organization, (2) data quality, and (3) data transformation. . These themes help structure the approach to handling data, ensuring it is properly organized, high-quality, and appropriately transformed for model building.

ETL (Extract, Transform, Load) is a common term closely related to data engineering, which refers to a process used in data warehousing and integration. Data engineers may utilize ETL processes to ensure that data is correctly collected, transformed, and loaded into the systems they manage. However, data engineering encompasses a broader range of responsibilities beyond ETL, including developing and maintaining data infrastructure, pipelines, and systems for efficient data management and analysis. It often intersects

with data exploration, creating potential feedback loops, but remains primarily focused on providing high-quality input data for models built during the model-building phase.

In this thesis, the primary focus in Data Engineering is on data transformation, which involves preparing the data so that machine learning algorithms can effectively operate on it. Correctly applying transformations is crucial and often challenging, as inappropriate application can lead to poor generalization on new data samples. Preventing data leakage, which can introduce bias, is especially important when working with large datasets. To avoid this, transformations should be trained on training data and only adjusted on new data. Additionally, it is essential to ensure that the target variable is appropriately separated to maintain the integrity of the model training process.

2.2.1 Imputation methods

A significant challenge in the field of machine learning is the handling of eventual data gaps that occur in datasets. This is a common problem across various domains, with a notable example being in medicine. Medical datasets often contain missing values for various reasons, such as patient dropouts, incomplete records, or errors in data collection.

Most machine learning algorithms are not designed to handle missing data natively. As a result, it is necessary to use data imputation methods to fill in these gaps before proceeding with analysis or model training. Without proper handling of missing data, the performance of predictive models can be severely compromised, leading to biased or incorrect conclusions.

Numerous data imputation methods have been developed to address the issue of missing data. These methods are predominantly designed by statisticians who use various techniques to estimate multivariate distributions and propose values for the unobserved data. Some of the most common imputation techniques include:

- Mean/Median/Mode Imputation. Simple methods that replace missing values with the mean, median, or mode of the observed data. These imputations are easy to implement but can introduce bias into data and reduce its variability.

- Hot Deck Imputation [7]. This group of methods involves replacing missing values with observed values from similar records (donors). The various methods differ in how they determine these similar observations, but in most cases, they preserve

the original distribution of the data. Hot Deck Imputation techniques are introduced as an extension of cold deck imputation, which uses values from an external data source, such as historical data. However, hot deck imputation can still be used when the additional data is unavailable or of poor quality.

- Regression Imputation. Regression models are also applied to predict and fill in missing values. For each variable with missing data, we build a regression model (linear or logistic regression) and model the values of that variable in relation to other variables. The user can specify a subset of variables. Such an operation is repeated for other variables with missing data. These methods can introduce bias if the underlying model assumptions are violated but also help maintain the dataset's natural variability.

- Machine Learning-Based Imputation [214]. The assumptions in this case are very similar to those in Regression Imputation, but machine learning models are used here. Often applied algorithms are k-nearest neighbors [65], random forests [26], neural networks, or gradient boosting [36]. These algorithms can capture complex patterns and interactions in the data.

- Expectation-Maximization Algorithm (EM, [182]). EM algorithm is a robust method for imputing missing data by iteratively refining estimates of the missing values. The algorithm operates in two main steps: the Expectation (E) step and the Maximization (M) step. Initially, it starts with an initial guess for the missing values. During the E step, the algorithm calculates the expected log-likelihood of the data, including both observed and missing parts, given the current estimates of the missing values. In the M step, it maximizes this expected log-likelihood to update the estimates of the model parameters. These updated parameters are then used in the next E step to re-estimate the missing values. This process is repeated until convergence. These methods can handle complex data structures and dependencies.

- Matrix factorization methods [86]. This is a class of algorithms for imputing missing data by decomposing a large matrix into the product of two lower-dimensional matrices. They are borrowed from Collaborative Filtering in recommended systems. Here, the original dataset is represented as a matrix with missing values and is ap-

proximated as the product of two matrices: one representing the latent features of the rows (e.g., users) and the other representing the latent features of the columns (e.g., items). Initially, these matrices are imputed with random values. The algorithm iteratively adjusts the values in these matrices to minimize the difference between the observed entries in the original matrix and their approximations.

The basic imputation method is Single Imputation, which performs a single imputation of a missing value according to one of the abovementioned techniques. However, the main problem with this approach is the lack of evaluation of the uncertainty and potential variance of the imputed values. The solution to this issue is to use Multiple Imputation [182], which consists of three steps:

1. generation of multiple pooled versions of the dataset by sampling observations. This approach gives us several versions of the dataset on which the imputation is performed.

2. independent imputation for each dataset,

3. aggregation of the imputed values and evaluation of their variance.

Data imputation techniques are primarily evaluated for the accuracy of the imputed data and compared with known values. In addition to accuracy, the behavior of the distribution of the original data is also evaluated. In both cases, it is best to have information about ground truth values so they are most often evaluated for simulated data.

2.3 Data Exploration and Exploitation

Model Building and Data Engineering have so far been included as an integral part of AutoML frameworks, while Data Exploration and Exploitation have been treated more as independent pre- and post- pipeline building steps performed by users. Moreover, they are the broadest aspects of AutoDS and can include many human interactions with data in the case of Data Exploration and model results in Exploitation. For this reason, descriptions of the aspects of these two subfields that are used in the original results of this thesis are presented in the corresponding chapters.

2.4 Selected methods to improve trust in AutoDS

In addition to methods and techniques typical of AutoML and AutoDS, this thesis employs additional techniques to address the need for greater trust in components of the Automated Data Science process.

2.4.1 Understanding evaluation: Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) methods are increasingly integrated into AutoML frameworks to enhance interpretability in the models they generate. In addition, XAI is increasingly being used to interpret the pipeline generation process with a particular focus on hyperparameter optimization [152, 187]. However, XAI provides more opportunities, and this section discusses XAI methods that can be used as an extension of the evaluation and selection process that occurs in AutoDS.

Variable Importance

One fundamental approach to explaining ML models is through model-agnostic variable importance measures [64]. This class of methods aims to evaluate the importance of each feature in a predictive model to understand which features have the most influence on the model's predictions.

A commonly used technique in this category is Permutation Variable Importance. This method evaluates the importance of a single variable by attempting to remove its effect on the model's estimation. One way to do this is by disrupting the relationship between the target variable and the variable in question by permuting the values of the considered feature.

The Permutation Variable Importance measure depends on the change in the model performance of algorithm $\mathcal{A}$ with hyperparameters $\overline{\lambda}$ before and after perturbations in a subset of variables X_k. Formally, we define the Permutation Variable Importance of variable X_k as

$$vip(k) = \frac{G(\mathcal{D}^{-k}, \mathcal{A}, \overline{\lambda})}{G(\mathcal{D}, \mathcal{A}, \overline{\lambda})}, \tag{2.7}$$

where $\mathcal{D}_i$ is original data distribution and $\mathcal{D}_i^{-k}$ is distribution ignoring the relationship between X_k and (Y, X_{-k}). This distribution is usually a product of marginal distributions of X_k and (Y, X_{-k}) from $\mathcal{D}_i$. Sometimes, instead of the ratio between the model performance before and after permutation, the difference of these two values is applied [16].

In a basic implementation, X_k consists of just one variable; however, it can also be extended to a larger set of variables. In that case, the relationships within X_k are preserved, while the other variables are permuted.

The method of generating perturbed observations, which affects the joint distribution of the data, is often pointed out as a drawback of this approach. However, other approaches that are more robust to disruption of the relationship structure between variables have a significantly higher computational cost [94]. In addition, the importance of selected samples has been raised repeatedly in the context of the data drift. If the model does not generalize well, then the initial model performance will be lower, and thus, the importance of the variables will also appear lower.

In addition to global methods that provide an overall ranking of importance across the dataset, local methods for measuring variable importance are being developed to assess the importance of features to individual predictions. One prominent method is Local Interpretable Model-agnostic Explanations (LIME, [178]), which approximates a model locally with an interpretable model by perturbing the input data around the instance to be explained. Meanwhile, Shapley Additive exPlanations (SHAP, [137]) derives from cooperative game theory and provides a measure of feature importance by assigning the difference between the actual prediction and the average prediction to each feature.

Triplot

Despite various modifications of variable importance measures, whether at the global or local level, it is still unclear how to deal with the structure of correlated variables. Triplot [162] is one method that addresses this problem and supports model analysis by exploiting the information about the correlation between variables. In addition, triplot provides a new visualization method for this multilevel technique.

The whole method consists of two components. The first focuses on capturing the correlation structure between variables. The assumption is that the correlated variables should be grouped together into a set called aspect. Since we do not know what degree

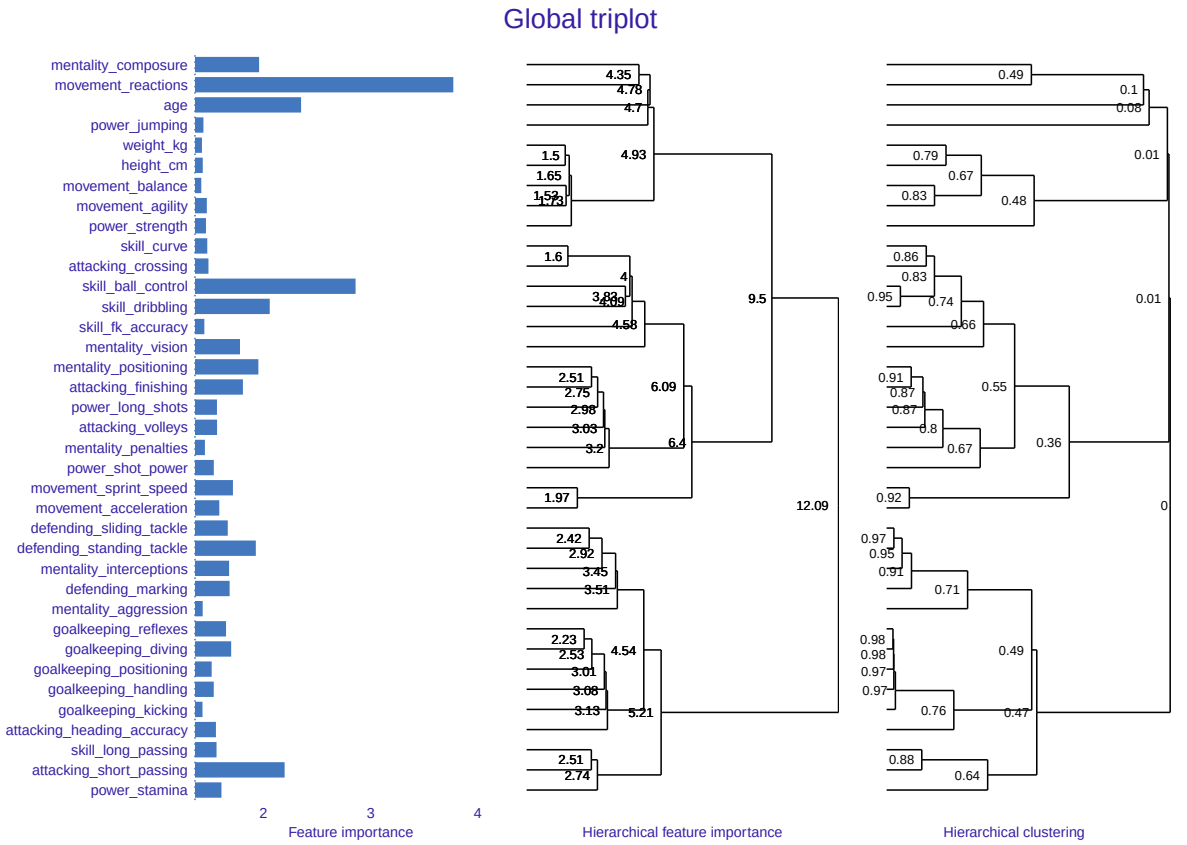

Figure 2.4: Example of the global variable importance in FIFA dataset taken from [162]. The left panel displays variable importance determined by permutation variable importance. The rightmost panel illustrates a hierarchical arrangement of variables, where each level represents aspects formed by variables linked by lines. The central panel depicts the same structure but includes information on the relevance of each aspect.

of correlation is relevant to the model's performance, we do not consider a single cutoff point but perform a hierarchical clustering operation on the variables. This gives us a hierarchical structure of the variables, and we can easily modify the cutoff to determine which variables are correlated. In addition to the groups of variables determined from the correlation structure, this method can be used in domain-specific applications where there is domain knowledge of which variables should be grouped together. For example, in the FIFA dataset, variables can be divided into characteristic sets of soccer skills important in particular positions.

The second component is the evaluation of the significance of the variables. Triplot can summarize both global and local variable importance. In the case of global, it uses grouped Permutation Variable Importance for each grouping into the aspects specified in the clustering. As local importance, it uses a modified LIME method.

To capture all aspects of this model analysis approach, the triplot is composed of

three panels. Figure 2.4 shows an example of the global importance of variables taken from [162]. The left panel shows variable importance based on permutation variable importance. The far right panel shows a hierarchical structure of variables where, at each level, aspects are formed by variables connected by a line. The middle panel shows the same structure but with the aspect's relevance information applied.

Partial Dependence and Individual Conditional Expectation Plots

In addition to summarizing the importance of variables, we can also study how changes in the value of a variable affect the prediction value. The primary global method used for this purpose is Partial Dependence Plots [66], while the local one for individual observations is Ceteris Paribus Plot [16] also called Individual Conditional Expectation Plot [75, 150].

The main concept behind constructing these profiles is to illustrate the expected value of the model's prediction changes as a function of a chosen explanatory variable. Thus, to determine the PDP profile for variable X_k at point z, we must calculate the expected value from the marginal distribution of the other variables as

$$g_k^{PDP}(z) = \mathbb{E}_{X_{-k} \sim \mathcal{D}} \, \mathcal{P}\left(\mathcal{A}, (X_{k|=z})\right),$$

where for simplicity $\mathcal{A}$ denote here already trained model. What is more, vector X_{-k} indicates the random vector with excluded k-th variable, and $X_{k|=z}$ stands for the random vector with fixed k-th variable to value z. The estimation of this expectation value is carried out by averaging the prediction for artificial observations from a marginal distribution with a fixed value of the variable X_k into a z. In practice, the most partial consideration is given to all observations in the dataset; only the value of X_k is replaced.

The equivalent of PDP for single observations is the Ceteris Paribus profile, showing the dependence of an instance X^* prediction as

$$g_k^{CP}(X^*, z) = \mathcal{P}\left(\mathcal{A}, (X^*_{k|=z})\right), \tag{2.8}$$

where $X^*_{k|=z}$ stands for a vector that has all variables set to the values of observation X^*, but the value of the variable X_k is modified.

CP can be seen as a decomposition of PDP profiles since aggregating and averaging CP profiles over observations yields just the PDP profile. In the case of additive models, the profiles for different observations are parallel to each other. If, on the other hand, they intersect, then there may be interactions in the model.

2.4.2 Understanding domain

In order to incorporate domain knowledge into AutoDS systems, it is first necessary to formalize this external information in an appropriate way. This is achieved through the use of an ontology.

Ontologies

Ontologies are formalized systems that describe a set of concepts and their relationships within a specific domain, providing a structured framework for organizing and integrating knowledge. In scientific and technical contexts, ontologies play a crucial role in enabling consistent data annotation, semantic interoperability, and automated reasoning. Here, we delve into the formal definition, components, and applications of ontologies.

Definition and Formalism

An ontology $\mathcal{O}$ is typically defined as a tuple $\mathcal{O} = (C, R)$ where C defines a set of concepts that represent a particular domain of knowledge, along with the relationships R between those concepts [80]. In order for the ontology to be properly defined, the classes and relations must take into account the axioms representing the rules and constraints in the domain. Algorithmically, ontology can be considered a Directed Acyclic Graph [209] when relations create a hierarchy or partial ordering of concepts, namely, each concept can have one or more parent and child relationships, but any concept is an ancestor of itself.

Ontologies have a wide range of applications across various scientific and technical fields, serving as essential tools for organizing and utilizing complex information. In biomedical informatics, ontologies such as the Gene Ontology (GO,[8]) and the Foundational Model of Anatomy (FMA,[180]) provide structured vocabularies that facilitate the annotation and integration of biological data, thus enhancing discoveries and promoting interoperability between databases. For example, GO defines classes such as *Biological Process*, *Cellular Component*, and *Molecular Function*, along with their interrelationships

and axioms, which enable queries and inferences about gene functions. In the context of the Semantic Web, ontologies are crucial for enabling machines to understand and process domain content in a meaningful way. The Web Ontology Language (OWL,[78]) is frequently employed to create and share ontologies in this context.

Through the structure of ontologies, semantic relationships and similarities between different concepts can be evaluated. There are many methods for evaluating semantic similarity using graph structure of connections. The most basic measure is the length of the shortest path between terms [172]. This simple method does not always correspond to the properties we want to represent, so other definitions have been developed using the specificity of a term. Examples of these are similarity measures developed in the biomedical field by Resnik [177], Lin et al. [133], and Jiang and Conrath [106]. Another approach to finding semantic similarity is to create embeddings of ontology terms and then evaluate the distance between their representations. The OWL2Vec [35] is an example of an ontology embedding approach.

Despite so many successful applications, from the AutoDS point of view, ontologies are a new tool and their potential has not yet been explored.

Part I

Evaluation of the individual stages of the AutoDS

In the first part of the thesis, we discuss the aspect of evaluation of the various elements of AutoDS. In order to increase confidence in the frameworks, we want to extend the perspective beyond just numerical evaluation and provide users with an in-depth analysis of why particular operations are applied.
In this part, we focus on three aspects of Automated Data Science.

Chapter 3 discusses the evaluation of imputation methods, which has to be carried out during the Data Engineering step. This aspect of data preprocessing has not been validated early on, and missing data is a relevant problem due to its prevalence and the fact that most machine learning algorithms do not handle missing data.

Chapter 4 is an extended evaluation of the meta-features used in meta-learning. This work fits into the Model Building field, as meta-learning is part of model optimization. This evaluation is based on a multi-faceted analysis of the meta-model with the application of explainable machine learning techniques.

Chapter 5 is to extend the evaluation of machine learning benchmarks to provide more information than just a comparison of performance measures. The EPP meta-score provides an additional probabilistic interpretation and a function aggregating the performance model between multiple repetitions of the experiment.

Chapter 3

Does imputation matter? Evaluation of preprocessing method

This chapter corresponds to the article Katarzyna Woźnica and Przemysław Biecek. Does imputation matter? Benchmark for predictive models. (arXiv:2007.02837), 2020. This article was presented at the *Workshop on the Art of Learning with Missing Values (Artemiss) at International Conference of Machine Learning.*

AutoDS background in relation to Hypothesis 1

Each preprocessing operation results in a data transformation and can affect the quality of the machine learning model. Given the number of potential preprocessing steps and the variety of techniques associated with each, including all of them in the search space of potential algorithms significantly increases the dimension of that space. A way to address this problem is to independently benchmark preprocessing techniques on how they may affect model performance. Conducting a benchmark that identifies preprocessing techniques can exclude those that have little impact on the performance model, and more resources are gained for other steps.

In addition, it is crucial to benchmark preprocessing methods with a view to applying them to AutoML and AutoDS. Often, AutoDS imposes a potential preprocessing technique to meet specific assumptions that do not necessarily need to be met in method development. For example, developers of preprocessing methods may have other priorities and pay attention to other aspects of the performance of the chosen

techniques, e.g., the stability of the selection of the same variables in feature selection or the accuracy of the values of the imputed data.
In the following chapter, we benchmark imputation methods as preprocessing techniques impacting the model performance.

3.1 Introduction

In practical tasks in data analysis and machine learning, one of the most common problems are missing values in collected data. On the other hand, many established machine learning algorithms require fully observed datasets without any missing entries. Due to this, imputation is a necessary step in preprocessing, and the subject of handling missing data is a challenge for practitioners. We can observe this, for example, on the Kaggle platform, where users compete in real-life machine learning tasks. Competitors often share their knowledge among other approaches to imputation data. This gives us insight into the trend of applied methods: there is a tendency to apply simple methods such as mean or mode replacement regardless of their limitations.

At the same time, many statisticians work on more complex methods with theoretical foundations. Rubin [183] was the first to formalize the universal three categories of the process generating: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). Since then numerous techniques of substituting missing data were developed. In R package wide range of single imputation are implemented: missForest [201], softImpute [87], VIM [120], missMDA [114] . A variety of multiple imputation techniques are also available: mice [214], Amelia [93], missMDA [114]. Most of these implementations deal with imputing missing entries in continuous and categorical variables. In addition, most of these packages enable more than one method of imputation. In the platform R-miss-tastic [142] can be found a comprehensive summary of existing techniques.

Due to the plenitude of available packages and methods, global evaluation of existing techniques is desirable. So far only a few articles address this necessity [125, 103]. These comparisons focused on the quality of imputed data. They considered simulated data and applied several imputation methods was assessed in terms of the accuracy of predicting the missing values.

In most cases, handling missing data is a prepossessing step to complete data before the primary modelling task. For practitioners, a crucial aspect in choosing an imputation method is the impact of the selected procedure on the predictive power of the ML model. Recently, in machine learning, more attention has been paid to benchmarking and comparison of various predictive algorithms or the importance of hyperparameters, but only a few papers have taken into account the selection of prepossessing techniques. Brown and Kros [28] provide a descriptive study of imputation impact on machine learning algorithms but did not support these conclusions with any empirical results. Hutter et al. [97] considered substitute missing values with mean, median, or mode as one of the hyperparameters in importance analysis, but imputations were limited to simple ad-hoc approaches, and their impact was inappreciable.

Contributions. In this chapter, we research the contributions of imputation methods to the improvement of predicting the power of machine learning classification algorithms. We provide a benchmark on 13 real-life tasks and face simple methods with more sophisticated ones. The proposed benchmark has a universal nature and can be applied to assess the influence of any imputation methods on a wide range of datasets and algorithms.

3.2 Experiments settings

A reliable source of real-world datasets is OpenML, especially the collection of classification task OpenML100 [19]. In this experiment, we focus on binary classification and pick datasets with at least one column with missing values. We select from the OpenML database thirteen datasets meeting these criteria. In Table 3.1, we present a summary of basic information about considered tasks. Every dataset was split into two parts: a training sample consisting of 80% of instances and a test sample. ML models were trained on the training part and then metrics were validated on test data.

3.2.1 Imputation methods

We select seven methods of handling missing data. For some data, some methods did not work. Next to the imputation name, in brackets, we give a number of datasets on which particular methods succeed in imputing. We test two simple ad-hoc methods: **random** (13) - every missing entry was replaced with a value drawn independently from

Table 3.1: Statistics of considered OpenML datasets: number of instances, percentage of missing values, number of continuous variables, number of continuous variables with missing values, number of categorical variables, number of categorical variables with missing values.

dataset name (dataset ID)	# obs	prc of missings	# num.	# num. w. missings	# cat.	# cat. w. missings
ipums_la_99-small (1018)	8844	7%	15	0	41	14
adult (1590)	48842	1%	4	0	9	3
eucalyptus (188)	736	3.9%	14	10	2	0
dresses-sales (23381)	500	14.7%	1	1	12	9
colic (27)	368	16.3%	5	5	15	13
credit-approval (29)	690	0.6%	6	2	10	5
sick (38)	3772	2.2%	6	6	22	1
labor (4)	57	33.6%	8	8	9	8
SpeedDating (40536)	8378	1.8%	59	58	64	3
hepatitis (55)	155	5.4%	6	5	14	10
vote (56)	435	5.3%	0	0	17	16
cylinder-bands (6332)	540	5.1%	19	18	15	7
echoMonths (944)	130	7.5%	6	6	4	1

observed values of the considered feature, **mean** (13) - filling missing values with mode for categorical variables and mean for continuous variables of complete values in a feature. Moreover, we consider four single imputation methods. The first is **softImpute** (10) - for numeric variables fit a low-rank matrix approximation to a matrix with missing values. For categorical variables, missing values are imputed with mode. The second is **missForest** (11) - imputation with predictions of random forest model, trained on complete observations. Available for both numeric and categorical variables, From VIM package, we choose two methods: **VIM kknn** (13) - k-Nearest Neighbour imputation can be applied to numeric and categorical features, and **VIM hotdeck** (13) - sequential, random hot-deck algorithm. As representative of multiple imputation, we include **mice** (10) - we test default methods from this package: predictive mean matching for numerical variables and polytomous logistic regression for categorical ones.

Some of the above-mentioned methods depend on additional parameters. In this benchmark, we used default values. Reproducible scripts are available at `https://github.com/ModelOriented/EMMA`. We also tried Amelia and missMDA, but they failed to impute most datasets, so they were excluded from the benchmark. To prevent data leakage and simulate a real-world application, we should fit imputation methods on the train dataset and then apply this to the test sample. Unfortunately, in used packages

implementations, this is impossible, so we decided to impute data separately on train and test data. For the same reasons, the target variable was excluded from the imputation step.

3.2.2 ML Algorithms

We select five types of algorithms that should capture the different structures of data:

logistic regression with regularization (implemented in glmnet package), classification tree (rpart), random forest (implemented of ranger package), k-nearest neighbors, and XGBoost. In this benchmark we leave aside hyperparameter tuning, every algorithm was trained with default settings.

In the first step, for every datasets on train and test part, missing values are substituted with seven imputation methods. Then, on train data, five types of algorithms were fit, and on test data, we reported the value of two performance measures obtained on test data: Area Under Curve (AUC) and F1. We consider two types of measures because of that some datasets have imbalanced response variable, and F1 captures this aspect of quality of performance.

3.3 Results

Because selected measures are incomparable across datasets, next to the comparison of values of metrics we create the ranking. For every task and every machine learning algorithms we rank imputation methods, scores can range from 1 to 7. The higher and better measure the lower rank obtain this imputation technique. Methods that did not work on specific tasks get the lowest score (7). For some datasets and algorithms, despite different imputation methods, some models achieve exactly the same measure values. In case of ties we assign a maximum value of ranks, so rank 1 corresponds to evidently the best model.

In our analysis, we focus on three main questions about globally the best imputation method and the interaction between imputation and classifiers. We check trends in obtained results and attempt to draw conclusions about the optimal workflow for incomplete data.

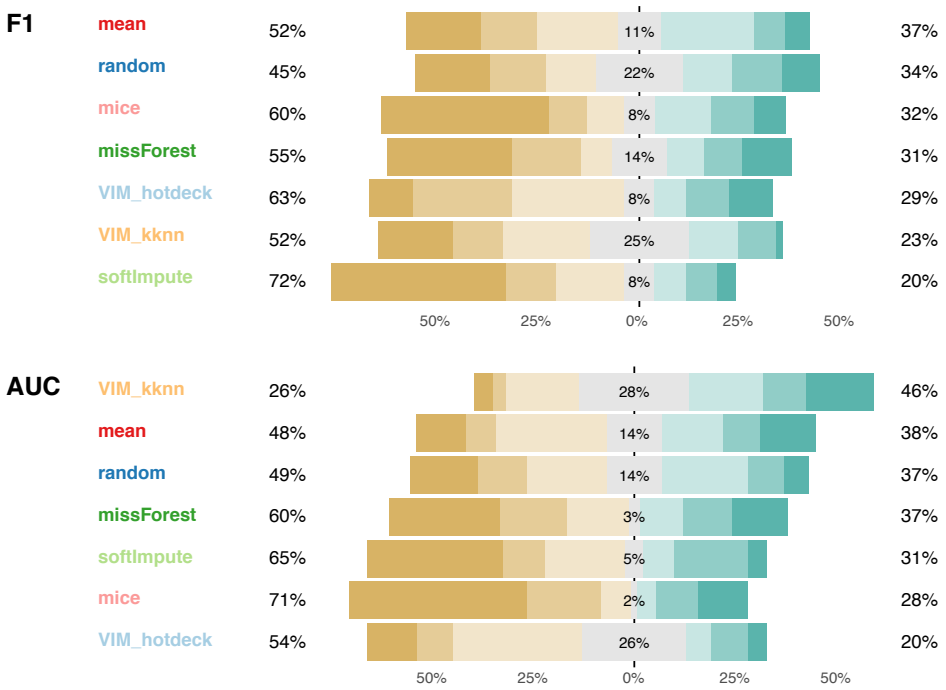

Figure 3.1: Bars describe how often a given method of imputation had the best results (rank 1, dark green) or the worst results (rank 7, dark orange) for a particular pair ML-model/dataset. The top ranking is based on the F1 measure, while the bottom one is based on AUC. The percentages on the right describe how often a method was in positions 1 to 3. The percentages on the left describe how often a method was in position 5 to 7.

3.3.1 Does exist the best universal imputation method?

In Figure 3.1 for every imputation method, we show the distribution of ranks based on F1 and AUC measure. A single score corresponds to one task and one algorithm, so for every method, there are 65 scores. We can interpret this figure in various ways. If we assume that the best methods are to achieve rank 1 to 3 most frequently, then *mean* substituting is the winner for F1 measure and *kknn* method for AUC measure. On the other hand, every imputation method gives the best measure of F1 and AUC for at least one task and algorithm.

For both measures, top positions are taken by simple methods as *random*, *mean*, or *kknn* from the VIM package. Against this background, arises the question of whether these methods work effectively on similar tasks and ML algorithms or rather oppositely. In Figure 3.2 we present the percentage of covered best results in rankings of F1 and AUC measure by single imputation methods and all pairs of them. As a single imputation, we would choose *random* and *mean* or *kknn* for F1 and AUC, respectively. At the same time, combinations of two methods work definitely better, and they are able to cover above 50%

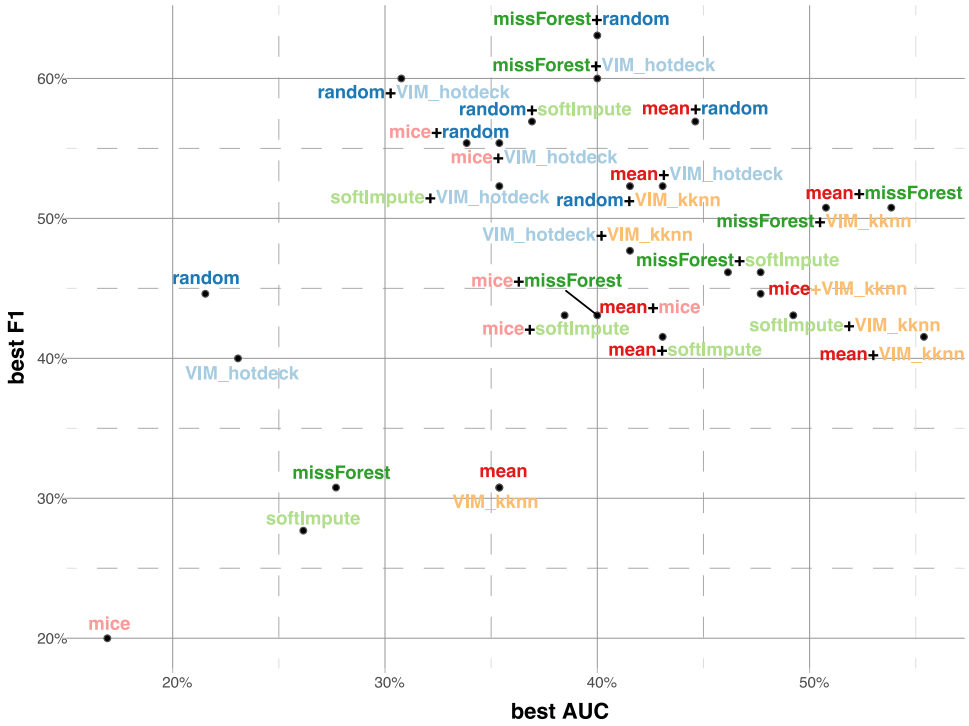

Figure 3.2: The OX axis shows how often the indicated imputation method has the best results measured by the AUC. The OY axis shows how often the indicated imputation method has the best results measured by F1. The points marked A+B refer to the better of the two indicated methods (parallel max).

of best results. For F1 measure *missForest* and *random* methods cover an outstanding percentage of best methods. Three of the most effective methods for F1 are *random*, *missForest* and *VIM_hotdeck*, but *missForest* imputation result in improvement of coverage of AUC best results. For the AUC measure, the optimal pair is *mean* and *VIM_kknn* substituting but *mean* and *missForest* are very close to them and, what is more, achieve better results for the F1 measure. We may notice that *missForest* imputation takes high positions for both measures.

To extend this approach we perform a greedy search to find the optimal sequence of imputation methods covering a wide range of tasks and algorithms. In Figure 3.3 on the left panel on the OY axis, we see these sequences for F1 and AUC, respectively. We see that for F1 *random*, *missForest* and *hotdeck* covers above 75% of optimal imputation methods for combinations of dataset and ML model. For AUC, the first two positions are the same as in Figure 3.2, but the third is *mice* imputation. It suggests that this imputation method works complementarily to simple approaches.

As we see, shown results may be concluded in different ways depending on the considered measure and there is no one answer to the question about universal the best

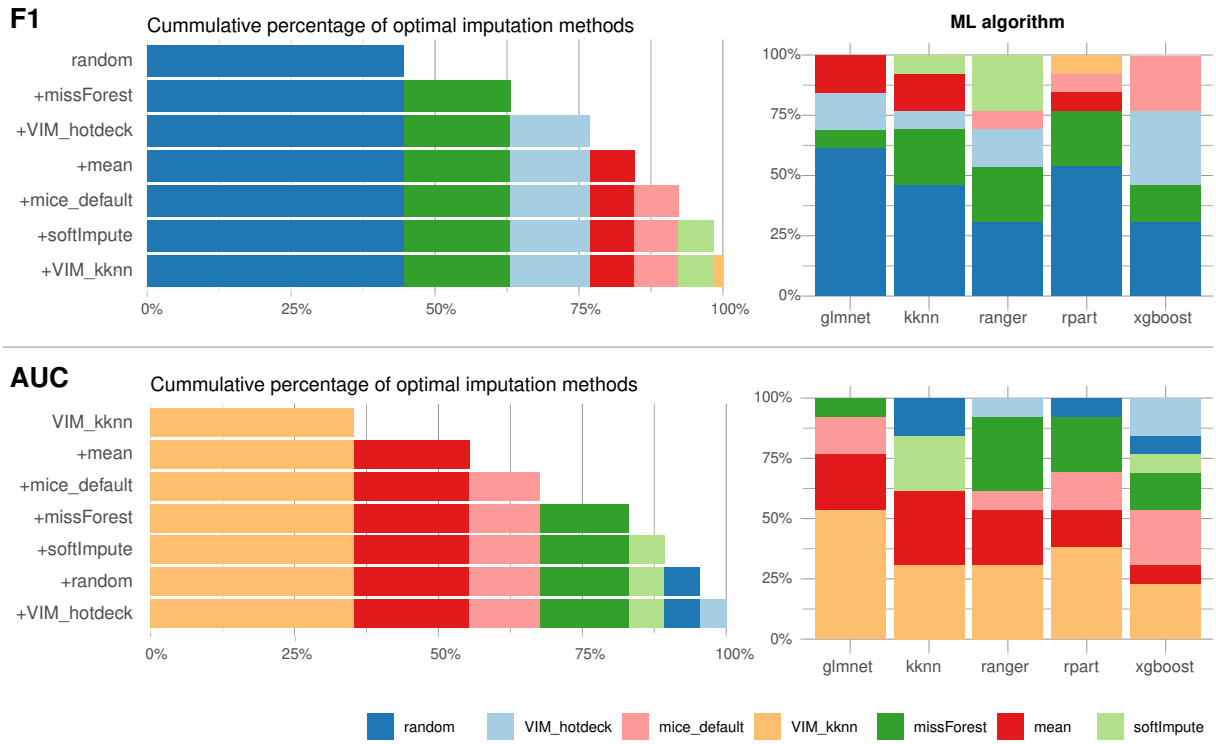

Figure 3.3: Left panel: On the OY axis, there are consecutive imputations from the greedy search for F1 and AUC, respectively. On the OX axis, there is a cumulative percentage of tasks and ML models for which one of the imputation methods was optimal. Right panel: Contribution of subsequent imputations from greedy search broken down by ML algorithms. On the OY axis there is a cumulative percentage of covered the best imputation.

imputation method. For considered tasks, it is difficult to select an imputation technique that maximizes both measures. The next question arising from this analysis is an interaction between imputation methods and the type of classifier algorithm.

3.3.2 Does exist the best imputation method for a machine learning algorithm?

In Table 3.2 we present averaged across datasets ranking based on F1 measures for imputation methods and classifier model. The lower score indicates better methods. According to averaged ranking, *mean* imputation is the best in 2 out of 5 models, for glmnet and kknn model. For ranger ML-method, *random* imputation wins, but *missForest* takes the top position in rpart model. For XGBoost, *hotdeck* achieves, on average, the best score. Deeper insight into the interaction of imputation and classifiers gives principal component analysis (PCA) performed on averaged rankings in Figure 3.4. The first PCA coordinate positively correlates with averaged ranking so *mean* method gives the best re-

Table 3.2: Average rank for a particular method of imputation and method of construction of the classifier

	glmnet	kknn	ranger	rpart	XGBoost
mean	4.38	3.85	4.77	4.69	4.62
mice_default	5.54	4.92	4.85	5.15	4.23
missForest	5.15	4.38	4.85	4.62	4.46
random	4.46	4.00	4.00	4.77	4.31
softImpute	5.69	4.69	4.92	5.92	5.46
VIM_hotdeck	4.85	4.62	4.15	4.85	3.92
VIM_kknn	5.15	4.23	5.08	4.15	4.69

sults. The second coordinate reveals model preferences. *Mean*, *missForest* and *VIM_kknn* methods cooperate with rpart and kknn while *mice* works with ranger and XGBoost. This conclusion goes along with Figure 3.3 on the upper right panel, where we present results for a greedy search of the optimal set of imputations splitting by ML models. We see that for the ranger, XGBoost, and rpart models, *mice* provide enhancement of substituting missing values in relation to *random* and *missForest* imputation.

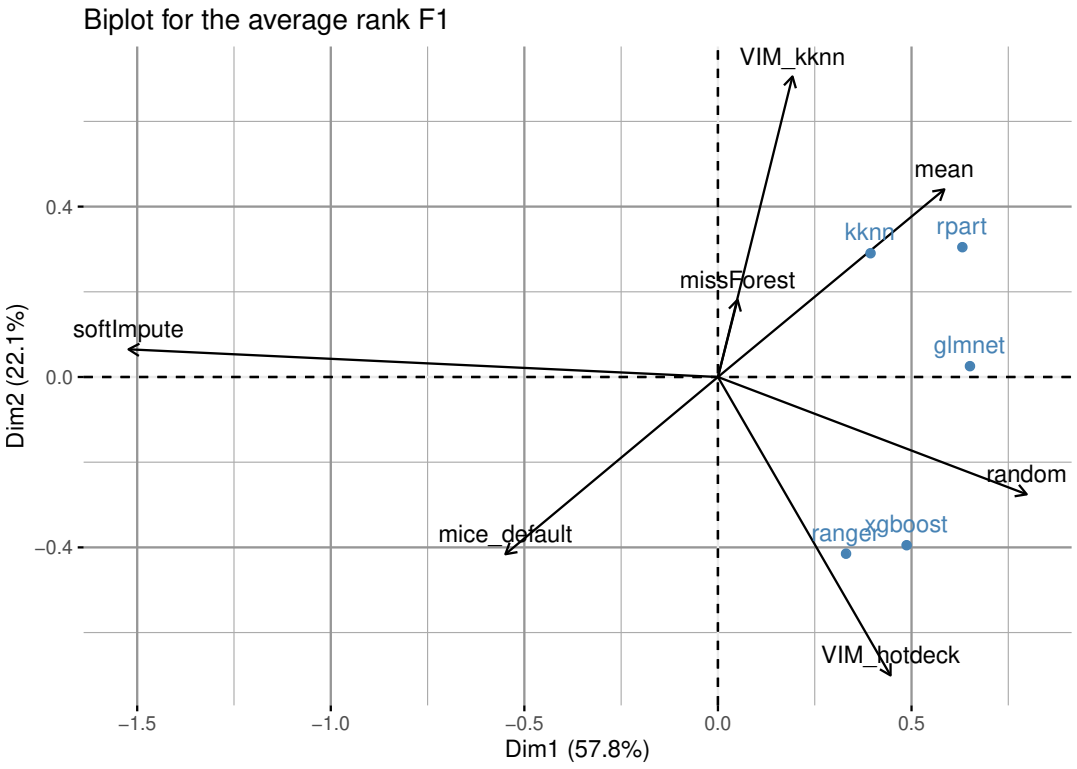

Figure 3.4: Biplot for a PCA made for an average of the rankings for pairs imputation-method / ML-method. The first coordinate correlates with the average ranking, the best results have the method mean. The second coordinate reveals the method's preferences. The mice method works better for the ranger model than for the kknn model. See Table 3.2 for details.

3.4 Conclusions

To our best knowledge, this is the first empirical benchmark of imputation methods in terms of their impact on the predictive power of classifier algorithms. This kind of verification of proposed methods enables a better understanding of the pros and cons of imputation techniques. We proposed a general plan for the experiment, which can be extended to different datasets, imputation methods, and predictive algorithms. In our experiment, simple imputation methods achieved surprisingly good results, but we can not definitely conclude that more advanced methods should be given up. We focus on the impact on their predictive power, but methods with statistical foundations achieve better results in accuracy in imputed values.

The included analysis of results does not provide a single universal default for the best imputation method, even for a particular ML model. What is more, the selection of these imputation methods is sensitive to the considered performance measure. We observe some trends in results, but generally, their structure is very complex. For humans, it is very difficult to summarize this in a concise way. This is the area to employ a meta-learning model to capture these high-level interactions.

Chapter 4

Towards explainable meta-learning

This chapter corresponds to the article Katarzyna Woźnica and Przemysław Biecek. Towards Explainable Meta- learning. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, volume 1524, pages 505–520. Springer International Publishing, Cham, 2021. doi: 10.1007/978-3-030-93736-2_38. This article was presented International Workshop and Tutorial on eXplainable Knowledge Discovery in Data Mining (XKDD Workshop) at the ECML PKDD Conference.

AutoDS background in relation to the Hypothesis 2

One of the key elements of the Model Building process is meta-learning, which involves transferring information between experiments that have already been conducted and the new experiment. Based on the similarity measure of the datasets, algorithm configurations are proposed that can be used for the new dataset. However, it remains an open question of which meta-features of the datasets we should consider. This is not defined so far, and it is necessary to assess which meta-features correlate most with the quality of the model.

At this point, it is possible to use the internal processes of AutoML systems as a source of knowledge about which meta-features are most relevant and why we represent datasets in a given way. Extending the evaluation of the meta-learning stage can contribute to a greater understanding of what aspects of datasets are correlated with model performance.

This chapter aims to propose a methodology for the evaluation process of meta-models and selected meta-features. This extension employs explanatory machine

learning techniques, thus allowing for a more comprehensive examination of the impact of individual meta-features and their correlations with expected model performance across diverse configurations of hyperparameters.

4.1 Introduction

The effectiveness of meta-learning is highly dependent on the type and quality of meta-features [17]. So, it is crucial to consider a broad range of candidates [219]. The primary sources of meta-features that feasibly predict model performance are datasets' characteristics. The conventional approach is to express the whole dataset as a vector of simple engineered statistics.

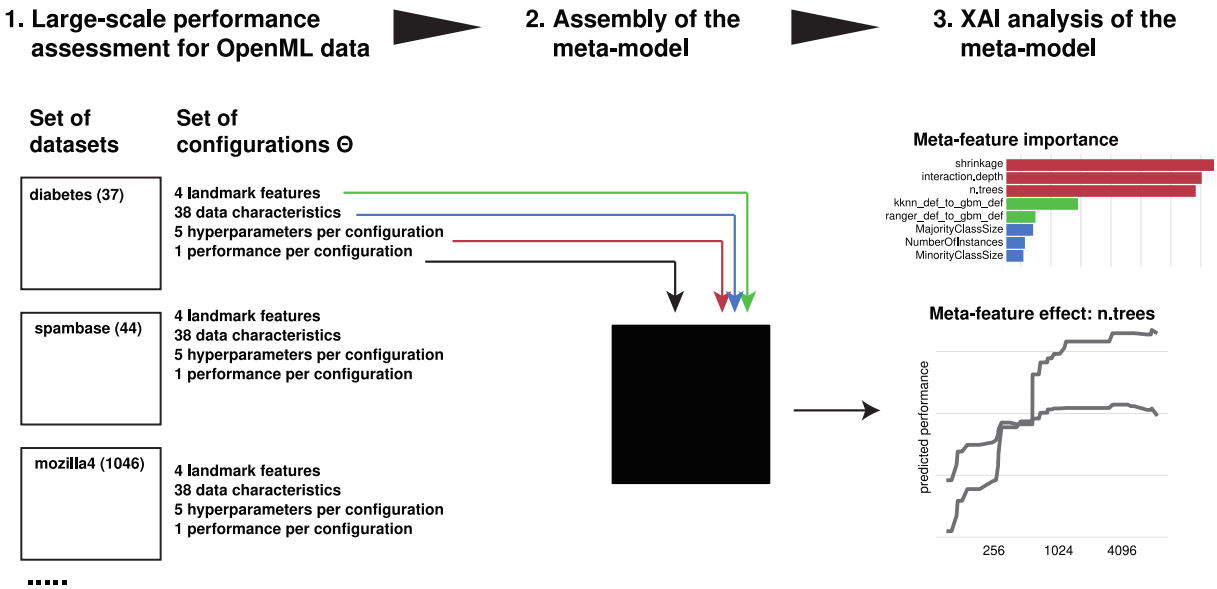

Figure 4.1: The proposed process of the explainable meta-model exploration. Firstly, we gather meta-features for selected datasets from the OpenML repository. Then we calculate model performance on these datasets for selected configurations of hyperparameters. Secondly, we assemble a black-box surrogate meta-model. Thirdly, we use XAI techniques to extract information about the relative importance of meta-features and their marginal responses.

A comprehensive summary of these commonly used meta-features is provided in [217, 179]. Autoencoders as meta-feature extractors [54, 90] avoid the problem of a priori defining meta-features, but they are limited to the same schema of meta-datasets. More versatility is offered by an extractor Dataset2Vec based on hierarchical modeling [111]. In addition to defining features through the internal data structure, we can also consider how difficult a given prediction problem is for a specific portfolio of machine

learning models. This is a concept of landmarking models introduced in [164]. It extends the list of meta-features by the relative performance of some predefined models, so-called landmarkers. As well as a dataset's properties, good meta-models take into account the tunability of particular algorithms concerning selected hyperparameters [170]. For that reason, a part of the meta-feature space is the detailed model configuration – user-determined hyperparameters.

Another subdomain of machine learning that has been developing intensively recently is eXplainable Artificial Intelligence (XAI). With the growing demand for highly engineered models, there is an apparent necessity to investigate black-box algorithms and identify the most critical aspects affecting model operation. There is a reservoir of XAI methods that address the justifications of a model's predictions for a single instance as well as dataset-level exploration [16, 149]. So far, these exploratory techniques have been applied to the enhancement of the ML model, but this kind of inference amplifies the meta-learning generalization ability. The transparency of the algorithm structure helps to identify which explanatory variables are included in the predictive process and how changes in their distribution affect a model. What is more, the effectiveness of meta-models greatly depends on the chosen set of meta-targets and on the approach to modeling the relationship between them and the meta-response. So, similarly to classic machine learning approaches, it can be a trial-and-error process that requires several iterations.

This work integrates these two promising directions (see Figure 4.1). We show how to use XAI techniques to extract knowledge and validate meta-models used in meta-learning. Focus is placed on the specific architecture of meta-models which enables the application of exploratory techniques. Using them, we extract essential informative properties of meta-features. We present an example for meta-learning based on OpenML100 datasets, but the proposed approach can be applied to any meta-model trained for a particular domain set of problems.

4.2 The Meta-OpenML100 surrogate model

In this section, we present a universal approach to exploration of the black-box meta-model explained in Section 2.1.4. Both the choice of dataset for which model evaluation and hyperparameters transfer was carried out and the choice of algorithm for the meta-learner should be considered as illustrative. Similarly to previous researches, this meta-

model works on classification problems from the OpenML100 benchmark [19]. We use a meta-model engineering methodology analogous to [176, 46, 170] and employ gradient boosting algorithm as meta-learner. In this section, we describe how this meta-model is built (Figure 4.1).

We build the meta-model based on all predictive tasks for binary classification in the OpenML100 suite; see the list of these tasks in Table 4.1. According to [219] recommendations, for each task, we calculate the following meta-features: four landmarkers' performance for baseline models and 38 statistical properties of the underlying dataset. The meta-model is built to predict the performance of the gradient boosting model parametrized with five hyperparameters. Finally, we train a meta-learner, also a gradient boosting model, to predict the performance of a model with selected hyperparameters on a dataset with the following meta-properties.

This black-box surrogate model is hereafter called Meta-OpenML100 in this article. In the following subsections, we provide a detailed description of its components.

4.3 Predictive tasks and their meta-features

Out of all tasks in the OpenML100 suite [19] we select only these for binary classification. The suite provides 61 meta-characteristics for these tasks [17], some correspond to properties of continuous variables, some for categorical variables, and some for datasets with mixed variables. Here the limit is to use only datasets with all continuous variables, i.e., 20 datasets listed in Table 4.1. For these datasets, we use 38 available statistical and information-theoretic properties.

4.4 Landmarkers

We use landmarker-based meta-features to characterize predictive problems. As landmarker models, we consider five machine learning algorithms with default hyperparameter configurations:*generalized linear regression with regularization*, *gradient boosting*, *k nearest neighbours*, two random forest implementations: *randomForest and ranger*. Various models have been proposed as landmarkers [9], but one of the requirements is diversified architectures of algorithm capturing interrelationship of different variables. We balance this diversity with the landmarkers' computational complexity [164].

Table 4.1: Meta-features for selected datasets from the OpenML repository. Four landmarkers (relative performance to default gbm model) and five hyperparameters for gbm model are presented. datasets' characteristics are omitted for brevity. Only optimal hyperparameters are listed in the last columns for the corresponding dataset.

	Hyperparameters for GBM model					Landmarks			
dataset (id)	shrink.	inter. depth	n.trees	bag fract.	min. node	knn	glmnet	ranger	random forest
diabetes (37)	0.00	4	1480	0.69	7	1.10	2.25	2.36	2.30
spambase (44)	0.04	5	1414	0.98	16	2.97	4.78	7.57	7.74
ada_agnostic (1043)	0.05	5	333	0.90	7	2.31	5.41	6.28	5.37
mozilla4 (1046)	0.09	4	1567	0.54	7	1.71	0.39	2.62	2.84
pc4 (1049)	0.01	5	2367	0.75	12	1.11	2.13	3.46	3.57
pc3 (1050)	0.04	1	949	0.90	11	0.86	1.34	1.85	1.88
kc2 (1063)	0.00	2	273	1.00	21	0.81	0.73	0.90	0.98
kc (1067)	0.00	3	5630	0.26	3	0.23	1.16	1.49	1.66
pc1 (1068)	0.00	4	6058	0.21	14	1.12	0.27	1.92	1.84
banknote authentication (1462)	0.03	1	8429	0.52	12	6.47	4.99	5.56	6.02
blood transfusion service center (1464)	0.01	1	394	0.21	10	0.64	1.23	0.86	0.71
climate model simulation crashes (1467)	0.00	2	654	0.26	15	0.35	1.36	1.12	1.15
eeg-eye-state (1471)	0.08	5	2604	0.28	14	2.48	0.93	3.34	4.16
hill-valley (1479)	0.08	5	2604	0.28	14	1.78	0.43	2.04	2.24
madelon (1485)	0.05	5	333	0.90	7	0.34	0.48	1.61	1.58
ozone-level-8hr (1487)	0.00	3	8868	0.33	11	1.41	3.40	4.43	4.36
phoneme (1489)	0.02	5	5107	0.60	18	4.89	2.90	7.12	8.13
qsar-biodeg (1494)	0.05	5	333	0.90	7	4.16	5.34	6.74	6.81
wdbc (1510)	0.04	1	949	0.90	11	1.56	0.63	1.88	1.91
wilt (1570)	0.00	4	6058	0.21	14	2.73	4.80	7.15	7.28

We apply 20 train/test split methods and compute AUC scores for each split to evaluate their predictive power. These five algorithms are ranked according to methodology in section 4.6. Because we predict the performance of gradient boosting models we compute landmarkers as a ratio of models rankings (knn, glmnet, ranger and randomForest) to a ranking of gbm model with the default configuration. As a result, there are four landmarker meta-features.

4.5 Algorithms and hyperparameters space

In this paper we explore the performance of gradient boosting classifiers (*gbm*) to the selection of the following hyperparameters: *n.trees*, *interaction.depth*, *n.minobsinnode*, *shrinkage*, *bag.fraction*. In order to do so, we sample 100 random configurations of hyperparameters in a similar way to [170] to examine their influence gradient boosting model predictive power. Additionally, we add one special configuration – the default settings for the `gbm` library [79].

4.6 Estimated predictive power of selected configurations

For every combination of 20 datasets from the OpenML100 suite and the 101 hyperparameter configurations, we try pre-specified 20 train and test data splits. Each model is fitted on each training subset, and afterward, AUC is computed on the test frame. This way, we obtain a meta-dataset for the performance of 40400 configuration/datasets/split combinations. Because performance for different datasets takes values in different ranges, we normalized these values using ranks per dataset (the higher AUC, the higher position in the ranking). Ratings are scaled to [0,1] intervals. Every configuration for each algorithm appears in the list 20 times because of train-test splits. To aggregate this to one value for every model, we computed the average rating for the model.

4.6.1 Surrogate meta-model

As meta-model, we select a gradient boosting algorithm with maximum interaction depth equal to 10 (i.e., a model that may be rich in interactions between meta-features). This kind of algorithm has already been applied inside sequential-model-based optimization, for example, in SMAC [96]. Tree-based algorithms are particularly well suited to handling high-dimensional and partially categorical input spaces. They are known for robustness and automated feature selection.

This surrogate meta-model configuration is selected as the best one according to the evaluation schema as follows. We apply one-dataset-out cross-validation: every model is trained on 19 datasets and then is tested on the remaining data frame for mean square

error (MSE), in addition Spearman's correlation between predicted rankings and actual meta-responses was also checked. Given meta-model achieves 0.017 MSE when the constant mean prediction has 0.041 MSE, so the performance of the surrogate meta-model is significantly better than the baseline mean prediction.

The meta-dataset with all meta-features and fully reproducible code can be found in this GitHub repository `https://github.com/woznicak/MetaFeaturesImpact`.

4.7 Explanatory analysis of Meta-OpenML100 model

the black-box meta-model approach, despite its effectiveness, does not provide new knowledge about how dataset characteristics translate into optimal choices for hyperparameters. In this chapter, we present the main result of this work – recommendations on how one can use selected XAI techniques [16, 149] for the analysis of the black-box meta-model. Each proposal is complemented with an example for the Meta-OpenML100 meta-model developed in the previous section. The knowledge drawn from the model often looks intuitive, but through the presented analysis, is it possible to get a quantitative validation of our assumptions.

The XAI methods allow the analysis of a single meta-model. Meta-OpenMl100 uses a one-dataset-out schema to explore the constructions of each of these meta-models independently or extend this approach and aggregate feature importance across cross-validation meta-models.

4.8 Meta-features importance

the meta-models are built on various sets of meta-features determined for datasets. For complex meta-models, it is difficult to discover which variables actually contribute to the model output. This investigation is needed to identify presumptive noisy aspects and may be significant in deliberation about excluding these meta-features from new generations meta-models. The solution to this problem is to use model agnostic permutational feature importance, which assesses how perturbations of a specific feature decrease model performance [64].

An example for Meta-OpenMl100 is presented in Figure 4.2. We can easily read that most important are hyperparameters followed by two landmarker features (`knn` and

`randomForest`) and two meta-features (`NumberOfInstances` and `MinorityClassSize`).

The considered meta-features form three groups because of the different approaches to creating them. Figure 4.2B presents the assessment of different groups' influence. As we see, the most important class of meta-features is hyperparameters, and this conclusion is consistent with the importance measure for individual variables. Landmarker and dataset characteristics have similar dropout values.

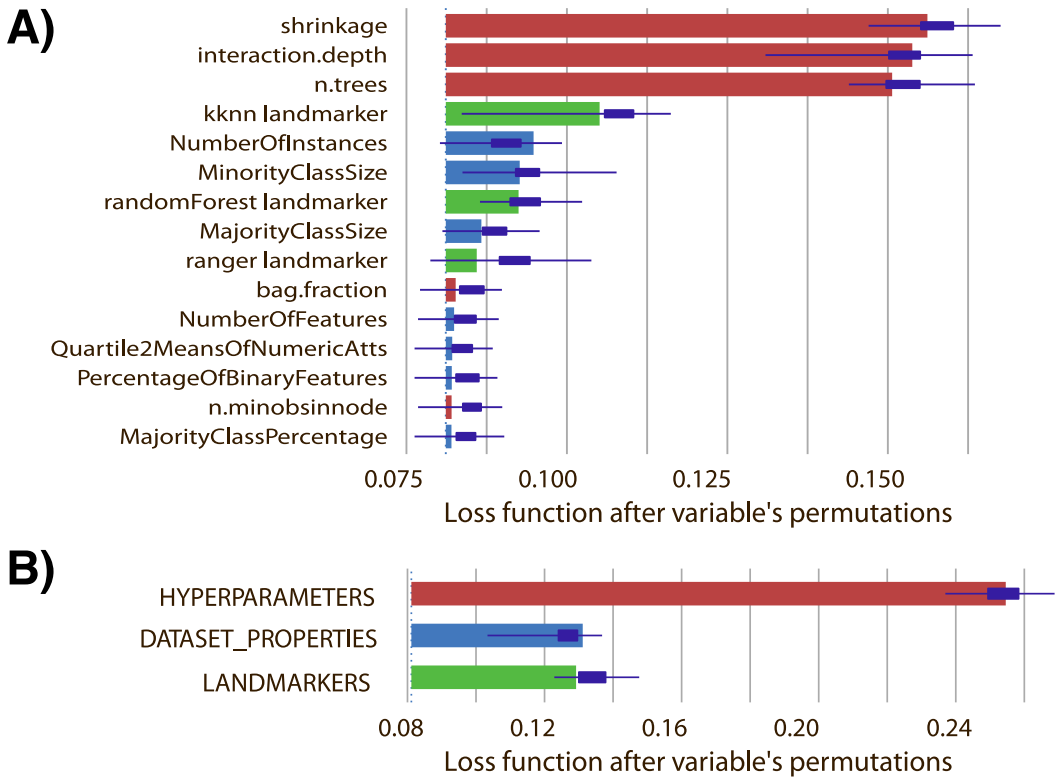

Figure 4.2: Panel A: Importance of the 15 top meta-features in GBM meta-model. Panel B: Cumulative importance of groups of meta-features.

4.8.1 Meta-features interactions

A good meta-model offers different optimal hyperparameters for different datasets to solve the problem of there being no one optimal hyperparameter across tasks. This means that the model detects interactions between variables, situations in which the values of one attribute (meta-characteristic) affect the effect of another attribute (hyperparameter). In general, the identification of interactions is a complex problem. For analysis of meta-models, we propose to use Friedman's H-statistic [67]. This method decomposes the prediction into components corresponding to two selected features. The variance of the difference between observed values and decomposed ones without interactions is used

to assess the strength of interactions. We use the implementation of Friedman's H-statistic from [150].

The example identification of interactions for Meta-OpenMl100 is presented in Figure 4.3. Firstly, we study two-way interaction between any two meta-features. The strongest interaction is for `bag.fraction` and `NumberOfFeatures`, i.e. between hyperparameter and statistical meta-feature. Based on such analysis, we can directly identify which meta-characteristics are related to the selection of particular hyperparameters, which is a competitive approach to [60].

In Figure 4.3B there is the overall assessment of the variable propensity to interact with any other meta-feature. This approach may be an alternative to functional ANOVA from [216]. In this case, the ranking is similar to this at the Figure 4.2. The most prone to interact variables are hyperparameters.

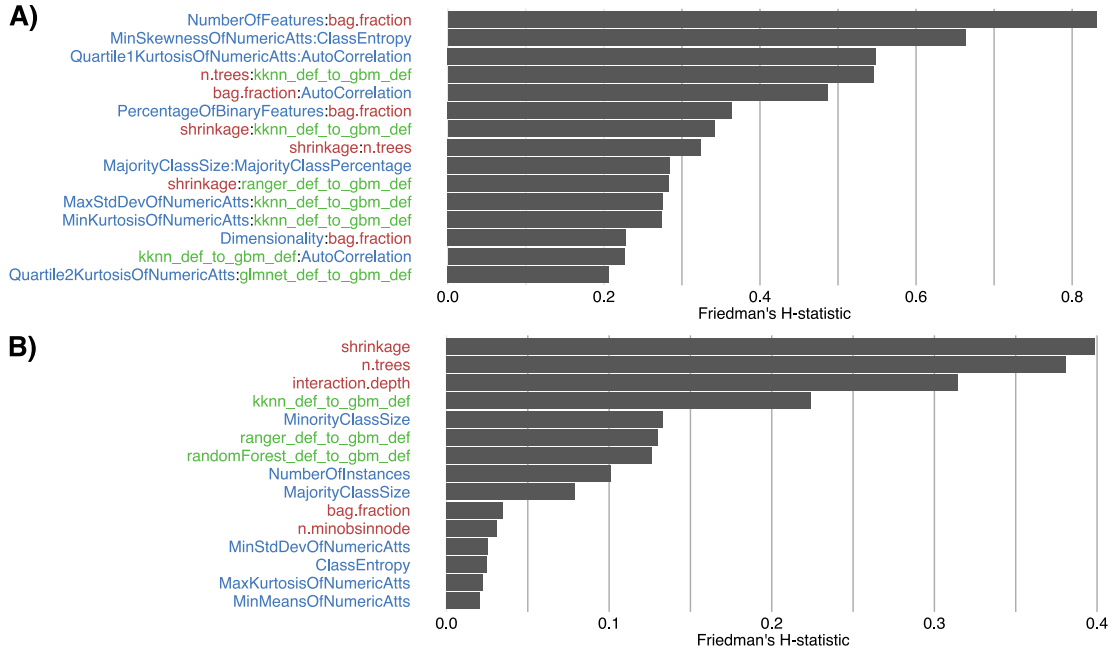

Figure 4.3: Panel A: Top 15 most important two-way interactions. Panel B: Top 15 most important meta-features in terms of overall propensity to interactions. Colors code groups of meta features

4.8.2 Importance of correlated meta-features

Meta-features are often correlated because they describe similar characteristics of datasets. It is not known whether adding more correlated features increases the quality of a meta-model anyway. To test the validity of a group of correlated variables, we propose to use the triplot technique [162]. Groups of meta-variables are determined according to hierar-

chical clustering based on correlations between meta-features. At each stage, the importance of the cluster of selected meta-variables is calculated.

In Figure 4.4, on the right panel, we see that most of the statistical meta-features have marginal contributions independently, and even after clustering, their accumulated importance is substantially less than individual hyperparameter contributions. Landmarkers connected with information about mean values and standard deviation of numeric columns are very close to the importance of hyperparameters independently.

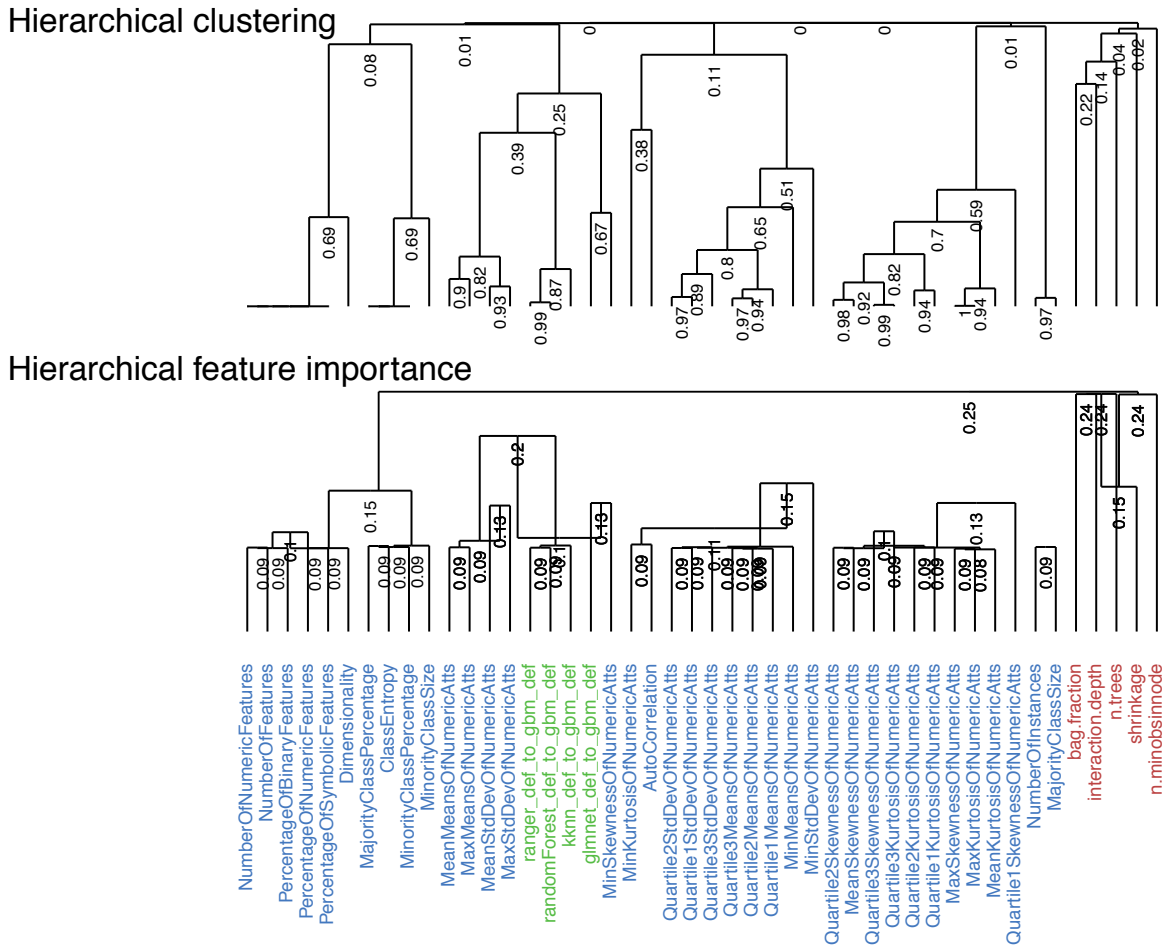

Figure 4.4: Triplot joint analysis of feature correlation and importance. The top panel shows hierarchical clustering for meta-features while the bottom one shows the importance of groups of features. Types of meta-features are color coded.

4.8.3 Hyperparameters informativeness

the majority of meta-learning is concentrated on obtaining the optimal hyperparameter configuration or effective warm-start-points for a novel task. Some of them present the estimated empirical prior distribution of hyperparameters [216]. Alternative approaches to

estimating the learning profile for a selected hyperparameter are Ceteris Paribus (CP) profiles, also known as Individual Conditional Expectation profiles [75]. This curve shows how a model's prediction would change if the value of a single exploratory variable changed. In essence, a CP profile shows the dependence of the conditional expectation of the dependent variable (response) on the values of the particular explanatory variable. This is equivalent to a partial dependence plot for an individual instance. In this analysis, we use its R implementation [15].

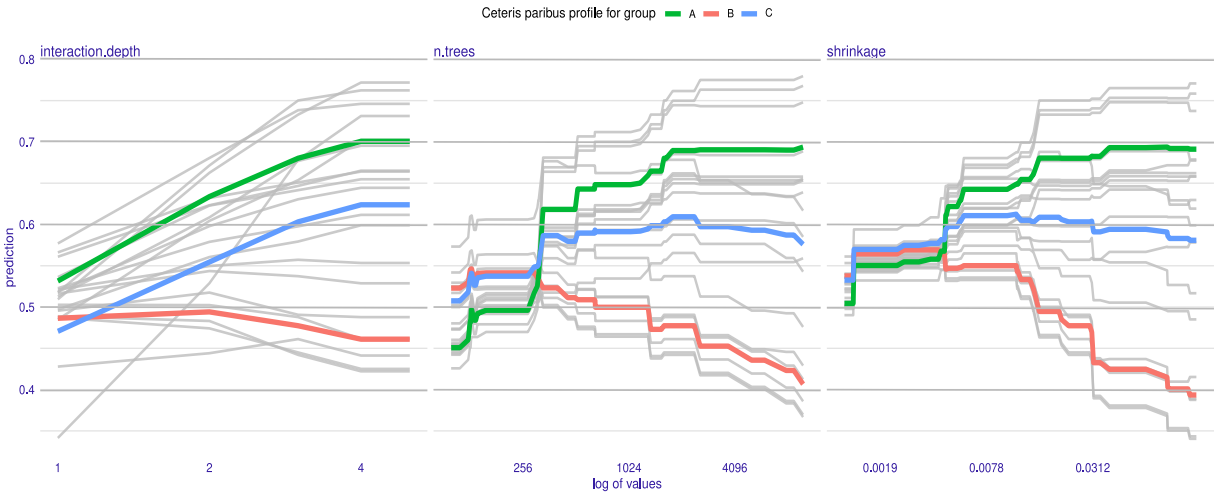

Figure 4.5: Ceteris paribus for hyperparameters for test instances. Thick colored lines are aggregated profiles for datasets clusters. Colors indicate groups. On the x-axis a logarithmic scale is applied.

As we argue, looking at the profile, we would like to point to the optimal hyperparameter value for independent data. Thus, we provide a CP profile for every dataset in meta-data. The curve corresponding to a particular dataset is extracted from meta-model from the cross-validation schema in which the selected task is the test example. As a result, we present CP profiles for every dataset when the prediction curve is independent of that data. This situation is equivalent to the production application of meta-framework optimization.

Obtained CP analysis for the selected hyperparameters is presented in Figure 4.5. To recall, the higher ranking, the better predictive power of the given gradient boosting model. So observing increasing lines in CP profiles indicates the better rating of the considered gbm models.

For `shrinkage`, `interaction.depth` and `n.trees` hyperparameters, three patterns of profiles are detected. We apply hierarchical clustering for profiles, and the aggregated profiles for three groups are shown in Figure 4.2. These groups are indicated with dis-

tinctive colors and termed A, B, and C. It is worth highlighting that the groups indicated A for two hyperparameters may consist of different datasets because the clusterings are performed independently.

For each of these three hyperparameters, group A has an increasing CP profile with a very strong trend for the interior values of variables. Profiles stabilize for high prediction for larger meta-feature values and the maximal value of hyperparameters would be pointed as an optimal warm start point to optimization. Similar behavior can be observed in group C. Group B is strictly different for `n.trees` and `shrinkage`: the highest prediction of rankings is obtained for the medium value of hyperparameters, and then predictions decrease. The CP profiles therefore confirm that there is no single value of hyperparameters that is optimal for all datasets, but at the same time suggest several hyperparameter values worth testing because they are optimal for a certain group of prediction problems.

4.8.4 Robustness of meta-data

For reliable validation, we select the *full* model as one specified meta-model from the Meta-OpenMl100 one-dataset-out schema: dataset 1471 is the test instance. For this data, we check the change in optimal hyperparameter values. The influences of the remaining datasets are estimated by sequentially deleting them as we describe above.

Figure 4.6A shows the scatter plot for the considered measures of disturbance caused by removing the effect of the single dataset. On the x-axis, there is a distance between the optimal shrinkage hyperparameter; on the y-axis, there is the value of Cook's distance. The biggest perturbations in the final predictions are caused by deleting the datasets 1485 and 37. For dataset 37 we also observe the most significant shift of optimal hyperparameter value.

In Figure 4.6B are CP profiles for the full meta-model and the selected limited meta-models with diverse values of Cook's distance. These profiles allow the assessment of the significance of the change in the choice of the optimal point. Only for dataset 37 we observe the transformation of the CP profile in comparison to the *full* model. This observation shows that the selection of meta-data in this example provides the robust selection of hyperparameter warm-start.

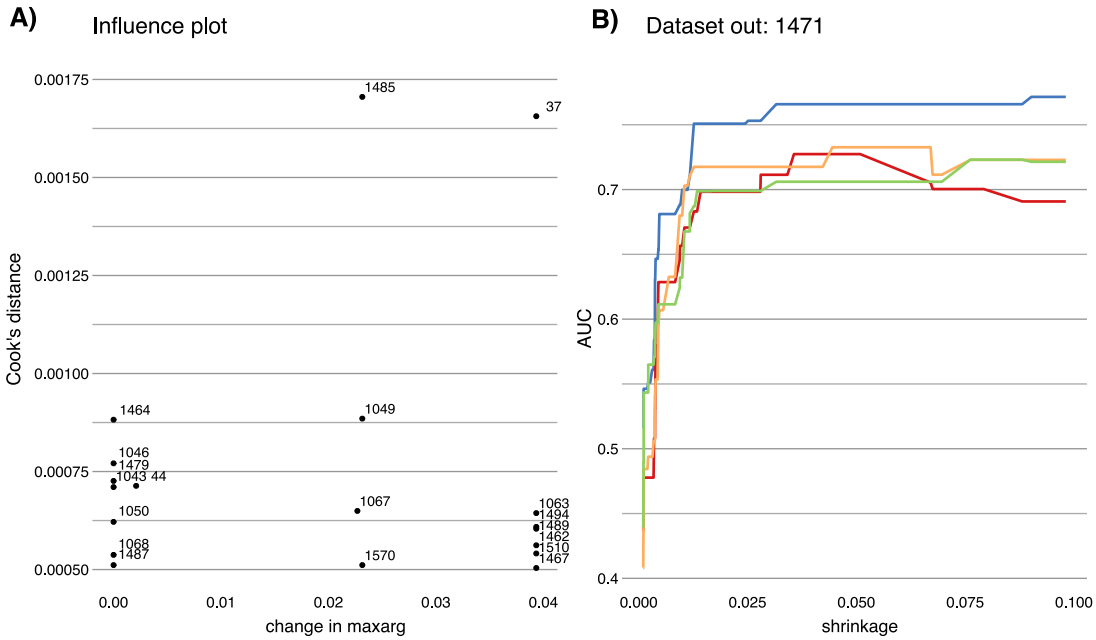

Figure 4.6: Panel A: Cook's distance vs. change in estimated optimal `shrinkage`. Numbers stand for the OpenML datasets ids. Panel B: Individual profiles for the most and the least influential dataset. Blue line corresponds to the *full* model.

4.9 Conclusions

Meta-learning is a promising approach for AutoML solutions. The main contribution of this is to show the applications of techniques known in the explainable artificial intelligence area in meta-learning. It turns out that many model exploration techniques not only increase knowledge of which meta-features are relevant, but also increase knowledge of the relationship of hyperparameters to model performance. In this work, we have shown how to use XAI techniques to build an effective meta-model, and extract knowledge about the importance of the particular meta-features in the meta-model. Variable importance technique and extension of this, taking into account the correlation structure of a meta-feature, amplify the selection of the most informative set of meta-features. This is crucial in the trial-and-error process of defining meta-space and brings us closer to find a transferable representation of datasets. Examining interaction between dataset based meta-features and Individual Conditional Expectation profiles for hyperparameters supports hyperparameter tuning. In further research, this may lead to a significant reduction of the dimension of the searched hyperparameter space and an improvement of automatic model selection processes. What is more, the choice of datasets determines the quality of meta-features and the evaluation of hyperparameters. Validating datasets'

informativeness through Cook's distance helps build a robust and reliable repository for meta-learning.

This approach is universal and generic to the explainable analysis of any meta-learning model presented in Figure 4.1. The OpenML may be argued that this is not the appropriate illustration for every meta-learning problem - the datasets are relatively small. However, at the moment, there are no publicly available repositories for big data problems. This approach can be reproduced on any repository of datasets from a specific domain or datasets of a selected size or complexity. It is sufficient to store a selected meta-targets, fixed hyperparameters, models performance, and then build a black box meta-model to predict model performance for novel data.

Chapter 5

Interpretable meta-score for model performance

This chapter corresponds to the article Alicja Gosiewska, Katarzyna Woźnica, and Przemysław Biecek. Interpretable meta-score for model performance. *Nature Machine Intelligence*, 4 (9):792–800. doi: 10.1038/s42256-022-00531-2

AutoDS background in relation to the Hypothesis 3

The comparison of the quality of machine learning models is essential for model selection and hyperparameter optimization. This process is typically conducted internally in the majority of AutoML frameworks. However, it also plays a significant role in the Exploitation of results. How model performance is reported is so important because it is based on these results that users decide whether to potentially implement the model or at least continue working with AutoDS.

In the final summary, users can assess the quality of different models by examining their values and rankings. This approach, however, is not sufficient for practical applications where users require a deeper understanding of the factors influencing the differences in model performance. However, it is often the case that we are dealing with non-technical audiences, and thus, it is necessary to communicate results in an easily accessible way. One example of such interpretation is the Elo-based Predictive Power (EPP) meta-measure, which provides a probabilistic interpretation of differences in model performance.

In this chapter, we present the application of the EPP measure in the benchmark of models built for OpenML datasets. Using the unique properties of EPP, in-depth analyses of this benchmark are presented.

5.1 Introduction

The current rapid development of machine learning area has resulted in a considerable increase in the number of new algorithms that need to be compared to the state-of-the-art ones. Along with this emerged the need to establish procedures for the systematic comparisons of algorithms. The current practice is to create benchmarks on predefined sets of tasks, for example GLUE [221], SuperGlue [222], or VTAB [235]. Benchmarks are created especially in deep learning because the aim here is to build unified algorithms with understanding beyond the shallow patterns in data, which is why it is crucial to compare algorithms on a wide variety of tasks from different domains. Another way to evaluate the progress in algorithms development are biological competitions, such as Critical Assessment of protein Structure Prediction (CASP, [123]), Critical Assessment of protein Function Annotation algorithms (CAFA, [239]), or Critical Assessment of PRediction of Interactions (CAPRI, [129]). Through this, the performance of algorithms for predicting new structures, properties, or interactions between proteins is regularly compared. Another popular approach for model comparison is storing and sharing the results of multiple algorithms on multiple datasets on platforms such as Kaggle[1], Papers With Code[2] or OpenML [19].

These days, it is difficult to imagine a high-quality article with a new algorithm without comparing it with the state-of-the-art methods on at least one of the benchmarks. Despite that, there is no unified description of benchmarks to refer to when describing new ones. What is more, performance measures of models currently used in benchmarks share many limitations, such as the lack of possibility to interpret differences in performance or the impossibility of comparing models between datasets.

Considering the shortcomings of existing benchmarks, the need for new approaches for comparing models and establishing new guidelines is being felt in the machine learning

[1] https://www.kaggle.com/
[2] https://paperswithcode.com/

community [141]. This urgency is borne out by the fact that in 2021, the organizers of the Thirty-fifth Conference on Neural Information Processing Systems provided a new track dedicated to datasets and benchmarks[3].

In this article, we propose an Elo-based Predictive Power (EPP) Meta-Score, which is a new way of aggregating model results and which overcomes the most common problems with benchmarks and the performance scores they use. The main contributions of this work are as follows.

- We propose a Unified Benchmark Ontology that allows for the uniform description of different benchmarks.

- We identify and demonstrate the limitations of the most common measures of machine learning model performance, such as the lack of interpretation of differences and incomparability between datasets.

- In light of the highlighted limitations of the most common measures, we propose a new Meta-Score named EPP that is built as an aggregation of other measures and enriches them by providing interpretable comparisons of models, even between datasets.

- We apply EPP on a large-scale benchmark from the OpenML repository and Visual Task Adaptation Benchmark. In both cases, we show how the use of EPP enriches the understanding of model performance.

5.1.1 Historical overview

The problem of model assessment is even older than modern statistics. Its origins can be traced to Laplace's work from 1796 on the nebular hypothesis. Since then, the increasing number of applications for models has led to an increase in the number of metrics describing their quality. The various measures of model performance differ in their properties and applications [169, 199]. The most common machine learning frameworks such as scikit-learn [161], TensorFlow [1], or mlr [18] rely on common measures such as accuracy, AUC, Recall, Precision, F1, cross-entropy for classification and MSE, RMSE, MAE for regression problems.

[3] https://nips.cc/Conferences/2021/CallForDatasetsBenchmarks

Beyond measure selection, there is an even more important problem related to evaluating whether the differences between its values are significant, or whether they come from noise in validation datasets. There have been many approaches to verify whether a new proposed algorithm improves the performance compared to previous state-of-the-art methods. The majority of them have been statistical testing procedures. Janez Demšar [48] reviewed commonly used practices and point out the vast number of problems with them. One of the earliest and most widely cited articles in this area is the one by Dietterich [50]. He gave a broad description of the taxonomy of the different kinds of statistical questions that arise in machine learning. He also introduced a new procedure for testing which of two classifiers is more accurate, called 5x2cv t-test and based on 5 iterations of 2-fold cross-validation. In each replication, two algorithms are trained on each fold and tested on the other fold. The test statistic is then a modified statistic of a paired t-test, where the standard deviation comes from the cross-validation. The 5x2cv test was later improved. Alpaydin [5] introduced a robust 5x2cv F test while Bouckaert [21] doubted the theoretical degrees of freedom and corrected them due to dependencies between experiments. However, the above methods do not fit into the actual trends in machine learning algorithms, where new algorithms are tested against multiple state-of-the-art models and over several datasets. For this purpose, more extensive methods began to be used. One of them is ANOVA [186], for example with Friedman's test [81, 168] for comparison of multiple models, however it can only give conclusions about whether there are differences in the performance of models. If there are differences, a post-hoc tests are needed to determine which model has better performance. One of the procedures used for post-hoc analysis is the Nemenyi test, that gives the statistical significance of performance differences for a pair of models. Results of Nemenyi test for many models and may be aggregated as Critical Difference plot that shows groups of models that are not different. However, these results are not transferable to new datasets (not used during testing) because tests results have no absolute meaning. Therefore, once we do a ranking, we cannot extend it without repeating the entire testing procedure. The first to use non-parametric tests for comparing models on multiple datasets was Hull [95]. Brazdil and Soares [25] used ranks to compare classification algorithms, yet they do not provide statistical tests.

Demšar [48] analyzed papers from five International Conferences on Machine Learning (1999-2003) that compared at least two classification models. The conference papers in-

cluded a wide range of approaches, from naive average accuracy over all datasets, through counting the number of times a model performed better than the others, to assessing statistical significance by pairwise t-tests. However, despite multiple hypothesis testing, only a few articles had Bonferroni correction, that is, a method to adjust tests' p-values in case of multiple comparisons.

The conclusion of the analysis was that there is no well-established procedure for comparing algorithms over multiple datasets. Furthermore, there are issues with common measures of model performance, such as uninterpretable differences between two values, or the inability to compare these values between datasets (see Section 5.3.3). Therefore, there is an emerging need to develop better solutions for models benchmarking. In this work, we introduce a method of model comparison that is based on the Elo ranking system used in sports, for example in chess and football.

5.1.2 Elo ranking system

The rating introduced by Elo [55] is a ranking system used for calculating the relative level of a player's skill. The difference between the Elo ratings of two players can be transferred into the probabilities of winning when they play against each other. Therefore, the difference in Elo scores is a predictor of the match result calculated on the basis of the history of players' matches. The scores of players are updated after each match they play, and a new rating is calculated on the basis of two components: the result of a match and the rating of the opponent. A player's level is not measured absolutely, although it is inferred from wins, losses, and draws against other players. After each match, the winner gains Elo points. The amount of received points is related to the strength of the opponent. If a player beats an opponent that has a higher Elo score, the victor would gain more points than if playing a weaker opponent. Conversely, a defeated player would lose more points if he lost a match against a player with a lower rating.

An Elo rating has many variations; one of them, popular in chess, is the algorithm of 400. It states that an average player has a rating of 1500, and reaching a rating over 2000 means that the player is one of the best. Let us consider two players, $Player_1$ and $Player_2$. $Player_1$ has an expected score of E_1, which is their probability of winning plus half of the probability of drawing with $Player_2$, and is expressed as:

$$E_1 = \frac{1}{1 + 10^{\frac{(S_1 - S2)}{400}}},$$

where S_1 and S_2 are ratings of $Player_1$ and $Player_2$. This formula shows an important property of Elo scores - the possibility to interpret them in terms of the probability of winning. For example, the difference of 200 rating points means that a more skilled player has a probability of winning $\frac{1}{1+10^{-\frac{200}{400}}} \approx 0.75$.

In addition to probabilistic interpretation, Elo rating has one more advantage. It is not necessary for every player to play against each other to provide a comparison of their skills. In the real world, it would be impossible to stage matches between all chess players, therefore Elo is used to find an approximation of true skill. Of course, the more matches played, the better the approximation; however, not all players need to play against each other.

5.2 Unified Benchmark Ontology

In this section, we introduce the Unified Benchmark Ontology for machine learning that fills the gap for a uniform description of benchmarks. Figure 5.1 contains a unified diagram for describing machine learning benchmarks. We use terms associated with sports tournaments, such as *Player*, *Tournament*, *Round*, *Leaderboard*. The detailed descriptions of all these components are in Table 5.1. In this article, whenever we refer to the components of a Unified Benchmark Ontology, we will indicate this with capitalization and italics. Each component may be assigned to a different machine learning element. The set of such assignments is a *Scheme*. Examples of *Schemes* are provided in the next subsection.

5.2.1 Example Schemes

In Table 5.2, we present example schemes, i.e., mappings between components of a Unified Ontology Benchmark and machine learning terms. *Scheme* Model/CV is one of the most standard benchmarking settings where models are compared on different cross-validation splits. *Scheme* Model/Task covers a situation when models are compared on several datasets where each is assigned to its own performance measure. The pair dataset and performance measure is a task. Examples of such benchmarks are SuperGlue and the Visual Task Adaptation Benchmark. The third example *Scheme* is dataset/Model. The aim

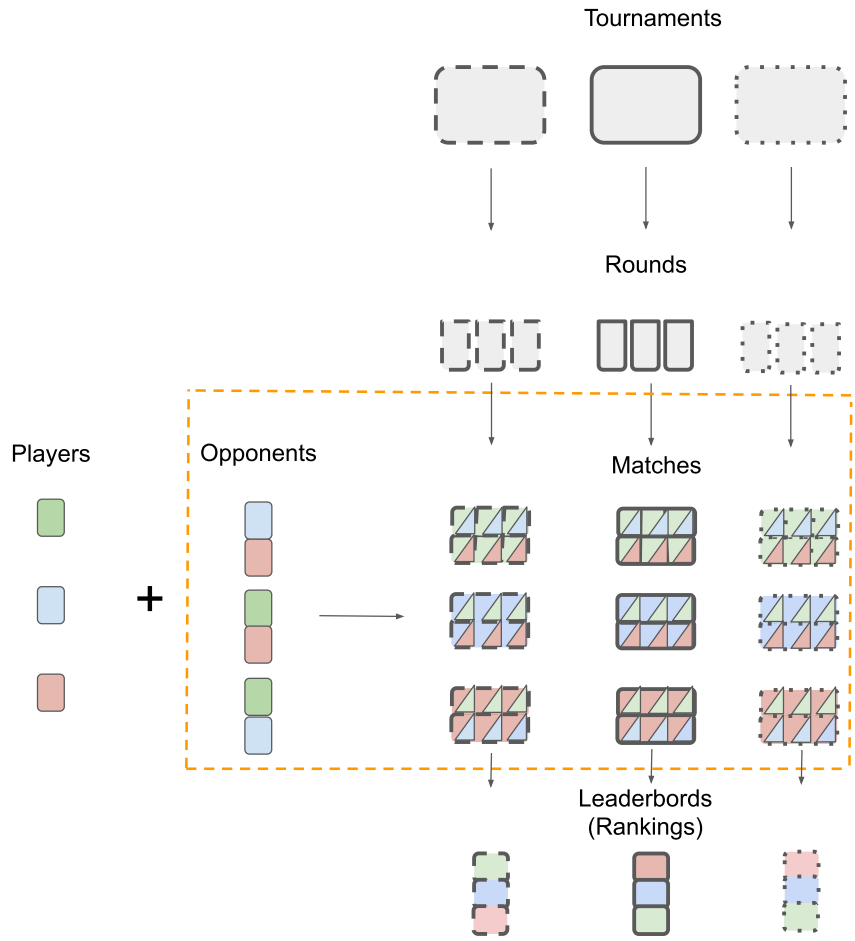

Figure 5.1: A diagram of the EPP Benchmark. The orange dashed lines shows EPP-specific parts of the benchmark and were not included in the unified benchmark.

Table 5.1: The descriptions of the EPP Benchmark components.

Component	Description	Example
Player $_i$	A single i-th participant of the EPP Benchmark.	Classification model
Score	A one-dimensional measure of a Player's strength. We assume that the order relation over Scores is given and monotonic.	Accuracy
Round $_r$	A single game environment for *Players*. The outcome of a *Round* $_r$ are score values of *Players*.	Cross-validation fold
Tournament	An independently replicated *Rounds*.	dataset
Meta-Score	A measure of a *Player*'s strength aggregated over all *Rounds* in a *Tournament*.	Mean
Leaderboard	The ordering of *Players* according to their overall strength on all *Rounds* in a *Tournament*.	Mean Accuracy of models over CV folds
Scheme	An assignment of EPP Benchmark components to specific machine learning pieces.	—

here is to compare datasets and assess how high performance models can score on it. This can be useful in assessing how simple it is to train a good model on a particular dataset.

Depending on the Scheme assumptions, the *Scores* for a particular *Player* may be

Table 5.2: Example *Schemes* for EPP Benchmark.

Component	**Scheme Model/CV**	**Scheme Model/Task**	**Scheme dataset/Model**
Player	Model	Model	dataset
Score	Performance measure, for example, AUC	Score defined separately for each dataset	Performance measure, for example, AUC
Round	Cross-Validation split	datasets with one train/test split each	Model
Tournament	dataset	One dataset	Set of models
Leaderboard	Separate rankings of models for each dataset	One ranking of all models	One ranking of all datasets

independent (when *Rounds* are different datasets) or correlated (when *Rounds* are cross-validation splits) across *Rounds*. In both cases, we assume that within *Round* the *Players*' *Scores* are comparable and can be ranked against each others's according to a given order relation.

5.3 Elo-based Predictive Power (EPP) Benchmark

In this section, we introduce the EPP Benchmark that fits into the nomenclature introduced in Section 5.2. In Section 5.3.1, we show key concepts of the Elo-based Predictive Power (EPP) score, while in Section 5.3.3, we show the common problems with state-of-the-art benchmarking methods and we derive the properties of the EPP score that overcome such issues.

Figure 5.1 presents the EPP Benchmark with the nomenclature from the Universal Benchmark Ontology. The thick arrow from Figure 5.1 is broken down here into additional components, such as *Opponents* and *Matches*. The detailed descriptions of components that are specific for EPP Benchmark are in Table 5.3.

Table 5.3: The descriptions of the EPP Benchmark components that extends the Universal Benchmark Ontology.

Component	**Description**
Opponent $_{i,j}$	*Player* $_j$ whose *Score* values are compared to the *Scores* values of the *Player* $_i$.
Match $_{i,j,r}$	A single comparison of the *Score* values of a pair of *Players*, i.e. *Player* $_i$ and *Opponent* $_{i,j}$ in *Round* $_r$.

5.3.1 The Concept of the Elo-based Predictive Power (EPP) Meta-Score

Elo-based Predictive Power (EPP) Meta-Score is used for establishing the *Players*' *Leaderboard* according to a single *Round* of the experiment. The *Players* are ranked according to their *Score* values in a single *Round*. However, the order of *Players* on the *Leaderboard* consistent with all *Rounds* may be impossible to determine. A single order might not have the property of connectivity with all *Rounds*, and therefore be nonlinear. That is why the common procedure is to aggregate, for example as an average, the *Scores* over *Rounds* and then obtain the *Leaderboard*. However, the mean is sensitive to outlier observations, thus models with strongly varying results will distort the aggregated ranking. In this work, we introduce an alternative approach that is EPP. The idea is not to aggregate values of *Scores*, but to compare the relative performances of *Players*. We ignore the absolute values of *Players*' *Scores* and the winner is the one whose *Score* is better (in terms of given order relation). Every *Round* $_r$ consists of *Match* $_{i,j,r}$ in which *Player* $_i$ competes with other *Player* $_j$ (*Opponent* $_{i,j}$). In consequence, for every pair of *Players* we get the sequence of win/lose results for every *Round* and can use these table for calculating the relative EPP Scores of *Players*' performances. This relativity of *Players*' performance makes EPP very similar to Elo, in particular in the way that both methods give a probabilistic interpretation of differences in score values.

However, the limitations of Elo ranking used in sports does not apply to the EPP for machine learning benchmarks. In classic Elo ranking, not every *Player* stands against every other. One hundred *Players* would have to play $\frac{100 \cdot 99}{2} = 4950$ *Matches*, which might be impossible for logistical or time reasons. Therefore, it is often hard to use all possible results of *Matches*. In the case of machine learning models, the cost of calculating EPP *Meta-Scores* is not as time consuming as human *Matches*. It is worth noting that a *Match* result is a comparison of *Players*' performances in one *Round* and the performance for a particular *Score* is the same, regardless of the *Opponent*. Therefore, for one hundred players we can obtain results of all *Matches* calculating only 100 values (performance of each *Player* in each *Round*). It is worth noting that it is not always possible to get *Score* value for each model in every *Round*, for example due to the missing data that only some of the models can deal with. However, EPP can still be calculated in the presence of missing *Players*' *Scores*.

In classic Elo ranking, the scores are updated after consecutive matches, therefore there is a natural order of updates. As the Elo points by which the winning player's score increases depends on his Elo and the opponent's Elo, the order in which the matches are played may affect the Elo's final score. However, it should be noted that the need to sequentially calculate Elo is due to the aforementioned weakness, which is the inability to play matches between all players at once. We propose an EPP model scoring method that does not require sequential calculation of match results and preserves the desired Elo properties, i.e. the possibility of interpretation on an interval scale.

It is worth noting that Elo [55] proposed a solution to the problem of how to measure the skill of all players with only partial information about the outcome of matches. The EPP score applies Bradley-Terry model [23] to machine learning models and therefore is a direct way to calculate the values approximated with Elo [41]. For that reason, the order of model comparisons is irrelevant for EPP.

5.3.2 Definiton of the EPP score

Now, we formally define the EPP meta-score in terminology of the unified EPP benchmark. Let $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ be a set of m *Players*. For a selected single *Tournament* T_t, we specify a set of r *Rounds* $\mathcal{R} = \{R_1, R_2, ..., R_r\}$ and *Score*.

Let denote the result of a single *Match* in a *Round* R_k between *Players* M_i and M_j as

$$y_{i,j}^k = \begin{cases} 1, & \text{where } \textit{Player } M_i \text{ has better } \textit{Score} \text{ value than } \textit{Player } M_j, \text{in } \textit{Round } R_k \\ 0.5, & \text{where } \textit{Player} M_i \text{ has the same } \textit{Score} \text{ value as } \textit{Player} M_j, \text{in } \textit{Round } R_k \\ 0, & \text{otherwise,} \end{cases}$$

and $\sum_{k=1}^{r} y_{i,j}^k$ is the number of wins the *Player* M_i over the *Player* M_j in all *Rounds*. The *Scores* are usually continuous, so the probability of a tie is near 0. Therefore, the empirical probability of winning in a random *Round* is equal

$$p_{i,j} = \frac{\sum_{k=1}^{r} y_{i,j}^k}{r}.$$

Definition 5.3.1. The odds(i,j) are odds that *Player* M_i has a better *Score* than *Player*

M_j, and are expressed as

$$odds(i,j) \;=\; \frac{p_{i,j}}{1-p_{i,j}},$$

where $p_{i,j}$ is the probability that *Player* M_i has a better *Score* than *Player* M_j in a random *Round* R.

Definition 5.3.2. The β_{M_i} and β_{M_j} are EPP *Meta-Scores* for *Players* $M_i, M_j \in \mathcal{M}$ respectively if they satisfy the following property

$$\log \frac{p_{i,j}}{1-p_{i,j}} = \beta_{M_i} - \beta_{M_j},$$

where $p_{i,j}$ can be estimated $\hat{p}_{i,j}$ in two exploratory variables logistic regression of the form

$$\log \frac{\hat{p}_{i,j}}{1-\hat{p}_{i,j}} = \hat{\beta}_{M_i} x_{M_i} + \hat{\beta}_{M_j} x_{M_j}, \qquad where \;\; x_{M_i} = 1 \text{ and } x_{M_j} = -1,$$

where $\hat{\beta}_{M_i}$ and $\hat{\beta}_{M_j}$ are estimated EPP *Meta-Scores*. For brevity, in the following sections, we refer to them simply as EPP *Meta-Scores*.

Definition 5.3.3. The EPP *Meta-Score Leaderboard* for *Tournament* T is the set of EPP *Meta-Score* values for the set of m *Players* $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ is

$$L_{\mathcal{M}}^{T} = \left(\hat{\beta}_{M_1}, \ldots, \hat{\beta}_{M_m}\right).$$

The properties of EPP *Meta-Score* are in the next subsection.

5.3.3 EPP handles the problems with common ML performance measures

In this section, we identify problems with the most common performance measures in ML benchmarks and we show that the EPP *Meta-Score* handles these issues. The attributes of EPP *Meta-Score* may be described with three aspects:

- EPP is a meta-approach based on values of other performance measures. EPP broadens the possibilities of comparing *Players* because of its unique properties

introduced in Sections 5.3.3 - 5.3.3;

- EPP is an alternative approach to aggregating results, such as mean scores for repeated measurements - *Rounds*. EPP gives the statistical possibility to assess the stability of *Scores* (see Sections 5.3.3 and 5.3.3);

- Unlike the methods used so far, EPP gives the possibility to compare Benchmarks. It allows the assessment of the quality of *Leaderboards* across *Tournaments*. (see Section 5.3.3).

The following sections are constructed as follows: first we discuss a problem with a real-life benchmark using terminology from the field of machine learning. In the second part of each section, we discuss at a general level and describe theoretical properties of the EPP that addresses the problem at hand. We describe such properties in terms of the EPP Benchmark. This distinction helps to better separate the examples from the theoretical part.

There is no interpretation of differences in performance

Table 5.4: Springleaf Marketing Response Kaggle Competition, `https://www.kaggle.com/c/springleaf-marketing-response`

Team Name	AUC
Asian Ensemble	0.80925
ARG eMMSamble	0.80907
.baGGaj.	0.80899

In Table 5.4, we show an example of Kaggle ranking. The difference between AUC value of the first and AUC value of the second model equals 0.00018. This absolute difference gives us no additional information. The AUC is useful for ordering models, but its differences have no interpretation, it does not provide any quantitative comparison of models' performances. There is no single accepted way to compare the power of enhancement of performance measures. Some say we should equate absolute differences regardless of the absolute values of the score, while others would suggest analyzing relative improvement. Both ways may lead to opposite conclusions, depending on the absolute value of a performance measure.

This ranking fits to the EPP Benchmark *Scheme* Model/Task with just one *Round*. EPP score provides the direct interpretation in terms of probability.

Property 5.3.1. The difference of EPP scores for *Players* M_i and M_j is the logit of the probability that M_i achieves better performance than M_j.

Indeed, from Definition 5.3.2 we have that

$$logit\left(\hat{p}_{i,j}\right) = \log\left(\widehat{odds}(i,j)\right) = \hat{\beta}_{M_i} - \hat{\beta}_{M_j}. \tag{5.1}$$

After reformulating Equation 5.1 we achieve direct formula for probability that *Player* M_i achieves better performance than *Player* M_j:

$$\hat{p}_{i,j} = invlogit\left(\hat{\beta}_{M_i} - \hat{\beta}_{M_j}\right) = \frac{\exp\left(\hat{\beta}_{M_i} - \hat{\beta}_{M_j}\right)}{1 + \exp\left(\hat{\beta}_{M_i} - \hat{\beta}_{M_j}\right)}.$$

There is no procedure for assessing the significance of the difference in performances

In Table 5.5, there are results of an IEEE-CIS Fraud Detection Kaggle Competition. The AUC values of all models in Table 5.5 differ in the third decimal place. There is no reference point to indicate whether this difference represents a significant improvement in prediction or not. Significance in the statistical sense means these differences are not on the noise level.

Table 5.5: IEEE-CIS Fraud Detection Kaggle Competition, `https://www.kaggle.com/c/ieee-fraud-detection`

Team Name	AUC
AlKo	0.968137
FraudSquad	0.967722
Young for you	0.967637

The scoring of models in this example are deployed in a *Scheme* Model/Task with just one *Tournament* and one *Round*. A similar situation with minor differences appears in many state-of-the-art benchmarks. When a *Player* gains improvement by a decimal place it would be desired to distinguish between real improvement and apparent improvement due to the noise coming from different *Round* setting e.g. splitting to train and test data. Currently, there are not many formal methods to assess the significance of differences. One way is to use a Kruskal-Wallis test for the equality of medians. But results from statistical tests are not transitive. We can compare two *Players*, but we would not get an overall *Leaderboard* for all of them.

EPP score allows for the assessment of the significance of score value, which gives an intuition whether the difference in performance is a noise or not.

Property 5.3.2. The values of the EPP *Meta-Scores* are coefficients of logistic regression model with intercept $\beta_0 = 0$.
The Equation 5.1 can be generalized to

$$logit(\hat{p}_{i,j}) = \hat{\beta}_{M_1} x_{M_1} + \hat{\beta}_{M_2} x_{M_2} + ... + \hat{\beta}_{M_k} x_{M_n}, \qquad where \;\; x_{M_a} = 1_{a=i} - 1_{a=j}. \tag{5.2}$$

$\hat{\beta}_{M_i}$ is the estimation of unknown β_{M_i} coefficients from the multiple exploratory variables logistic regression where x_{M_a} indicates if the *Player* is compared.

Because of calculating values of EPP *Meta-Score* from logistic regression, a logit of probabilities gives an additional benefit in the form of gaining a significance of EPP scores. This is an advantage over raw empirical probabilities.

Property 5.3.3. The statistical significance of the difference between EPP for two *Players* M_i and M_j may be tested as the null hypothesis that

$$\hat{\beta}_{M_i} = \hat{\beta}_{M_j}.$$

If *Round* performances are independent and sample size is sufficiently, this hypothesis may be tested with Wald test or Likelihood ratio test.

However, even when assumptions about independence of splits are violated and observations appear in different bootstrap samples, one can rely on tests results as they are robust. Another way is to use approximately unbiased bootstrap resampling [193, 194, 206].

There is no way to compare *Scores* between *Tournaments*

In Tables 5.4 and 5.5 differences between second and third best models for each dataset are around 0.00008. The question arises as to whether these differences are comparable between datasets. Does 0.00008 on Springleaf Marketing data mean the same increase of model quality on IEEE-CIS Fraud data?

There are at least three points of view. One is that the gaps are almost the same for both datasets, because the differences in AUC values are almost the same. The second is that the gap in the IEEE-CIS Fraud Competition is larger as the AUC value is close

to 1. Relative improvement for Fraud detection $\left(\frac{0.967722-0.967637}{1-0.967722} \approx 0.0026\right)$ is larger than relative improvement for Springleaf Marketing $\left(\frac{0.80907-0.80899}{1-0.80907} \approx 0.0004\right)$. The third point of view is that the gap between first and second place for Springleaf (0.00018) is smaller than the same difference for IEEE-CIS Fraud detection (0.000415). Therefore, the relative gain from the difference between second and third place for Springleaf is higher.

From the definition of EPP score, the probability of winning against an average *Player* (equivalent to an intercept $\hat{\beta}_0$) has the same meaning, regardless of the *Tournament*. The EPP scores are absolute values with a mean equals to zero. Therefore, comparison of EPP values between *Tournaments* is possible by comparing a probability of winning against an average *Player*.

Property 5.3.4. Probability that *Player* M_i would win against an average *Player* M_{avg} is

$$\hat{p}_{i,avg} = \frac{\exp\left(\hat{\beta}_{M_i}\right)}{1 + \exp\left(\hat{\beta}_{M_i}\right)},$$

from the Property 5.3.2 we have that intercept $\hat{\beta}_0 = 0$. In the logistic regression, intercept relates to the mean, therefore $\hat{\beta}_{M_{avg}} = 0$ and

$$\hat{p}_{i,avg} = invlogit\left(\hat{\beta}_{M_i} - \hat{\beta}_{avg}\right) = \frac{\exp\left(\hat{\beta}_{M_i} - \hat{\beta}_{M_{avg}}\right)}{1 + \exp\left(\hat{\beta}_{M_i} - \hat{\beta}_{avg}\right)} = \frac{\exp\left(\hat{\beta}_{M_i}\right)}{1 + \exp\left(\hat{\beta}_{M_i}\right)}.$$

Mean aggregation may be misleading: the Variance for *Round*

In Figure 5.2, we show four selected models from the Visual Task Adaptation Benchmark (VTAB) [235]. Every small point indicates the top-1 accuracy for one model on one of 19 specified datasets. The scores corresponding to the same dataset are connected with the thin lines.

We analyse the results of these models within pairs. The first pair is the Sup-Rotation-100% model and Sup-Exemplar-100% model. The second pair is Uncond-BigGAN model and VAE model. Averaged top-1 accuracy across datasets are close to to models in these pairs. The Sup-Rotation-100% and Sup-Exemplar-100% have evidently higher predictive power than Uncond-BigGAN and VAE models for most tasks.

Top-1 accuracy scores of Sup-Rotation-100% and Sup-Exemplar-100% models are very

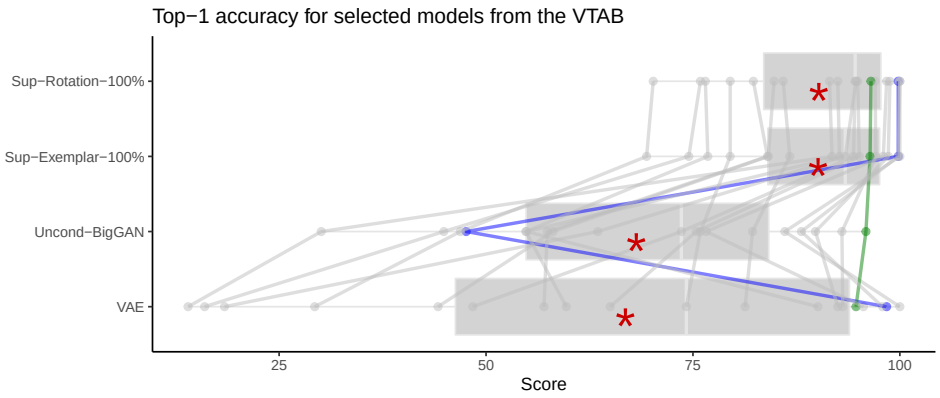

Figure 5.2: Boxplots of scores for four selected models from the Visual Task Adaptation Benchmark. Red stars correspond to mean scores across all VTAB tasks. Dots indicate scores in separate VTAB tasks. Thin lines connect points corresponding to the same task. Green and Blue lines are highlighted for example purposes in Section 5.3.3.

close to each other within specified datasets. This is represented as parallel lines connecting dots for two models in the top part of the plot. In the second pair, the relationship between Uncond-BigGAN and VAE is more ambiguous. Comparing just averages across datasets show that the Uncond-BigGAN model is comparable to VAE. However, when considering the green path in Figure 5.2, all four models have similar performance. On the other hand, when considering the blue path, VAE significantly defeats Uncond-BigGAN and performs comparably to Sup-Rotation-100% and Sup-Exemplar-100%. The probable reason for these two scenarios is the different tunability of datasets, which exhibits as diverse variance of top-1 accuracy for different models.

The VTAB ranking exemplifies the *Scheme* Model/Task of EPP Benchmark with just 19 *Rounds* determined with datasets. Averaging the *Score* values neglects the information about the distribution of *Score* values within *Rounds*. This is especially relevant in cases where we compare *Scores* of different definition and range of values for a sequence of *Rounds*. EPP also ignores the dispersion of *Score* values for a specified *Round* but this simplification comes, by design, from the definition of EPP computing.

Property 5.3.5. The EPP score is an aggregate over all rounds.

By fitting the logistic regression model from Equation 5.2 as dependent variables of observations, we use the results of *Matches*- whether one *Player* beats another.

The mean aggregation may be misleading: the variance for *Player*

As we see from the example of VTAB rankings in Figure 5.2, variation of VAE scores are higher than in the case of Uncond-BigGAN. Averaging the *Score* values neglects the information about the variance of *Score* values within *Player* results. Sometimes the standard deviation of *Scores* is used as description of stability of results but when *Scores* values come from various distribution depending the *Round* this summary is incorrect.

EPP benchmark from the definition unifies the definition of all *Matches* in the early phase and EPP *Meta-Score* is computed on the base of win/lose results coming from binomial distribution. Because of that, there is the procedure of assessing the stability of EPP *Meta-Scores* values.

Property 5.3.6. The $(1-\alpha)\%$ confidence interval of EPP for *Player* M_i is equal

$$CI^{\alpha}_{\hat{\beta}_{M_i}} = \left(\hat{\beta}_{M_i} - z_{\alpha/2} SE_{\hat{\beta}_{Mi}}, \hat{\beta}_{M_i} + z_{\alpha/2} SE_{\hat{\beta}_{Mi}}\right),$$

where $z_{\alpha/2}$ is the percentile of standard normal distribution.

If *Round* performances are independent and sample size is sufficient, the standard deviation of EPP score $\left(SE_{\hat{\beta}_{Mi}}\right)$ may be computed with maximum likelihood estimation method. Otherwise, the non-parametric bootstrap may be applied.

You cannot assess the quality of a *Leaderboard*

In the VTAB example in Figure 5.2, mean aggregation summarises the model performance with a single value. In Figure 5.2 we see that the residual values between *Score* value corresponding to a single task and mean value for Sup-Exemplar-100% model are much smaller than for Uncond-BigGAN or VAE. Averaging the top-1 accuracy values does not provide any statistics on how compatible the aggregated ranking is with rankings related to a single task, in other words how much information is lost in aggregation.

Assessing the quality of aggregation is crucial because the main objective is to create a *Leaderboard* emulating the relative performance power of models. If aggregated ranking is consistent with ranking for single tasks, it is reliable and this aggregation may be considered representative. Otherwise, the aggregated ranking is not an appropriate approach and a different aggregation measure or aggregation in subgroups of tasks should be considered. So far, aggregation methods which have been used to summarise many rankings

do not give any measure reflecting the quality of aggregation.

EPP *Meta-Score* is computed on the basis of logistic regression models. From the definition of the generalized linear model, there are tools to assess the quality of the aggregation procedure. We may compare the estimated probability $\hat{p}_{i,j}$ with the true values $p_{i,j}$ using likelihood function.

Property 5.3.7. The deviance of logistic regression for EPP *Meta-Score* is a measure of goodness-of-fit of a *Leaderboard* L for *Players* $\mathcal{M}$ and *Tournament* T.

$$D\left(\mathbf{p}_{\mathcal{M}}^{T}, \hat{\mathbf{p}}_{\mathcal{M}}^{T}\right) = 2\left(\log\left(\mathcal{L}(\mathbf{p}_{\mathcal{M}}^{T}, \mathbf{p}_{\mathcal{M}}^{T})\right) - \log\left(\mathcal{L}(\mathbf{p}_{\mathcal{M}}^{T}, \hat{\mathbf{p}}_{\mathcal{M}}^{T})\right)\right),$$

where $\mathbf{p}_{\mathcal{M}}^{T} = (p_{i,j})$ is a vector consisting of the actual empirical probability of winning for every pair of *Players* M_i and M_j in *Tournament* T and $\hat{\mathbf{p}}_{\mathcal{M}}^{T} = (\hat{p}_{i,j})$ is a vector consisting of the predicted value of the probability of winning, respectively. $\mathcal{L}\left(\mathbf{p}_{\mathcal{M}}^{T}, \mathbf{p}_{\mathcal{M}}^{T}\right)$ stands for the logistic regression likelihood function for a saturated model that provides a separate parameter for each observation and is the best fitted model and $\mathcal{L}\left(\mathbf{p}_{\mathcal{M}}^{T}, \hat{\mathbf{p}}_{\mathcal{M}}^{T}\right)$ is the logistic regression likelihood function for a considered model.

In addition, as the number of rounds $R \to \infty$ deviance converges in distribution to chi-square distribution $D\left(\mathbf{p}_{\mathcal{M}}^{T}, \hat{\mathbf{p}}_{\mathcal{M}}^{T}\right) \xrightarrow{\mathcal{D}} \chi^2_{m(m-2)}$, where m is the cardinality of *Players* set $\mathcal{M}$.

If the deviance of the *Leaderboard* is low, the EPP *Meta-Scores* and respective estimated probability $\hat{p}_{i,j}$ are close to empirical probability $p_{i,j}$ and this *Leaderboard* is more reliable than a *Leaderboard* with high deviance. In general, we cannot provide the absolute threshold that indicates whether the deviance statistic is low and *Leaderboard* is reliable. Yet, note that asymptotic distribution is chi-square and the number of degrees of freedom depends only on the number of *Players*, therefore for the same set of *Players* we can compare the relative quality of *Leaderboards* in *Tournaments*.

Property 5.3.8. Given the set of $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ *Players* and two *Tournaments* T_1 and T_2, the quality of EPP *Meta-Score Leaderboards* $L_{\mathcal{M}}^{T_1}$ and $L_{\mathcal{M}}^{T_2}$ can be compared using the deviance statistics. If

$$D\left(\mathbf{p}_{\mathcal{M}}^{T_1}, \hat{\mathbf{p}}_{\mathcal{M}}^{T_1}\right) \leq D\left(\mathbf{p}_{\mathcal{M}}^{T_2}, \hat{\mathbf{p}}_{\mathcal{M}}^{T_2}\right),$$

the *Leaderboard* $L_{\mathcal{M}}^{T_1}$ for *Tournament* T_1 is better fitted to actual probabilities than the *Leaderboard* $L_{\mathcal{M}}^{T_2}$ for *Tournament* T_2.

The number of degrees of freedom of deviance statistic is order of m^2, therefore deviance statistics can take very large values if the number of *Players* m is in the order of 100 or higher. From χ^2 distribution properties, the skewness of this asymptotic distribution decreases with the number of *Players* m. If the number of *Players* is sufficiently high, $D\left(\mathbf{p}_{\mathcal{M}}^{T_1}, \hat{\mathbf{p}}_{\mathcal{M}}^{T_1}\right)$ converges in distribution to normal distribution $\mathcal{N}(m(m-2), 2m(m-2))$. The deviance statistics can be scaled and shifted depending on the number of *Players*.

Property 5.3.9. Standardized deviance statistics for set of *Players* $\mathcal{M}_i$ and *Tournament* T_j is

$$\widetilde{D}\left(\mathbf{p}_{\mathcal{M}_i}^{T_j}, \hat{\mathbf{p}}_{\mathcal{M}_i}^{T_j}\right) = \frac{D\left(\mathbf{p}_{\mathcal{M}_i}^{T_j}, \hat{\mathbf{p}}_{\mathcal{M}_i}^{T_j}\right) - m(m-2)}{\sqrt{2m(m-2)}},$$

where m is the cardinality of *Players* set $\mathcal{M}_i$.

If the number of rounds $R \to \infty$ deviance and m is sufficiently large, $\widetilde{D}\left(\mathbf{p}_{\mathcal{M}_i}^{T_j}, \hat{\mathbf{p}}_{\mathcal{M}_i}^{T_j}\right) \xrightarrow{\mathcal{D}} \mathcal{N}(0,1)$.

It is worth emphasizing that the transformation of deviance statistics is done by constants, depending only on the number of *Players*. We do not need to estimate additional coefficients.

Due to the approximation with a standard normal distribution, we can compare the loss between estimated probabilities and the observed values in different *Schemes* of *Tournaments*, but not necessarily for the same set of *Players*.

Property 5.3.10. Given the set of $\mathcal{M}_1 = \{M_1^1, M_2^1, ..., M_m^1\}$ *Players* in *Tournament* T_1 and the set of $\mathcal{M}_2 = \{M_1^2, M_2^2, ..., M_n^2\}$ *Players* in *Tournament* T_2, the quality of EPP *Meta-Score Leaderboards* $L_{\mathcal{M}_1}^{T_1}$ and $L_{\mathcal{M}_2}^{T_2}$ can be compared using the deviance statistics. If

$$\widetilde{D}\left(\mathbf{p}_{\mathcal{M}_1}^{T_1}, \hat{\mathbf{p}}_{\mathcal{M}_1}^{T_1}\right) \leq \widetilde{D}\left(\mathbf{p}_{\mathcal{M}_2}^{T_2}, \hat{\mathbf{p}}_{\mathcal{M}_2}^{T_2}\right),$$

the *Leaderboard* $L_{\mathcal{M}_1}^{T_1}$ for *Tournament* T_1 is better fitted to actual probabilities than *Leaderboard* $L_{\mathcal{M}_2}^{T_2}$ for *Tournament* T_2. Which means that *Leaderboard* $L_{\mathcal{M}_1}^{T_1}$ captures the relationship between models' performance better than *Leaderboard* $L_{\mathcal{M}_2}^{T_2}$.

An alternative approach to comparing deviance statistics is the comparison of p-values corresponding to chi-square distribution (Property 5.3.8) or for standard normal distribution (Property 5.3.10) for deviance statistics. The higher p-value corresponding to deviance statistics indicates that the leaderboard fits more accurately. However, with the increase of the number of *Players*, we may observe the discretization of p-values related to deviance for tournaments.

5.4 Real data examples

In this section, we show that EPP *Meta-Score* improves the existing real data benchmarks for tabular data as well as Computer Vision and Natural Language Processing problems.

Firstly, we regard the OpenML100 benchmark for table data [19], which fits the Model/CV scheme from Section 5.2.1. Every model (*Player*) is tested on a different train/test split of cross-validation (*Round*). The rankings are created per dataset (*Tournament*).

There is a different perspective in VTAB and SuperGlue benchmarks. Similarly to the OpenML, the compared items are different neural architectures - *Players*. But the *Rounds* are determined by different independent tasks.

5.4.1 Computing EPP on the OpenML100 benchmark

Now, we demonstrate the advantages of EPP *Meta-Scores* in *Leaderboards* on a MementoML [122] database, that is a large-scale benchmark on 30 binary classification datasets from the OpenML. We selected 5 machine learning algorithms: gradient boosting machines (`gbm`), a generalized linear model with regularization (`glmnet`), k-nearest neighbors (`kknn`), and two implementations of random forest (`RF` and `ranger`). Each algorithm was trained with 400 different, randomly chosen hyperparameter configurations. For each

dataset, models were tested on 20 random train/test splits with AUC as a performance measure. This gave us an overall number of AUC values equal to $30 \cdot 20 \cdot 5 \cdot 400 = 1200000$.

On the computed AUC *Scores*, we calculated the EPP *Meta-Score* introduced in Section 5.3 and Figure 5.1. As a single *Round*, we consider a train/test split. A *Match* is a comparison of performances of two models with specified hyperparameters on the same dataset, yet not necessarily on the same train/test split. As a result, we have obtained EPP values for each data-model-hyperparameter combination, which gave us $30 \cdot 5 \cdot 400 = 60000$ values of EPP *Meta-Scores*.

The performance of models is highly variable due to the dataset, which can be seen in Figure 5.3, where the distributions of EPP *Meta-Score* values across models and datasets are shown. The longer boxplot means greater potential for model tuning; for example, we can see that tree-based models (`gbm` , `RF`, and `ranger`) perform better on dataset *madelon* than the other two models. Also, all of the EPP values for random forest are positive, which means that, generally, the performance of random forest is above the average. Due to the independent sampling of hyperparameters and an excessive L_1 penalty in regularization, a part of the `glmnet` models achieved AUC score equal to 0.5 or less. The models with AUC=0.5 always lose against other models, which causes a huge range of values of EPP scores for `glmnet`.

In Table 5.6, we show AUC and EPP values for the four selected models for the *ada_agnostic* dataset from experiments described earlier. To recall, in Section 5.3.3, in the example of Kaggle ranking, we postulate that AUC score does not provide a probability interpretation. EPP addresses this issue, so we can assess the probability of one model winning against another model according to Property 5.3.1. The descending order according to the averaged AUC is different from EPP ranking. The lowest EPP value has the `kknn` model, even though the lowest averaged AUC corresponds to `glmnet`. The difference between AUC of the first and AUC of the second model equals 0.001497, the difference between AUC of the third model and the fourth score equals 0.00395. Due to EPP, we can estimate the probability that `gbm` beats `ranger` with $logit(1.27 - 1.08) \approx 0.55$ probability. In the second pair of `glmnet` and `kknn` models, despite the close to averaged AUC, there is a 0.83 likelihood that `glmnet` will defeat the `kknn` model. These dissimilarities are not emphasized by AUC score, since the averaged crossvalidation scores miss the variability of metrics.

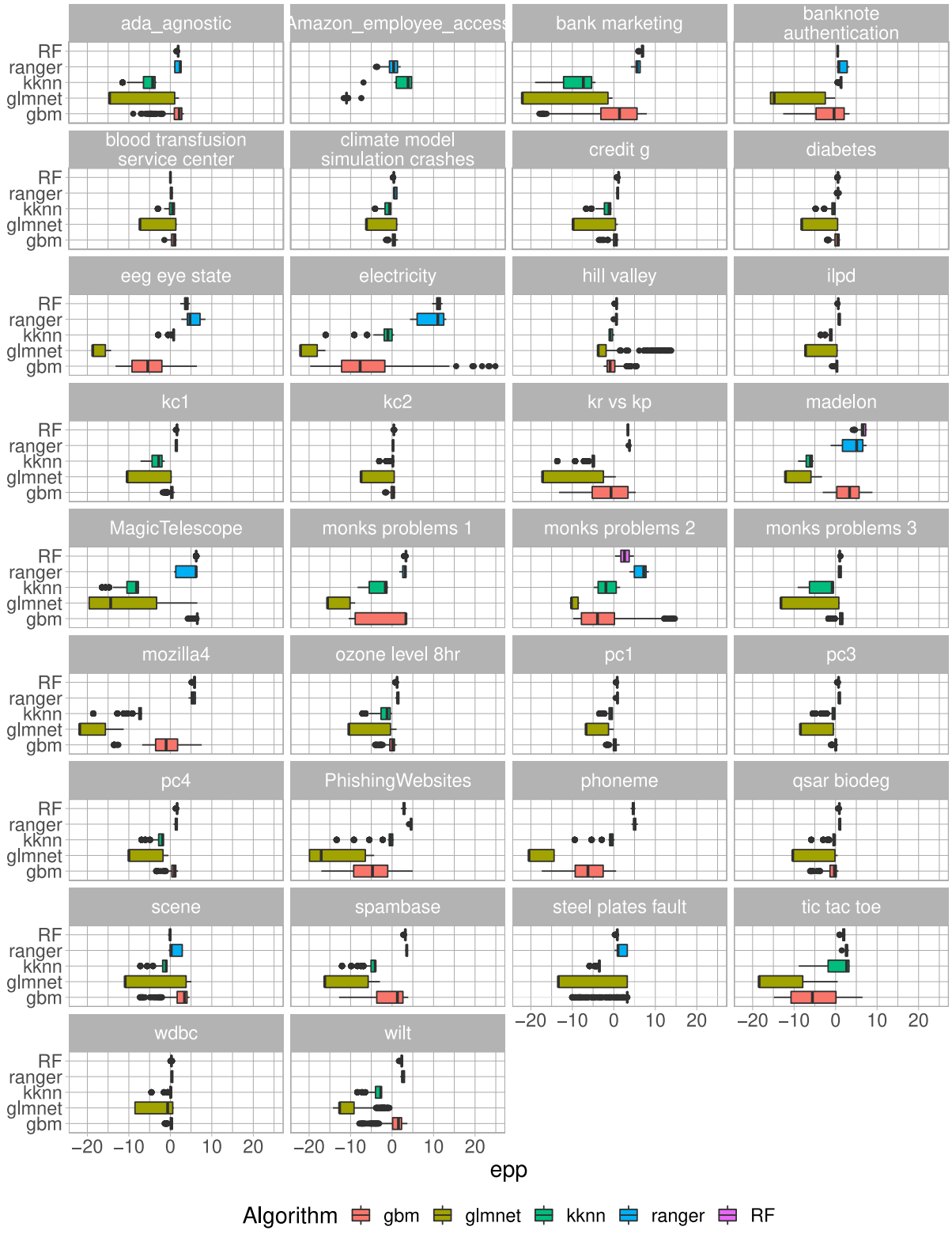

Figure 5.3: Boxplots of EPP scores for different algorithms across datasets. Each boxplot aggregates EPP scores of all models trained on all datasets.

Table 5.6: EPP of selected models for *ada_agnostic* dataset. AUC values are averaged.. The numbers of models are IDs from the MementoML benchmark.

Model	AUC	EPP
`gbm1305`	0.890	1.27
`ranger1088`	0.888	1.08
`knn1396`	0.816	-7.52
`glmnet1242`	0.812	-5.91

Table 5.7: The best models in algorithm class for *mozilla4* dataset. AUC values are averaged. The numbers of models are IDs from the MementoML benchmark.

Model	AUC	EPP
`gbm1184`	0.986	7.49
`ranger1106`	0.984	6.25
`RF1106`	0.984	6.22
`knn1016`	0.942	-6.78
`glmnet1011`	0.922	-11.24

With respect to Property 5.3.4, EPP *Meta-Score* enables the analysis of performances between datatsets. Because of the lack of interpretation of AUC differences, comparison between model scores may be made in various ways, as described in detail in Section 5.3.3. Table 5.7 and Table 5.8 present rankings for best-in-class models for two datasets from our experiment, *mozzilla4* and *credit-g*. Even though absolute differences of AUC between the first and second model in each ranking are around 0.002, the rankings have different levels of AUC scores (approximately 0.98 and 0.81 for *mozzilla4* and *credit-g* respectively) so distinct approaches provide dissimilar claims. The EPP overcomes this problem and we can draw consistent conclusions regardless of the absolute value of a considered metric. Due to differences in EPP values, *mozzilla4* dataset `gbm` model has 0.77 probability that beats the best `ranger` model. In *credit-g* ranking, the `RF` model has only a 0.53 likelihood that it will defeat the `ranger` model.

Recalling Property 5.3.8, for every dataset (EPP *Leaderboard*) we can use the deviance statistics to compare the quality of rankings. The two leaderboards related to the lowest and the highest deviance of model computing EPP values are *banknote-authentication* and *wdbc* respectively. In Figure 5.4 we present the actual empirical probabilities of winning among every pair of models versus predicted probabilities computed on the base of EPP values. The exact fit should be placed on the black line plotted on the graph. The higher deviance statistics reflect the greater consistency with empirical results. In addition to the relative comparison of deviance statistics, we can compare the quality of

Table 5.8: The best models in algorithm class for *credit-g* dataset. AUC values are averaged. The numbers of models are IDs from the MementoML benchmark.

Model	AUC	EPP
`RF1155`	0.809	1.29
`ranger1212`	0.807	1.16
`gbm1136`	0.807	1.16
`glmnet1379`	0.802	0.97
`kknn1038`	0.769	-0.54

the resulting *Leaderboards* as a fit to empirical probability values. For 12 of the 33 *Tournaments*, the obtained model rankings are not statistically worse than the *Leaderboards* corresponding to perfect fits to the true observed odds of winning.

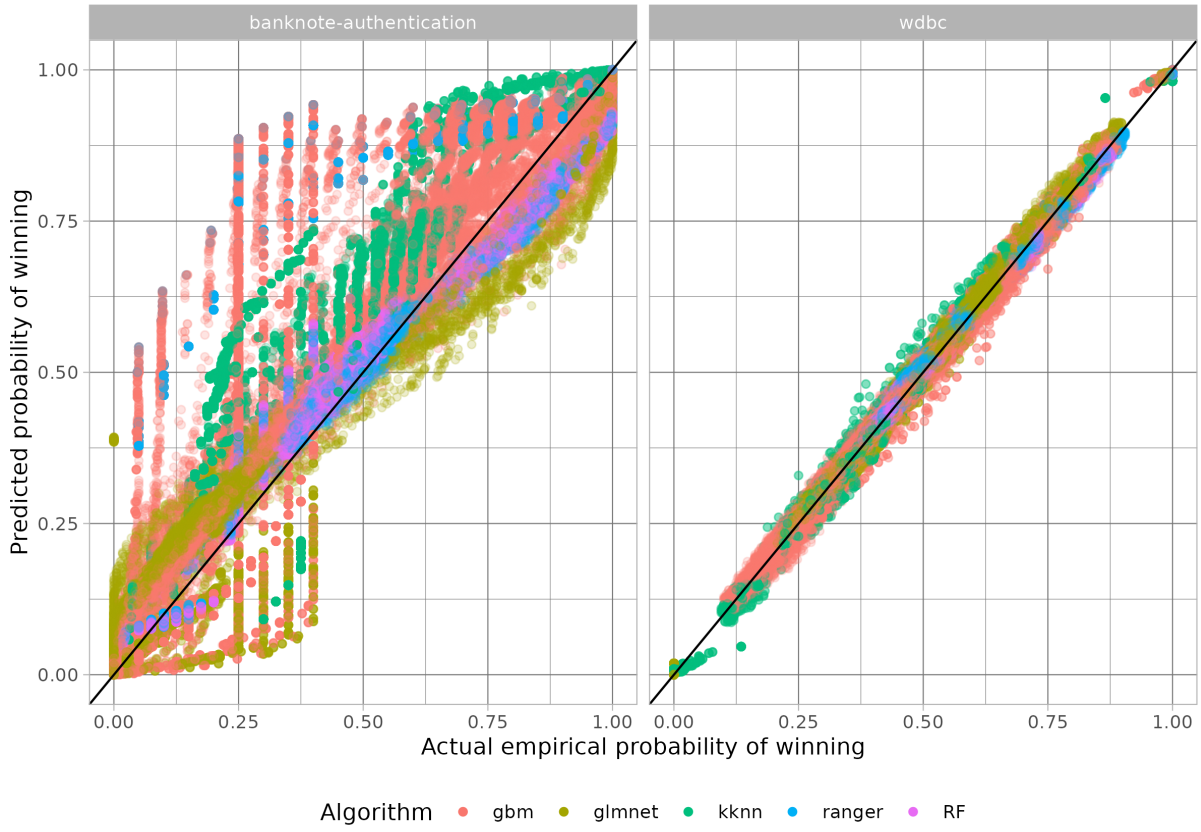

Figure 5.4: Actual empirical probability of winning and predicted probability computed on the basis of EPP meta-score value for datasets: *banknote-authentication* (least accurate fit according to deviance) and *wdbc* (the best fit according to deviance).

It is worth noting that the proposed scheme of EPP application in this benchmark is not fully consistent with the previously described ontology for Model/CV scheme, where the single *Round* is a single dataset split. Here, we consider as one round the comparison of *Scores* on different train/test splits. This approach allows for more matches between

players and the EPP *Meta-Score* values are more reliable. This extension is valid because the properties of crossvalidation and the assumption of estimating the same value of the performance measure across train/test folds. Next to this issue arises a question about the stability of EPP *Meta-Score* values and how many *Matches* are needed to estimate EPP.

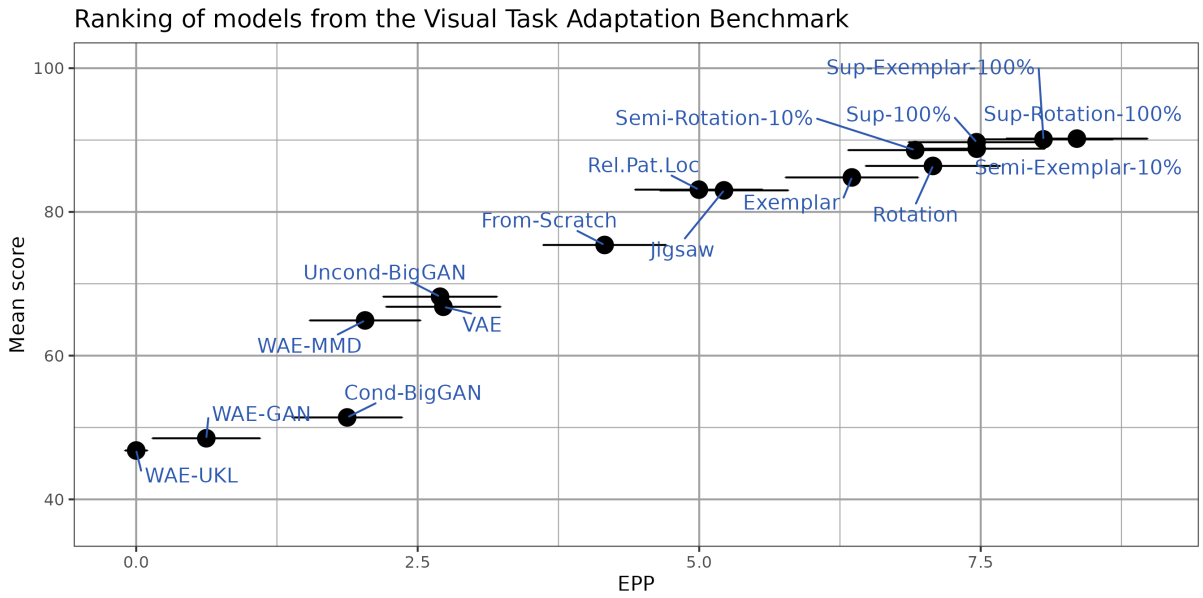

Figure 5.5: Elo and Mean of VTAB scores. Each black dot represents one model. Horizontal bars are confidence intervals for EPP scores.

5.4.2 Use-case on the Visual Task Adaptation Benchmark

The Visual Task Adaptation Benchmark (VTAB) [235] is a suite of tasks designed to evaluate general visual representations. The VTAB benchmark consists of 16 different architectures. Each architecture is evaluated on 19 datasets. The overall score of architecture is the mean of scores across datasets. In terms of the Unified Benchmark Ontology from Section 5.2, VTAB fits *Scheme* Model/Task.

In Figure 5.5, we show the comparison of the mean VTAB score and EPP meta-score for each model. The overall trend for the mean score and EPP is similar; however, there are some differences in the rankings. For example, Semi-Rotation-10% has a higher mean score than Rotation, but lower EPP. It is caused by the fact that EPP only takes into account whether a model is better or worse than another, while the mean depends on the differences in results.

The independence of tasks in the benchmark allows for the computing of confidence

intervals that show whether difference in EPP scores is significant. The analysis of confidence intervals in Figure 5.5 distinguishes groups of models that truly differ in performance. In particular, mean scores and EPP meta-scores for Semi Rotation 10% and Rotation models differ, while EPP confidence intervals overlap, which means we cannot state that there is a difference in model performances. Additionally, due the Property 5.3.3, we can test the differences between two models. The p-value of the Wald test between Semi Rotation 10% and Rotation at the significance level 0.05 equals 0.65. Therefore, there is no significant difference between these models' performances.

In addition to group analysis, one can also compare pairs of models. The mean for the top 2 models is almost the same; however, the EPP *Meta-Scores* can be used to calculate the probability that on a new dataset Sup-Rotation-100% will perform better than Sup-Exemplar-100%. The probability of winning is the inverse logit of the difference of scores (see Property 5.3.1). Therefore, Sup-Rotation-100% (EPP=3.41) will obtain higher performance than Sup-Exemplar-100%g (EPP=3.16) with the probability equal to $\frac{\exp(3.41-3.16)}{1+\exp(3.41-3.16)} = 0.56$.

5.5 Conclusions and future work

In this work, we introduced a new performance meta-score, the EPP. By introducing the Unified Benchmark Ontology, we demonstrated how universal and applicable the EPP measure is across different machine learning domains. In addition, we highlighted the most important objections regarding existing metrics and pointed out EPP properties which cover these limitations.

The versatility feature combined with EPP statistical properties enhances the inference that comes from existing benchmarks, which is shown in the use-case of VTAB and OpenML. The most important is the possibility of transofming differently defined evaluation scores to the same scale, i.e. the probability of winning against a competitive model. On the VTAB benchmark we show that the EPP *Leaderboard* amplifies the original approach and provides a confidence interval for EPP value, so we are able to assess the significance of differences between architectures. On the basis of the OpenML repository we illustrate how EPP empowers the systematic benchmark with the well-defined space of machine learning models and hyperparameters. EPP *Meta-Score* enables the comparison of predictive power for different *Tournaments*, in this case datasets. *Tournaments* are

comparable in terms of quality of EPP *Leaderboards*.

Hence, EPP may be considered as competitive to commonly applied scores in rankings of machine learning challenges and as an alternative to existing approaches to aggregating scores. What is more, EPP extends the existing benchmarks and does not require to recompute them (see VTAB use-case in Section 5.4.2).

The EPP *Meta-Score* has statistical foundations and this should be the key aspect of further research. The need remains to examine how the interdependence of *Rounds* affects the EPP *Meta-Score* estimation and how many *Rounds* are required to obtain stable EPP values.

5.6 Future applications

We see several possible extensions of EPP score. The TrueSkill [89] Elo-based system allows the grading of human skills in games for more than two players; it can be applied to machine learning and used for assessing the performance of model ensembles. It could make it possible to assess separately the performance of a single model, performance of the ensemble of models, and the potential of the model in the ensembles.

Due to interpretation of differences and comparability of EPP across diverse datasets, new measures provide the opportunity to research and verify state-of-the-art AutoML benchmarks in a new light. So far, the researchers have to make assumptions to simplify finding optimal configuration of algorithm settings across multiple datasets [170]. The EPP *Meta-Score* does not require the same scale of score and adds an interpretation for comparing *Leaderboards*. The second major opportunity is to use EPP for navigated hyperparameter tuning. EPP score can be used to assess the probability that we can improve performance if we continue searching the hyperparameter space.

5.A Appendix

5.A.1 EPP fitting error

Comparison of the error of fit of the estimated probability to the observed proportion of matches won between each pair of players can be carried out using two approaches.

Deviance of EPP

Table 5.9: Deviance of EPP *Leaderboard* for every benchmarked dataset from OpenML described in Section 5.4.1. For all *Leaderboards*, `deviance` has χ^2 distribution with 3996000 degrees of freedom. `std.deviance` stands for normalized deviance computed by subtraction of the mean value of χ^2 distribution and then division by the standard deviation χ^2 distribution. `p.value` comes from one-sided t-test.

id	data set	deviance	std. deviance	pvalue
1043	ada_agnostic	4025457.0	10	<2e-16
1461	bank-marketing	3908379.0	-31	1
1462	banknote-authentication	43888510.0	14111	<2e-16
1464	blood-transfusion-service-center	2226673.0	-626	1
1467	climate-model-simulation-crashes	5423963.0	505	<2e-16
31	credit-g	2904705.0	-386	1
37	diabetes	1720012.0	-805	1
1471	eeg-eye-state	36066114.0	11344	<2e-16
151	electricity	14954722.0	3876	<2e-16
1479	hill-valley	13492159.0	3359	<2e-16
1480	ilpd	2915470.0	-382	1
1067	kc1	7259304.0	1154	<2e-16
1063	kc2	1339686.0	-940	1
3	kr-vs-kp	4166641.0	60	<2e-16
1485	madelon	5362849.0	484	<2e-16
1120	MagicTelescope	13854813.0	3487	<2e-16
333	monks-problems-1	20237849.0	5745	<2e-16
334	monks-problems-2	5349152.0	479	<2e-16
335	monks-problems-3	2175298.0	-644	1
1046	mozilla4	3762485.0	-83	1
1487	ozone-level-8hr	3631091.0	-129	1
1068	pc1	8165732.0	1475	<2e-16
1050	pc3	3409399.0	-208	1
1049	pc4	5706084.0	605	<2e-16
4534	PhishingWebsites	9131773.0	1817	<2e-16
1489	phoneme	4626573.0	223	<2e-16
1494	qsar-biodeg	3929698.0	-24	1
312	scene	7911292.0	1385	<2e-16
44	spambase	4605587.0	216	<2e-16
1504	steel-plates-fault	41210355.0	13164	<2e-16
50	tic-tac-toe	6676955.0	948	<2e-16
1510	wdbc	982508.0	-1066	1
1570	wilt	15151134.0	3946	<2e-16

EPP vs. Mean - Mean square error

In deviance approach, assumptions about the binomial probability distribution are taken into account. An alternative naive approach that ignores the distribution of estimated values is the mean square error (MSE). By ignoring theoretical assumptions it is possible to compare fitting error for the EPP method and the mean as the default aggregation of the chosen measure.

Since EPP coefficients are transformable to probabilities $\hat{p}_{i,j}$ (according to Property 5.3.1) and then

$$MSE_{EPP} = \sum_{i,j} (p_{i,j} - \hat{p}_{i,j})^2. \tag{5.3}$$

In the case of the mean aggregation we predict that a player wins with probability equal to 1 if the mean score is higher than that of the opponent. So mean square error for mean aggregation equals

$$MSE_{mean} = \sum_{i,j} \left(p_{i,j} - I\left(mean(score_i) \geq mean(score_j)\right)\right)^2. \tag{5.4}$$

In Table 5.10 we compare the MSE of EPP and mean aggregation for OpenML benchmark. For many data sets MSE for EPP method is many times lower than the baseline method (see Ratio column).

Simulations

To confirm results showing the superiority of EPP over Mean aggregation we run simulations changing the benchmark plan. In addition to the classical estimation of EPP coefficients using unpenalized logistic regression, we also use EPP estimation by Ridge regression. The comparison of both EPP estimation methods allows to empirically assess the quality of the fitting error for both approaches versus mean aggregation.

The simulations are conducted for 10, 50, 100, 200, 500 number of players and for 3, 5, 10, 15, 20, 30, 40, 50 rounds. Simulation for each configuration was repeated 12 times.

For each number of players ($n_{players}$) and number of rounds (n_{rounds}) we sample the AUC score, then create *Leaderboard* using EPP and mean aggregation. For all three methods, MSE is computes according to following steps:

1. Sample $\mu_0 \sim \mathcal{U}(0.7, 0.95)$ and $\sigma_0 \sim \mathcal{U}(0.005, 0.01)$, where $\mathcal{U}$ stands for uniform distribution. This value simulates the expected difficulty of task and the tunability of the task respectively.

2. For every $player_i$, $i = 1, \ldots n_{players}$, sample $\mu_i \sim \mathcal{N}(\mu_0, 0.01)$ and $\sigma_i \sim \mathcal{N}(\sigma_0, 0.001)$. These values simulate the expected score and its variance for player i.

3. For $player_i$ draw n_{rounds} values of a_i^k - score of performance (AUC) in k-th round

Table 5.10: Comparison of MSE values for EPP aggregation and Mean aggregation every benchmarked data set from OpenML described in Section 5.4.1.

id	data set	EPP MSE	Mean MSE	Ratio
1043	ada_agnostic	0.00016	0.058	372
1046	mozilla4	0.0001	0.037	356
1049	pc4	0.00022	0.071	324
1050	pc3	0.00015	0.097	626
1063	kc2	8.6×10^{-5}	0.15	1732
1067	kc1	0.00025	0.068	272
1068	pc1	0.0003	0.093	306
1120	MagicTelescope	0.04	0.12	2
1461	bank-marketing	0.00018	0.029	164
1462	banknote-authentication	0.019	0.13	6
1464	blood-transfusion-service-center	0.00017	0.1	604
1467	climate-model-simulation-crashes	0.00031	0.098	316
1471	eeg-eye-state	0.0014	0.027	19
1479	hill-valley	0.00053	0.075	140
1480	ilpd	0.0002	0.1	511
1485	madelon	0.00016	0.028	174
1487	ozone-level-8hr	0.0002	0.073	358
1489	phoneme	0.00012	0.039	337
1494	qsar-biodeg	0.00031	0.085	276
1504	steel-plates-fault	0.009	0.078	8
151	electricity	0.00089	0.019	21
1510	wdbc	0.00012	0.15	1301
1570	wilt	0.00049	0.059	120
3	kr-vs-kp	0.00014	0.05	357
31	credit-g	0.00016	0.088	549
312	scene	0.00024	0.053	217
333	monks-problems-1	0.017	0.1	6
334	monks-problems-2	0.00028	0.024	88
335	monks-problems-3	0.00031	0.086	282
37	diabetes	0.00011	0.12	1055
4135	Amazon_employee_access	0.00036	0.065	180
44	spambase	0.00013	0.054	422
4534	PhishingWebsites	0.00028	0.031	112
50	tic-tac-toe	0.0014	0.049	35

$(k = 1, \ldots, n_{rounds})$ with mean μ_i and variance σ_i sampled in Step 2.

4. On the base of drawn n_{rounds} values of AUC for each of $n_{players}$ calculate EPP_i and EPP_i^{Ridge} to create *Leaderboards*.

5. For estimated EPP_i, EPP_i^{Ridge} and mean value of AUC, compute MSE_{EPP}, $MSE_{EPP\ Ridge}$ and MSE_{mean} according to 5.3 and 5.4.

6. For every value of $n_{players}$ and n_{rounds} we repeat steps 1-5 12 times.

Figure 5.6 summarizes the results across number of players and number of rounds. EPP estimated by two methods is more accurate in MSE terms than mean aggregation.

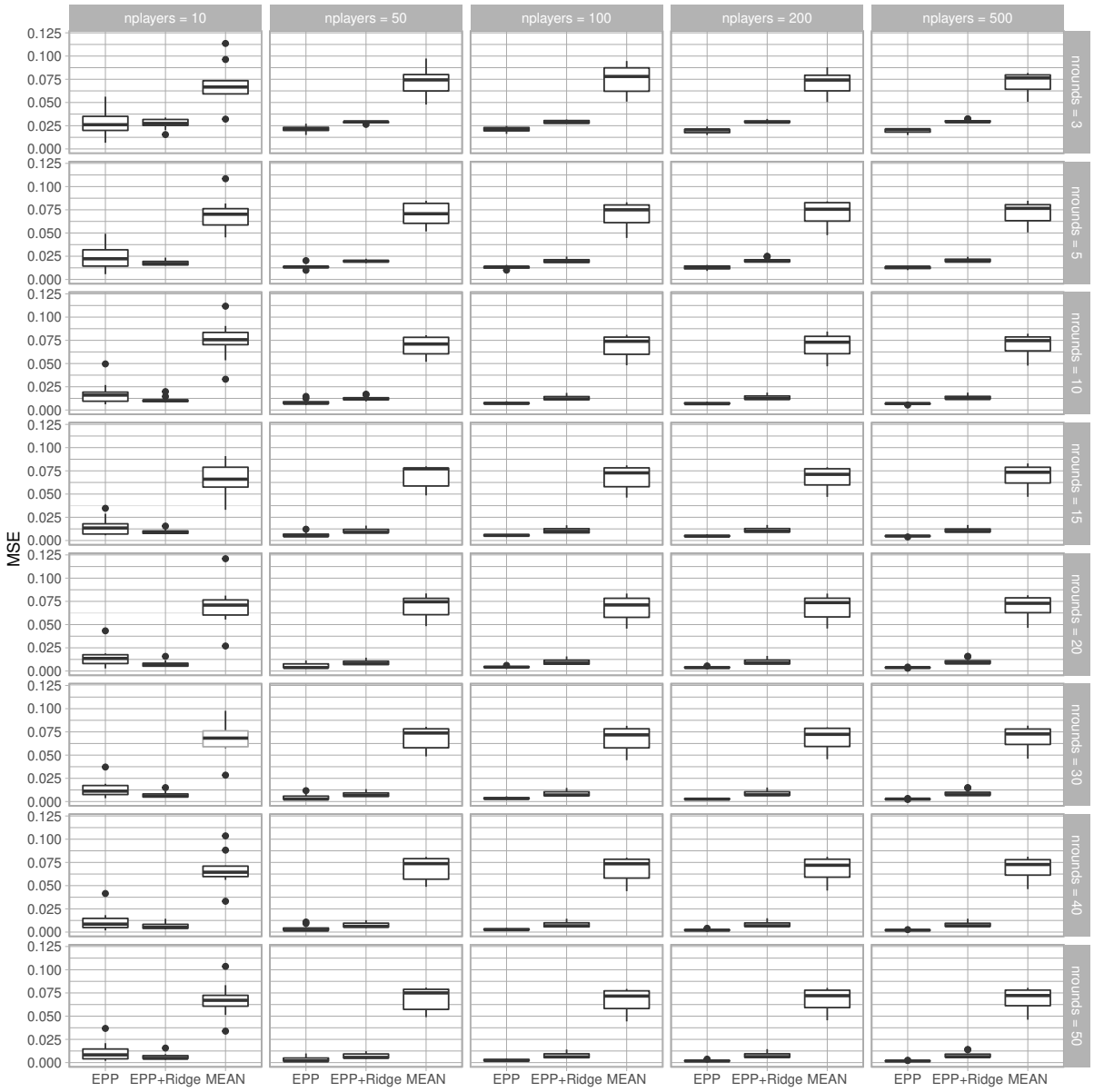

Figure 5.6: Boxplots of MSE values for EPP aggregation, EPP estimated with Ridge regression and Mean aggregation for $n = 12$ repetition per aggregation method. Horizontal bars indicate largest (smallest) value within 1.5 times the interquartile range above (below) the third (first) quartile. As the number of players and the number of rounds increase, the difference between EPP and Mean aggregation becomes more apparent. For low number of players EPP estimated with Ridge regression is more accurate than estimated with unpenalized regression.

5.A.2 Stability of EPP across rounds

The purpose of the following simulation is to investigate the stability of the EPP estimation as a function of the number of rounds and the number of players. Usually, due to limited availability of data, resampling methods, such as, cross-validation and bootstrap are used to correctly estimate the performances of models. As a result of the resampling mechanism, the scores obtained in subsequent rounds are correlated with each other.

In this experiment we investigated the behavior of the EPP, assuming that, the AUC values come from a multivariate normal distribution. All rounds have a constant expected value and a constant variance. Relationships between rounds are expressed by a fixed correlation value. In the experiment, we assumed the pessimistic case where the correlation between rounds is twice as large as their variance.

The simulation was conducted for 10, 50, 100, 200, 500 number of players and for 3, 5, 10, 15, 20, 30, 40, 50 number of rounds. The simulation for each configuration was repeated 10 times.

We simulate AUC for $n_{players}$ in the following steps:

1. Sample $n_{players}$ AUC mean from the uniform distribution $[0.5, 0.9]$.

2. For each player sample 50 values from the joint multivariate normal distribution with one of the means from step 1, and fixed covariance matrix with variances equal to 0.0002 (values on the diagonal of the covariance matrix) and covariances equal to 0.0001 (values off the diagonal). Thus, correlations between rounds equal 0.5. Samples simulate vectors of AUC values from bootstrap subsets in rounds.

3. For each of $n_{players}$ players and for each number of rounds $n_{rounds} \in \{3, 5, 10, 15, 20, 30, 40, 50\}$ take the first values of the simulated AUC. Calculate the EPP ranking for those values.

4. Repeat Steps 1-3 10 times.

Figure 5.7 shows the results of EPP stability simulation. Values on y-axis are epp_{diff} which are the mean of EPP difference of the same players relative to the previous number of rounds, which can be described by the following formula. Let $epp_{i,r}$ be an EPP value for i-th player in EPP ranking calculated on r rounds:

$$epp_{diff}(20) = \frac{1}{n_{players}} \sum_{i=1}^{n_{players}} (epp_{i,10} - epp_{i,20}).$$

Even in the case of dependence between rounds that we consider, the proposed estimation method provides stable EPP values. From the plot we can see that for smaller number of models (10, 50) EPP is stable for more than 40 rounds. For larger numbers of models (100, 200, 500) EPP is stable from 20 rounds.

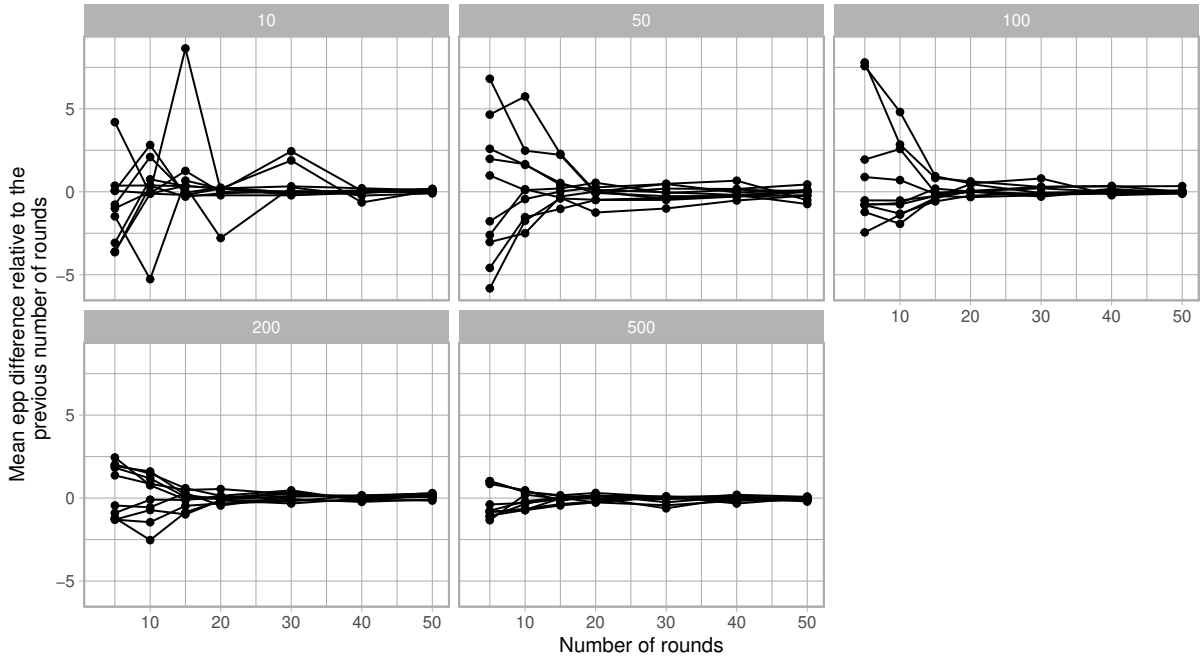

Figure 5.7: Simulation of the stability of EPP. Each panel shows results for different number of players. On the x-axis there is number of rounds. On y-axis is the mean difference of EPP for the same player relative to the previous number of rounds. The values on y-axis around 0 indicates that the values of EPP do not change when the number of rounds changes.

Figure 5.8 shows the changes in EPP distribution when the number of players increases. The ranges of boxplots for larger numbers of players do not differ from the ranges of boxplots for smaller numbers of players, therefore EPP do not inflate with the increasing number of players.

5.A.3 EPP stability in new player scenario

The aim of this simulation is to verify, whether EPP is stable when new player is added. We used values of $n_{players}$: 10, 50,100, 200, 500 and values of n_{rounds}: 3, 5, 10, 15, 20, 30, 40, 50.

We simulate AUC in the following steps:

1. Sample AUC mean from distribution N(0.8, 0.05).

2. Sample AUC variance from distribution N(0.08, 0.005).

3. Sample $rank_{base}$ that consists of AUC for $n_{players}$ and n_{rounds} with mean and variance sampled in Steps 1 and 2. Where mean and variance of AUC is different for each player and the same for each player's rounds.

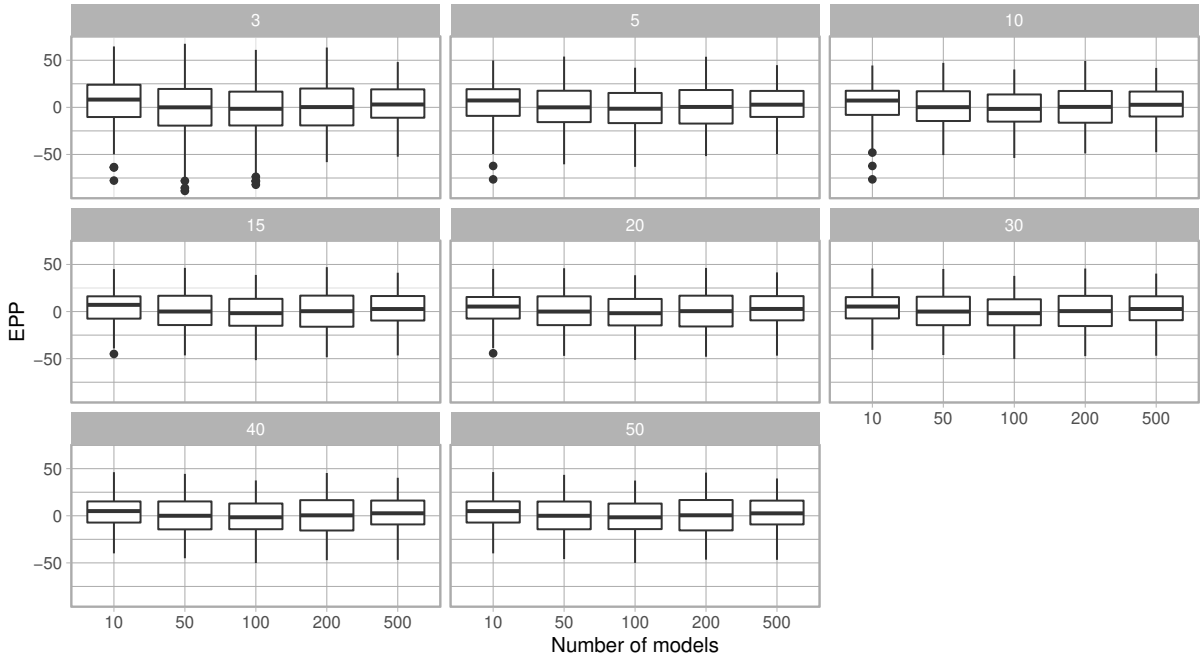

Figure 5.8: Simulation of the EPP inflation for the increasing number of players. Boxplots of EPP values split by number of players/models (n=10 per number of players) . Horizontal bars indicate largest (smallest) value within 1.5 times the interquartile range above (below) the third (first) quartile. Each panel is different number of rounds.

4. Calculate EPP_{base} for $rank_{base}$.

5. Generate $rank_{new}$ that consists of players from step 3 and $player_{new}$ with AUC=0.75

6. Calculate EPP_{new} for $rank_{new}$.

7. Calulate Kendall tau kt for model ranking from EPP_{base} and for model ranking from EPP_{new}, skipping $player_{new}$ in $rank_{new}$.

8. Repeat steps 1-7 100 times

Figure 5.9 shows differences in probabilities of winning between consecutive models before and after adding a new player. For more than 10 models differences are around 0, therefore addition of a new player do not disturb the relationships between models in the ranking.

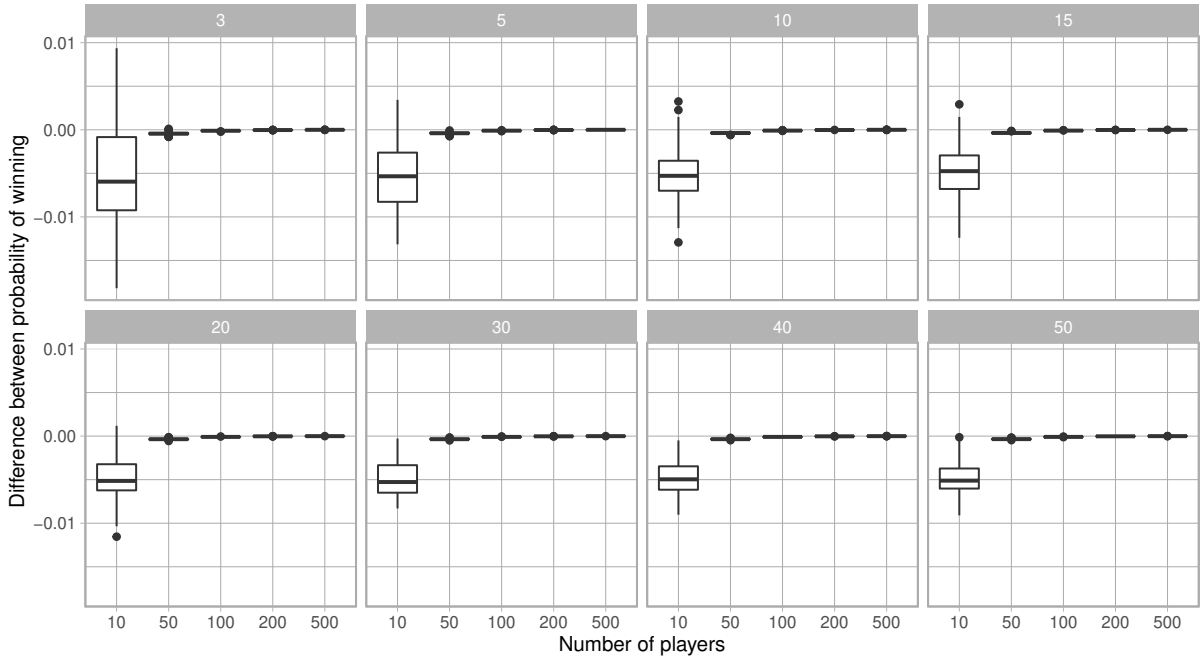

Figure 5.9: Difference of the probability of winning between models. Boxplots of differences of probabilities of winning between consecutive models before and after adding a new player split by number of players (n=100 per one number of players). Horizontal bars indicate largest (smallest) value within 1.5 times the interquartile range above (below) the third (first) quartile. Panels show different number of rounds.

Part II

Integrating domain knowledge into the AutoDS

In the second part of thesis, we focus on integrating domain knowledge into AutoDS frameworks. Users often want to validate models against specific domain requirements, but this process becomes challenging due to the current limitations in incorporating such knowledge.

We explore how to effectively embed domain expertise into AutoDS frameworks, concentrating on two key areas: Model Building and Data Exploration.

In Chapter 6, we address the problem of building a meta-train set of datasets reflecting prior knowledge of the dataset's structure. We call this approach to building a meta-train consolidated learning. Based on the prepared metaMIMIC benchmark, we show that the transfer of hyperparameters can be more beneficial than the previously used non-domain meta-training sets.

Chapter 7 introduces the SeFNet to methodology, which allows structuring tabular datasets based on the systematic meaning of variables found in the data. Thanks to this method, a broader exploration of datasets belonging to the same domain is possible. The resource that has been prepared can be the beginning of a feature-centric perspective in AutoDS.

Chapter 6

Consolidated learning: a domain-specific model-free optimization strategy

This chapter corresponds to the article Katarzyna Woźnica, Mateusz Grzyb, Zuzanna Trafas, and Przemysław Biecek. Consolidated learning: A domain-specific model-free optimization strategy with validation on metaMIMIC benchmarks. *Machine Learning*, 113:4925–4949, 2023. doi: 10.1007/s10994-023-06359-0

AutoDS background in relation to the Hypothesis 4

The field of Model Building is considered a relatively universal subfield of AutoDS, irrespective of the application domain. This universality stems from the fact that the algorithms employed in Model Building are largely similar across various fields. However, the data used in modeling are no longer universal. Data can often be characterized by unique features depending on the specific domains. For instance, in the survival analysis of a disease, these unique features might include related response variables such as long-term and short-term mortality rates. Taking into account the specificity of new datasets has not yet been considered in meta-learning for hyperparameter optimization.

So far, meta-train sets have been constructed for very diverse datasets, neglecting additional relations between predictive problems within the domain. Considering domain-specific meta-set offers two potential advantages. First, it introduces

important new information into meta-learning that can improve hyperparameter transferability. Second, it increases the transparency of the meta-learning stage. This chapter presents the consolidated learning methodology of constructing a meta-train set. This methodology restricts the datasets that are included in the meta-train to better reflect the internal structure of the new dataset. In addition to the methodology, this chapter includes a metaMIMIC benchmark created to validate this scenario of dependencies between datasets.

6.1 Introduction

In order to effectively use the full capabilities of available machine learning algorithms, we have to pay great attention to the hyperparameter values. On the one hand, hyperparameter tuning may be costly due to the dimensionality of the search space. On the other hand, it is necessary because the default settings of hyperparameters do not guarantee good model quality [126, 170]. Therefore, automatic hyperparameter optimization methods are being developed to avoid a manual, trial-and-error-based search for the optimal set and thereby support users in building effective predictive models. They have become a part of AutoML frameworks [208, 14, 157, 62] and resulted in increased attention to the proposed methods' ease of use, implementation, parallelization, and computational complexity. It is essential to adapt to the considered prediction problem and provide anytime performance, i.e., to propose a good configuration of hyperparameters even if only a few evaluations have been performed.

So far, two main groups of hyperparameter optimization techniques have been recommended and are used as baselines in papers proposing new solutions. The most basic class of methods are grid search and random search [12]. They are entirely independent of the dataset; for each case, optimization must be started from scratch for a pre-specified hyperparameter grid. Many optimization evaluations are needed to find near to optimal solutions. In addition, these methods do not use the information obtained in the earlier iterations, namely which algorithm settings resulted in good performance model. The second class, Bayesian Optimization-based methods, addresses that problem. It automatically extracts knowledge from the response surface. Then the surrogate model proposes a new hyperparameter configuration weighing the benefits of exploring new, unseen re-

gions against sampling from the known regions with good performance [96, 13, 197]. This is an example of online hyperparameter optimization adapting to the dataset response function and updating the expected model performance as a function of its hyperparameter values. Nevertheless, these methods still do not provide anytime performance and require independent optimization for every new prediction problem. Population-based evolution strategies [58, 4] or reinforcement learning optimization [130] are other examples of online hyperparameter optimization methods. Both are quite complex, require a different definition of the optimization problem and they are less popular in real-world applications [22].

In addition to the techniques that require performing a complete optimization for each new task, there is an increasing need for an offline approach that involves building a portfolio of several hyperparameter configurations [229, 228, 62, 167, 63, 139]. Furthermore, the hyperparameter portfolio is a particular case of meta-learning since at least one portfolio configuration should parameterize a good-quality model for previously performed experiments and should transfer this good performance to a new dataset. We assume that at least one configuration will be promising for new, unknown data. The data repository, based on which we determine the portfolio is called a meta-train set, and new target prediction problems are called a meta-test.

A predefined, limited set of hyperparameter configurations optimized for a wide range of datasets has been shown to give comparable results to Bayesian optimization [229, 167] and proves to be even better when considering anytime performance. Moreover, the portfolio approach may be seen as an extension of the default hyperparameter values that is easy to share and parallelize. In the first studies introducing this method, all meta-train datasets have the same relevance for the portfolio composition due to their independent weighting. Therefore, to enhance the impact of meta-learning, Feurer et al. [62] use meta-features, (i.e. vectors of dataset characteristics) to evaluate the dataset similarity. Subsequently, during portfolio development, a higher weight is given to a good configuration of hyperparameters from meta-train sets more similar to the new data under consideration. This approach combines the online and offline procedures since a static portfolio leverages the most effective configuration for similar datasets, assuming their optimized functions have similar learning curves. The use of meta-train datasets reduces the time for the early iterations in Bayesian methods.

Techniques employing portfolios built on meta-features and assessing dataset similarity are intuitive to humans and resemble an expert's use of domain knowledge. The difficulty of this approach is the use of meta-features. Firstly, computing meta-features may be expensive and generate errors [63]. Secondly, we do not know how to describe prediction problems and datasets using meta-features in an effective and discriminative way. Namely, whether they should be predefined, based on statistical definitions [217, 179], landmarkers [164], or perhaps automatically trained extractors based on neural networks [54, 90, 111]. Due to its availability, a set of meta-features based on statistical definitions is most often used, but their correlation with model performance is questionable [232]. Likewise, the definition of distance and similarity between datasets is underdetermined [230, 60].

In addition to meta-features, the choice of meta-train is crucial for the effectiveness of the portfolio. The standard choice of dataset repositories is OpenML [19]. It includes prediction problems from diverse domains and may be a satisfying source to build a portfolio that speeds up the optimization for general, random data. Nonetheless, we may have some external knowledge about specific characteristics in many applications, e.g. high target imbalance in insurance claims frequency models or interactions between specific blood tests in medical data. Domain-specific AutoML frameworks [2, 82, 213, 156] already exploit these unique properties. This work shows that instead of searching for meta-features describing these relevant attributes we can appropriately select the meta-train, limiting it to the representative datasets from a specific domain. We call this refined meta-train a *consolidated meta-train* and we denote the subsequent creation of a portfolio to transfer hyperparameter configurations from that meta-train as *consolidated learning*.

Our contributions are as follows. 1) We purposefully restrict the meta-train distribution, taking into account only domain-specific characterizations of considered tasks. Defining a *consolidated meta-train*, we highlight the importance of design decision in the selection of meta-train. 2) We leverage a consolidated collection of the prior experiments to determine the portfolio of hyperparameters transferred from the meta-train to meta-test tasks. We employ two model-free portfolio selection strategy methods: greedy search and average ranking. 3) To mimic a real case and validate *consolidated learning*, we create a metaMIMIC repository extracted from the medical MIMIC-IV database [109]. Our experiments reflect various levels of consolidation between the meta-train and the meta-test (see Figure 6.1) based on the definition of the input and output

space for every task. The metaMIMIC repository is the first benchmark to verify the utility of *consolidated learning*. 4) In our experimental setup, we empirically show an improvement of *consolidated learning* over the baseline methods (random search and Bayesian optimization) as well as predefined portfolios extracted from the OpenML repository. We confirm the hypothesis that *consolidated learning* for MIMIC-IV enhances the transfer of XGBoost [36] hyperparameters in the early stage of optimization. The consolidated portfolio combines the advantages of the two approaches used so far: it extends the idea of the defaults, and it is easy to share such a ranking of subsequent algorithms. What is more, at the same time, we take into account the specifics of a given dataset using the best configurations of hyperparameters for similar data. The proposed method does not require any additional optimization, is parallelizable, and has strong anytime model performance. This property is significant when aiming to achieve good results with a limited time budget. In this way, *consolidated learning* becomes a support for data scientists preparing entire collections of models for similar prediction problems or other subsamples of observations. In the long run, applying *consolidated learning* to model deployment can significantly reduce optimization budgets.

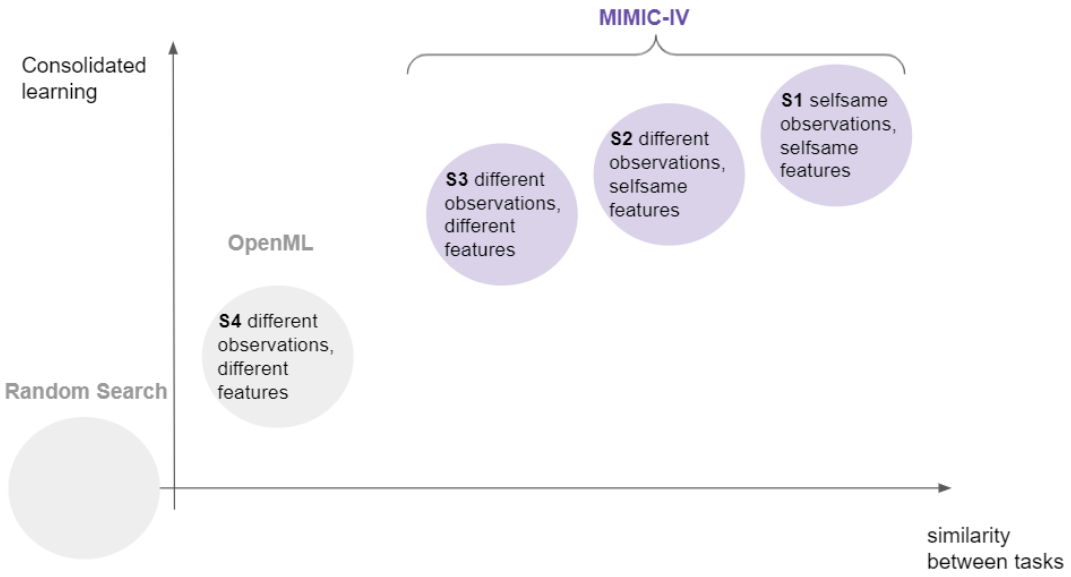

Figure 6.1: Relationship between the similarity of tasks and *consolidated learning*. Correspondence between the space of prediction problems allows the meta-feature-free technique of hyperparameter tuning to incorporate the advantages of meta-learning. The greater the similarity in task design, the more *consolidated learning* increases.

6.2 Related work

Until now, it has been common for individuals to use the defaults implemented in the software or to use simple tuning methods. With the development of machine learning, more advanced hyperparameter optimization methods have been proposed. However, their usage requires additional expertise in the configuration itself, which is why some data scientists find them deterrent and why they often neglect to tune. Random search methods were conducive to automatic hyperparameter optimization gaining in popularity. Previously, it was known that the configurations for many algorithms are crucial for the performance of trained models, but effective tuning of the settings was lacking. Random search facilitates the determination of a low dimensional effective subspace of hyperparameters faster than a grid search or manual tuning, but it is still susceptible to the dimensionality of the search space. To eliminate low-efficiency configurations, faster multi-armed bandit methods such as Successive Halving [104] or Hyperband [131] are used. However, these more advanced methods work only for iterative algorithms and are far less common than simple random search or defaults.

Bayesian-based optimization methods are a class of techniques that particularly require expert knowledge in implementation. Their great advantage is using the knowledge acquired from the previous evaluations and adjusting the optimization process to the characteristics of the considered dataset. However, the selection of a surrogate model is crucial for optimization effectiveness. The most popular are variants of Sequential Model-Based Optimization (SMBO) [112] such as Sequential Model-Based Algorithm Configuration (SMAC) [96], Tree-structured Parzen Estimator (TPE) [13] or Spearmint [197]. However, all of them are computationally demanding, difficult to parallelize and depend on the choice of a starting point [230]. What is more, straightforward Bayesian-based optimization methods do not provide anytime good solutions. To eliminate this problem, adaptive resource allocation and early-stopping of unpromising configurations are combined with Bayesian optimization [59]. Despite these modifications, we still do not leverage the information gathered so far in the previous experiments for other datasets; the only way to provide additional information is the prior distributions of the hyperparameters [155, 200, 163].

In addition to the predefined portfolio of hyperparameters employed in this article, there are different attempts to combine the strengths of online and offline approaches.

The most common is the injection of the portfolio information into Bayesian optimization. The main goal is to exploit the adaptability of online methods while leveraging offline portfolios to quickly propose a good, though perhaps not the best, configuration of hyperparameters. The most common approach is to define the starting points in Bayesian optimization not as random ones but considering their model performance in the prior experiments [60, 62, 231]. These methods emphasize the adaptation of the surrogate model to the new considered dataset, and the portfolio is used only for the initialization. An alternative is to use the results from the previous experiments and build a black-box surrogate model that predicts the performance for a selected dataset and hyperparameter configuration. Then, based on the data collected offline, we can predict the response surface for the new dataset [219, 176, 46, 170].

6.3 Consolidated learning

We define a technique of transferring the hyperparameters from the consolidated design meta-train set to the meta-test task as *consolidated learning*. The motivation behind this modification comes from a practical perspective on building predictive models.

Motivation

When we look at the applications of predictive models in various fields of science and business, it can be seen very quickly that the expectations of machine learning are significantly different from those applied in AutoML. Experts in the relevant fields consider problems with complex structures and ambiguous characteristics, so they often analyse various approaches to building predictive models. That results in multiple analyses not for just one dataset but for a whole collection of related problems by definition. This multiplicity of approaches gives a great deal of potential for enhancing machine learning models using meta-learning for a new prediction problem. We examine the previous research to point out the potential amplification of meta-learning coming from a composition of meta-data.

- **Shared variables.** In real-world use cases, prediction tasks for specific domains often include standard explanatory variables shared between many datasets so that the models can exploit analogous dataset characteristics. In medical research, clinical patient data are often collected according to a set protocol. The introduction

of Electronic Health Records (EHRs) has increased the consistency of reported predictive variables and support for the multi-center research [191, 91, 151, 181, 32].

In addition to the unification of data collection, the reason for the existence of similar datasets is that in the diagnosis of rare diseases, the testing of specific predictive factors is required. Without them, the study is considered incomplete, and any predictive model is not deemed reliable so most examinations use these features. For example, according to [124], 75% of analyzed articles concerning the prediction of Alzheimer's disease dementia progression use cognitive assessments as features. In the treatment of cancer, on the other hand, TNM staging is key - its inclusion in the study is essential for a proper assessment of treatment effects and disease progression [27].

This schema of the similar structure of descriptive variables is also common in experimental sciences such as chemistry, physics, or biology. In biochemistry, a great deal of attention is paid to Quantitative structure-activity relationship (QSAR) studies [115]; based on the encoded structural features of individual molecules, the models are supposed to predict the activity of a given compound in, for example, drug development. In the CheMBL database [68], we can find collections of diverse tasks used in the meta-learning approach in [156]. On the other hand, in high-energy physics, machine learning is used in particle detection from collected data at the Large Hadron Collider (LHC) [30].

- **Related targets.** Many prediction problems do not have a clearly defined endpoint, and several different correlated target definitions are examined. In [196], in addition to the occurrence of acute kidney injury (AKI) within 24h of admission, the occurrence of AKI at certain stages of progression and the risk of death are extracted as the target variable. The same is true for mortality prediction, as endpoints are often considered with differently defined short- and long-term mortality [171, 185]. From a logistical point of view, predicting the length of stay in hospital is also an important issue [171, 211]. If, in addition, the explanatory variables have a similar structure, then the predictive algorithms should capture similar relationships across the tasks.

- **Out-of-time data.** Data scientists working for a specific entity often have one

large database and build multiple models for different data samples extracted, such as [121] building a sequence of machine learning models to predict an acute kidney injury. Another case is the need to update the model for samples from different periods. In that case, updating the set of observations and training the model anew is common. Such models use the same set of variables, so the models should have close properties to the previous versions. The question remains on optimizing the new model, but the retraining of the algorithm with the same hyperparameters turns out to be an effective approach [33].

These dependencies and shared definitions of features between meta-datasets are a different scenario from what has been considered in research papers on meta-learning so far. To capture these circumstances, from the design of the sets used for training, we assume a similarity between the prediction problems. We expect that if machine learning algorithms can use similarly or identically distributed features, then they should detect similar feature interactions or treat the same shared variable similarly. Since the only parameterizations of the model that we know of are hyperparameters we assume that such relationships between prediction tasks will positively affect the transfer of hyperparameters.

Formalization

We formalize the *consolidated meta-train* and *consolidated learning* using the terminology from Section 2.1.3. Restricting meta-train tasks to the representative for specific domain results in dependency between $\mathcal{X}^i_{meta-train}$ and $\mathcal{X}_{meta-test}$ for some $i = 1, \ldots, N$. In particular, explanatory features with the same marginal distribution may occur in two different meta-train and meta-test distributions. In other words, some of the explanatory variables may be shared between the two prediction problems under consideration. Let $P(\mathcal{X}^i_{meta-train}), P(\mathcal{X}_{meta-test})$ be marginal probability distributions for i-th meta-train prediction problem and meta-test, respectively. Then, we denote this situation as $P(\mathcal{X}^i_{meta-train}) \cap P(\mathcal{X}^j_{meta-train}) \neq \emptyset$ for $i \neq j$ or $P(\mathcal{X}^i_{meta-train}) \cap P(X_{meta-test}) \neq \emptyset$. If the features set is identical we denote this by $P(\mathcal{X}^i_{meta-train}) \equiv P(\mathcal{X}_{meta-test})$. We define this constrained meta-train as a consolidated set in which common explanatory variables occur between the sets contained in the meta-train set and the meta-test set. Based on *consolidated meta-training*, a portfolio is composed (according to any strategy) and this

process is called *consolidated learning*.

Correspondence between *consolidated meta-train* and meta-test is significantly higher than between unrelated tasks within the OpenML repository. The assumption about shared variables allows us to propose a meta-feature-free strategy of *consolidated learning*, namely hyperparameter transfer which provides anytime solutions.

6.4 metaMIMIC repository

This section describes the methodology for creating a meta-dataset to imitate the *consolidated learning* environment. Therefore, based on the MIMIC-IV database [109] we create a collection of binary classification tasks of varying similarity. We weigh three scenarios of similarity between the extracted tasks. In the real world, such repositories are naturally collected during model development. However, to our knowledge, such a repository is not publically available for research purposes. Behind the choice of the MIMIC database as a source for the collection of prediction problems is its wide use in the research for machine learning applications in medical diagnosis [154, 238, 147, 136]. We employ this collection as a benchmark for evaluating hyperparameter transfer in *consolidated learning* and assessing the improvement in tuning.

6.4.1 MIMIC-IV database

MIMIC-IV (Medical Information Mart for Intensive Care) is an extensive, freely available database comprising de-identified health-related data from patients admitted to the intensive care unit (ICU) of the Beth Israel Deaconess Medical Center. It contains data of over 380,000 patients admitted to the ICU in the years 2008-2019. We include patient tracking data, demographics, laboratory measurements sourced from patient-derived specimens, and information collected from the clinical information system used during the ICU stay.

To determine the cohort selection, we have to define the patient inclusion criteria taking into account machine learning principles [110, 147, 171]. We consider only the first admission of every patient to preserve the independence of all observations. Every patient must be at least 15 years old at the time of hospitalization and their admission must correspond to at least one chart event, one lab event, and one diagnosis recorded in

the database. The hospital stay length must be shorter than 60 days. In total, 34925 unique patients met all the above conditions.

6.4.2 Prediction tasks

To determine multiple predictive problems, we decided to predict the occurrence of a specific disease. We examined the 50 most commonly appearing conditions and hand-picked groups of diseases that have a representation in both ICD-9 and ICD-10 codes (see Table 6.1). It resulted in 12 targets for binary classification. We also considered whether the selected targets can be successfully predicted with the data available in the MIMIC-IV database (at least 0.7 mean ROC AUC in 4-fold cross-validation after tuning).

Table 6.1: Selected targets with corresponding ICD codes and frequency in the considered cohort.

Condition	ICD-9	ICD-10	Frequency
Hypertensive diseases	401-405	I10-I16	59.8%
Disorders of lipoid metabolism	272	E78	40.3%
Anemia	280-285	D60-D64	35.9%
Ischematic heart disease	410-414	I20-I25	32.8%
Diabetes	249-250	E08-E13	25.3%
Chronic lower respiratory diseases	466, 490-496	J40-J47	19.5%
Heart failure	428	I50	19.4%
Hypotension	458	I95	14.5%
Purpura and other hemorrhagic conditions	287	D69	11.9%
Atrial fibrillation and flutter	427.3	I48	10.5%
Overweight, obesity and other hyperalimentation	278	E65-E68	10.5%
Alcohol dependence	303	F10	7.7%

We selected 58 features, hand-picking ones with the lowest number of missing values. Besides gender and age, we included only numerical variables related to the purposeful medical examination. Most features were recorded several times, so we aggregated them to minimum, average, and maximum values. In total, this resulted in 172 variables. The missing values are imputed with a mean of all observations for each task independently to avoid data leakage.

6.4.3 Task correspondence

In addition to specifying the response variables and the explanatory variable space, we also considered various assumptions for choosing the subset of observations and available

variables. Generally, in applications such choices are forced by the available data, such as the size of the sample of observations that can be used, or how model validation is defined.

We mimic different selection scenarios that affect the intuitive perception of similarity between the obtained tasks in this work. The process of task design always consists of three choices – which predictors to use, which observations to consider, and which target to predict (see Figure 6.2), giving us three scenarios of task correspondence. To verify the impact of similarity between the tasks on the *consolidated learning*, we compare them with baseline transfer from a wide range of OpenML datasets.

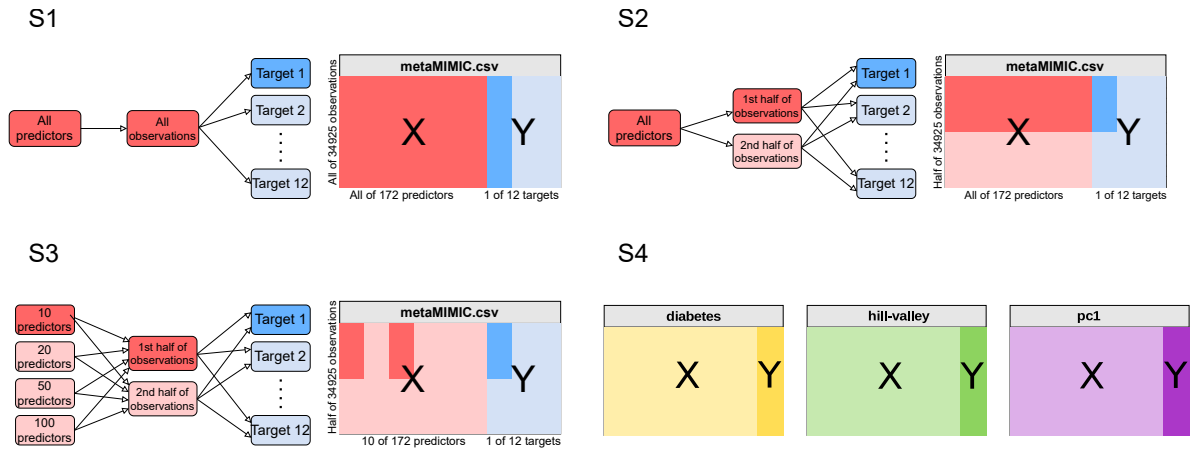

Figure 6.2: Schemas of design decision in creating scenarios of similarity between meta-train and meta-test. In S1-S3 from MIMIC-IV we extracted feature space, a sample of observations, and target disease. In S1 models for meta-train and meta-test use all predictors and observations but targets vary. Scenario S2 contains models built for the same predictors but disjoint sample of observations. In S3 we consider different subsets of predictors. In S4 meta-train is composed of the OpenML datasets unrelated to MIMIC.

1. In the first scenario (S1), we predict different targets considering the same observations and using the same variables. Using formal notation feature space are identical $\mathcal{X}^i_{meta-train} \equiv \mathcal{X}_{meta-test}$ for every $i = 1, \ldots, N$. Therefore, the only real choice to make is to select which target to predict (Figure 6.2 S1). This setup reflects a situation where these targets are determined by historical data of the hospital's patients comprising basic diagnostic tests and diagnoses. The only difference between the tasks is the diagnosis we want to predict, so there are 12 prediction sets.

2. In the second scenario (S2), various targets are predicted considering different sam-

ples observations but using the same set of variables. To avoid leakage of information occurring in S1, we consider two random, disjoint samples of observations (Figure 6.2 S2) but the models are provided with the same 172 variables. This experimental setting corresponds to the situation where we consider models built on out-of-time samples of patients but from the same distribution. When considering any two prediction problems, we can examine models built on independent sets of observations. In this setup, we get $2 \times 12 = 24$ prediction tasks.

3. In the third scenario (S3), we manipulate not only observations samples but also a set of variables to predict defined tasks. We select a different number of the most important predictors for each task (Figure 6.2 S3). The choice of predictor set was realized by selecting top n variables with the highest permutation variable importance value [26, 64], calculated using the XGBoost model with default settings. We determine $n = 10, 20, 50, 100$ out of 172 features. This scenario imitates the transfer of knowledge between models built upon different targets, but now the scenario takes into account not identical feature space. Many prediction problems are based on a core set of variables and these are available in many tasks. An example is blood tests performed and used in the diagnosis of most diseases. So when considering a broad class of medical problems, many of the tasks contain variables describing such measurements. But there are also more specific tests for example cognitive testing is the primary diagnostic of Alzheimer's or ECG for heart diseases. If we are considering sets of models that predict these diseases then the datasets will have corresponding variables. In this setup we get $4 \times 2 \times 12 = 96$ prediction tasks.

4. As a baseline meta-set and meta-learning approach (S4), we use 22 datasets from the OpenML repository. Using this collection for meta-learning, even in an online approach, has proven better than using random search or uninformed Bayesian optimization.

6.5 Experiment methodology

The proposed method of model tuning for the MIMIC-IV database is based on hyperparameter transfer within a collection of medical prediction problems. We validate the effectiveness of using a MIMIC family of prediction problems by comparing analogous tuning

strategies determined for an unrelated family of datasets with OpenML.

We decided on the XGBoost algorithm because it is a very flexible algorithm, and for many prediction problems expressed as tabular data, it achieves the best results by far [195]. On the other hand, XGBoost depends on more than a dozen hyperparameters, both continuous and discrete. These parameters interact with each other, and often their structure is hierarchical. Since XGBoost is widely considered to be a very tunable algorithm [170], we validate the potential of *consolidated learning* for this algorithm.

6.5.1 Hyperparameter grid

As the hyperparameter search space, we use the grid from the MementoML study [122] to validate the *consolidated learning* with the results obtained from 22 machine learning tasks from the OpenML repository. The designed grid comprises 1000 sets of 8 different XGBoost hyperparameters sampled independently from the predefined distributions. The considered hyperparameters and the distributions they are sampled from are presented in Table 6.2. If `gblinear` is selected as a booster, not all hyperparameters are active.

The predefined grid of hyperparameters exemplifies the discretization of the searched space. However, the predefined grid approach has been used in several works on optimization [230, 229] so we decided to create a fixed random grid. It uses the advantages of random search and allows efficient space search while ensuring the reproducibility of results.

Table 6.2: Hyperparameters and their underlying distributions. U stands for a random variable sampled from a uniform distribution with corresponding lower and upper bounds. Booster can be either `gblinear` or `gbtree` with equal probability. With * we indicate the active hyperparameters when booster = `gblinear`.

	Hyperparameter	Type	Lower	Upper	Distribution
*	n_estimators	integer	1	1000	U
*	learning_rate	float	0.031	1	2^U
*	booster	discrete	-	-	{`gblinear`, `gbtree`}
	subsample	float	0.5	1	U
	max_depth	integer	6	15	U
	min_child_weight	float	1	8	2^U
	colsample_bytree	float	0.2	1	U
	colsample_bylevel	float	0.2	1	U

For each task in scenarios S1- S3, we train XGBoost models for a given grid of hyperparameters using 4-fold cross-validation (CV). In scenario S4 we use results from Memen-

toML. ROC AUC is used as the model performance measure. Due to the incomparability of AUC values between the tasks, mean 4-CV ROC AUC is scaled to interval $[0, 1]$ for each task individually.

6.5.2 Tuning strategies

We test four hyperparameter tuning methods for the XBGoost algorithm. Two of them are meta-learning based, so they use the model performance results for other meta-train sets. For each scenario, the following strategies are tested: the transfer is performed with metaMIMIC (S1-S3) and OpenML (S4) independently. In the hyperparameter transfer within the metaMIMIC, we used one-task-out validation. For scenario S4, we studied the transfer of hyperparameters configuration into the optimization for metaMIMIC tasks from scenario S2. As a meta-learning strategy of formation portfolio we considered:

- Average Sequential Model Free Optimization (A-SMFO) [229] using the greedy algorithm to determine a sequence of hyperparameters to test on the new task. The order in the hyperparameter portfolio is initially optimized for the meta-train set and the best configurations for each meta-train dataset are included. In the consecutive iterations, we add configurations among the feasible candidates, not considering the previously chosen ones. This offline algorithm aims to create a diverse configuration portfolio covering a wide range of prediction problems.

- Average Ranks Ranking Method (AR) [25] determining the order of hyperparameters according to the average ranking obtained by configurations for every meta-train dataset. This method is elementary and does not require additional computations.

Both meta-learning methods are limited to hyperparameter configuration derived from the grid defined in Section 6.5.1. As a baseline tuning strategy, we tested random search and Bayesian optimization as two strategies not exploiting results from other datasets.

- Random search (RS) is simulated as a random walk within a defined hyperparameter grid. Due to this, we did not perform a random search several times to estimate the expected learning curve and its variance; we could calculate the theoretically expected model performance after t iterations determined by the expected value of the beta distribution with the relevant parameters and empirical parameters quantiles.

- Bayesian optimization (BO) is performed using the implementation available in the `scikit-optimize` package based on uniform distributions of hyperparameters, with bounds corresponding to the MementoML grid. Since Bayesian optimization may propose different configurations of hyperparameters from our given grid of hyperparameters, it also validates the quality of the proposed search space.

Since A-SMFO, AR, and RS use evaluation from the same hyperparameter grid for every task, the observed maximum of AUC measure is the same for all the three tuning strategies. They only vary in the sequence of proposed configurations. Bayesian optimization is not limited to this grid only and can find better or worse optimal AUC than the other optimizations.

6.5.3 Evaluation of tuning strategies

The objective of the experiment is to see if we can improve hyperparameter tuning for one dataset from metaMIMIC using meta-learning for scenarios S1 - S3 relative to S4. Let us recall that we use a one-task-out schema for scenarios S1-S3 to build the meta-train set. Furthermore, to avoid information leakage for scenarios S2 and S3, we always exclude the target for which we optimize from the meta-learning-based optimization strategy. For example, let us consider scenario S2 and diabetes target variable with the first subsample of observations in meta-train. We include only the tasks for the second subsample of observations but exclude the tasks for diabetes. For scenario S4, the OpenML dataset repository is independent of the metaMIMIC, so we do not address this problem.

We compare the optimization strategies for all scenarios and each dataset individually. For every iteration of optimization of a given strategy, we consider the best performance obtained so far. That is why we are interested in reaching the maximal value as soon as possible and in finding out which strategy achieves this. To aggregate this information for the entire collection of datasets, we recorded the development of the average rank among different hyperparameter tuning strategies. Furthermore, to assess the speed of convergence to the observed optimum, we use the average distance to maximum AUC (ADTM) value [230].

In our experiment setup as meta-test we consider all MIMIC-based tasks available in examined scenario S1-S4. For each meta-test task the portfolio is determined by the corresponding meta-train, according to one-dataset-out validation.

6.6 Effectiveness of consolidated learning

To assess the improvement in transferability of hyperparameters brought by *consolidated learning* we perform optimization for every metaMIMIC task from Scenario S2. We build portfolios upon the meta-train in Scenarios S2 and S4. For S2, we consider the meta-train including the same sample of observations and a disjoint sample of observations as in the meta-test task. We also test two strategies for creating a static portfolio, A-SMFO, and AR. The baseline collates meta-learning results with random search, Bayesian optimization, and the default XGBoost algorithm.

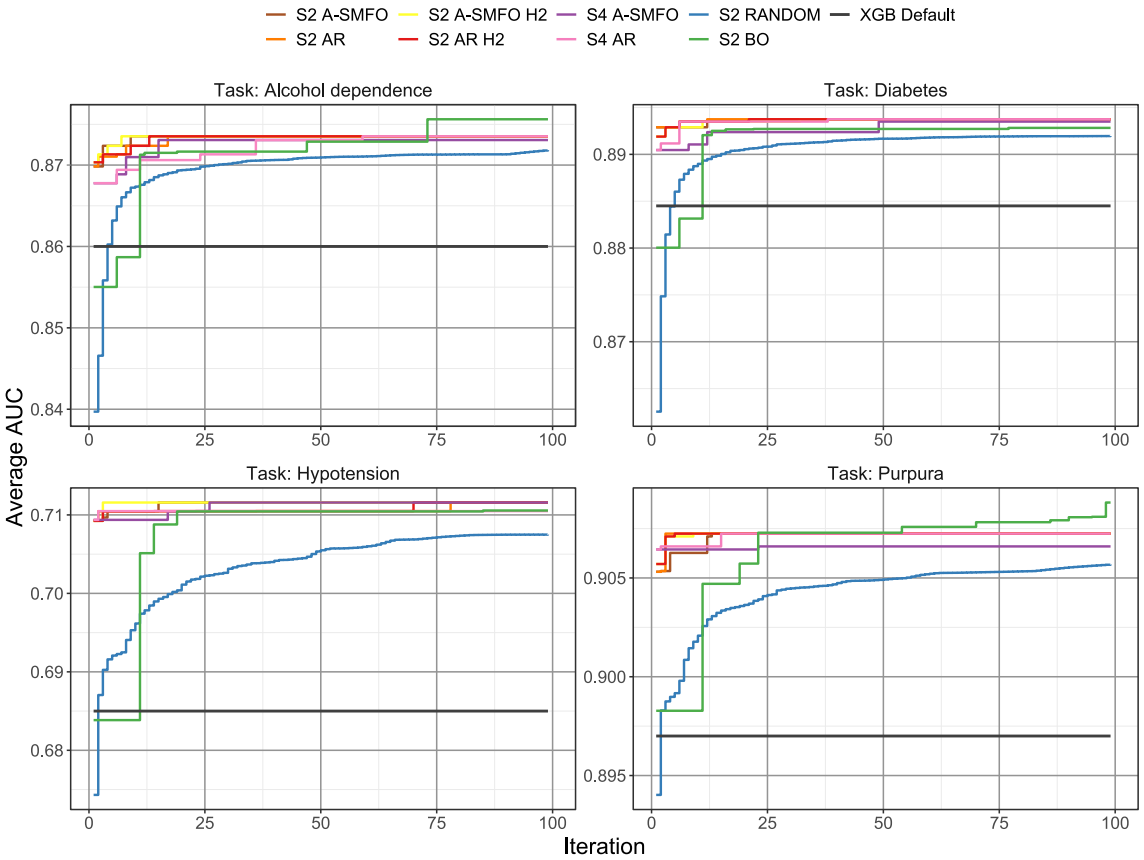

Figure 6.3: Hyperparameter tuning velocity of different methods and multiple tasks. Purpura is the only task for which OpenML initially outperforms MIMIC-IV among the 12 tasks considered.

Figure 6.3 shows the results for four selected meta-test targets. Looking at the model performance during hyperparameter tuning, we see that methods based on meta-learning provide configurations close to the observed maximum already in the first iteration. The distance between the learning curves for scenarios S2, S4, and the baselines is evident for the first few iterations. These results are significantly better than for the model without tuning. Despite being able to go beyond the specified grid, Bayesian optimization

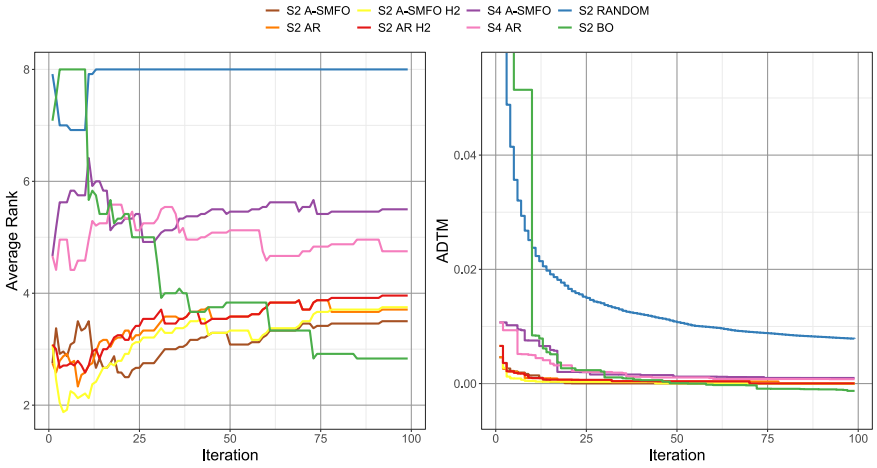

Figure 6.4: Comparison of the aggregated performance development with the increasing number of iterations for different optimization strategies. In the left plot, changes in the average rank of strategy are used to summarize overall efficiency. In the right, ADTM is applied. As meta-test tasks always are used MIMIC-based task from scenario S2. For meta-train, we use tasks from S2 for the disjoint subset of observations (S2 A-SMFO, S2 AR), the same subset of observations (S2 A-SMFO H2, S2 AR H2). As baseline strategies, we use meta-train from scenario S4 (S4 A-SMFO, S4 AR), random search (S2 RANDOM), and Bayesian optimization (S2 BO).

obtains better results for only two targets, so we may assume that the predefined grid covers the space of good configurations.

Let us take a closer look at the meta-learning-based methods. The differences in model performance between the transfer for *consolidated learning* in scenario S2 and the transfer based on OpenML are of the order of 10^{-3}, but in most cases, domain-based strategies reach maximum AUC before OpenML-based methods. Furthermore, the difference between the transfer in scenario S2 regarding the identical and disjoint sample of observations is negligible, so the effect of the subset of observations is not significant. A-SMFO and AR strategies of building a portfolio give close learning curves even in the first iteration.

To further summarize the impact of strategy selection between different tasks and scenarios, we examine the change in average rankings for each strategy (Figure 6.4). We see that in the earlier iterations, portfolios from S2 *consolidated learning* get a better rank than the configurations extracted from OpenML, especially in the early iterations. All strategies based on *consolidated learning* have a similar average rank, regardless of a sub-sample of observations and a method of creating a portfolio algorithm. Only Bayesian optimization exceeds the consolidated optimization but requires about 30 iterations to approach the S2 strategies. In Figure 6.4 we see how fast the tuning strategies con-

verge against the best hyperparameter configuration on average. Similarly, we observe the learning curves for *consolidated learning* converge considerably faster than the other strategies. Again, this marked difference is more substantial in the OpenML meta-data set. As the rank of each strategy changes with time, we see that all lines associated with S2 scenario converge to the observed maximum the fastest. OpenML-based strategies achieve slightly worse AUC but reach the maximum after about 10 iterations. Bayesian optimization goes beyond the fixed-parameter grid, and to see if it finds better hyperparameters than those included in the grid, ADTM for this optimization is computed assuming that the maximum observed value is equal to the maximum observed on the predefined grid. Hence, negative ADTM values for BO over 75 iterations.

Thus, we can conclude that meta-learning is effective and even using unrelated datasets allows us to reject unsatisfactory configurations and provide a decent model performance for several trials in tuning. We can accelerate the tuning by employing *consolidated learning*. Random search and Bayesian optimization need to go through several iterations to achieve comparable results as methods based on a static portfolio created from the previous experiments.

6.6.1 Size of meta-train

To better understand how to build a *consolidated learning* scenario, we investigate the impact of the size of the meta-train set on the hyperparameters' transferability. For each of the 12 tasks from metaMIMIC, we change the size of the meta-train set (between 1 and 11 for S2-metaMIMIC and between 1 and 20 for S4-OpenML) sequentially adding one dataset at a time and then build a ranking of the hyperparameter configurations. For every composition of meta-train we build the hyperparameter portfiolio using A-SMFO strategy. Then we calculate the ADTM after 10 iterations of optimization. Since the order in which sets are added to the meta-train is not fixed, we repeat this operation for 20 different permutations of the order of extension of the meta-train.

In Figure 6.5, we show the distribution of ADTM values depending on the size of the meta-train set and whether it was built on metaMIMIC or OpenML. Here we present the results for 3 selected datasets as a meta-test. For data from S2 we include results from 1 to 11 because this is the maximum size of the available meta-train set; over 11 we only have ADTM values for S4. Boxplots reflect the variability of ADTM values. For the *con-*

solidated learning scenario, we get significantly better results even for a small meta-train set; very quickly, the ADTM values converge to 0. For all meta-test datasets, by using the Wilcoxon test, we can reject the hypothesis that both methods have the same mean ADTM values in favor of the hypothesis that the meta-train based on metaMIMIC has a lower ADTM than the one based on OpenML. For 2-4 sets in the meta-train the model performance in 10 iterations reaches similar values as for the whole 11-element meta-train set. Thus, in this experiment, 4 datasets from the metaMIMIC repository are enough to make the transfer of hyperparameters faster than with selected datasets from OpenML.

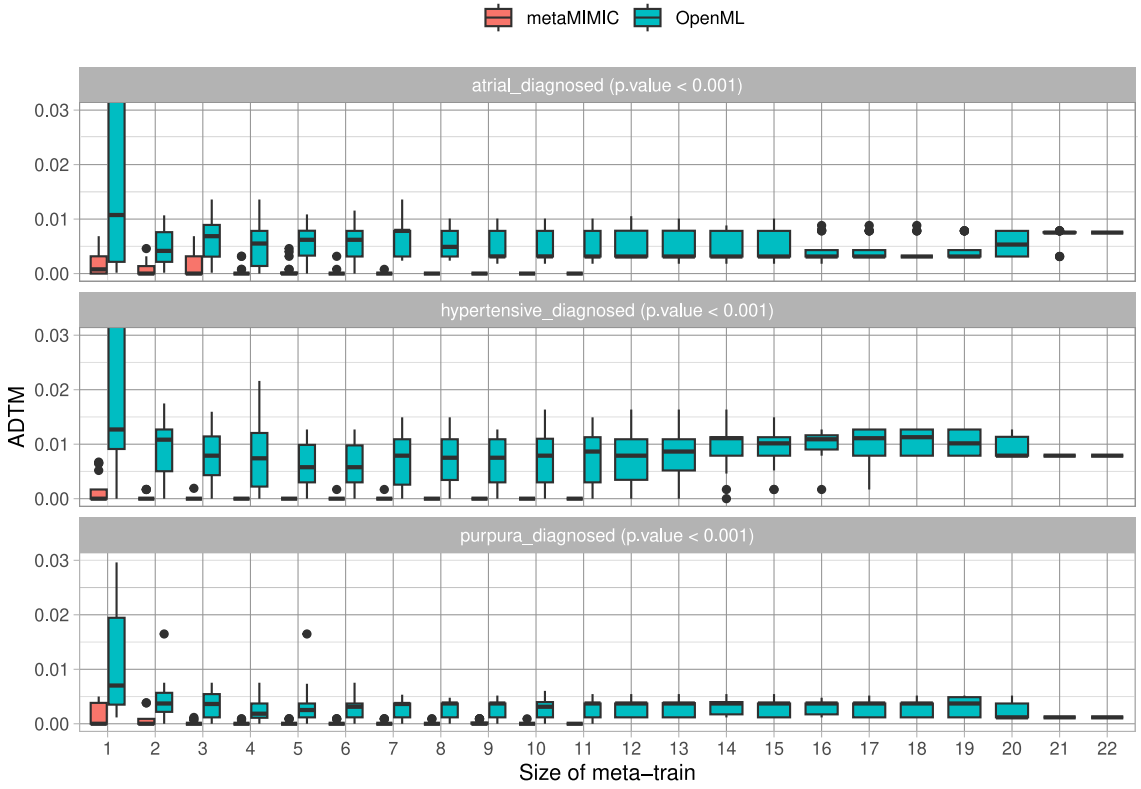

Figure 6.5: Transferability of hyperparameters changes with the number of available datasets in meta-train. For every task we modify the size of meta-train in Scenarios S2 and S4 and check the distribution of the ADTM values after 10 optimization iterations. Grey headings indicate target from metaMIMIC treated as meta-test. In brackets, the p-values of the Wilcoxon test are reported. The alternative hypothesis is that the mean ADTM value for hyperparameter transfer from the metaMIMIC portfolio is less than the mean ADTM value for hyperparameter transfer from the OpenML portfolio for the 4 datasets in the meta-train set. A correction for multiple testing is applied. We obtain similar results for other meta-train cardinalities.

6.7 Robustness of transferability

In Section 6.6, we saw that meta-learning-based methods find the optimum observed on the defined grid after only 10 iterations. In this section, we explore the similarity between hyperparameter spaces in model performance terms. We also investigate the effect of task correspondence on the strength of hyperparameter transfer. This is especially important in order to provide anytime solution for optimization.

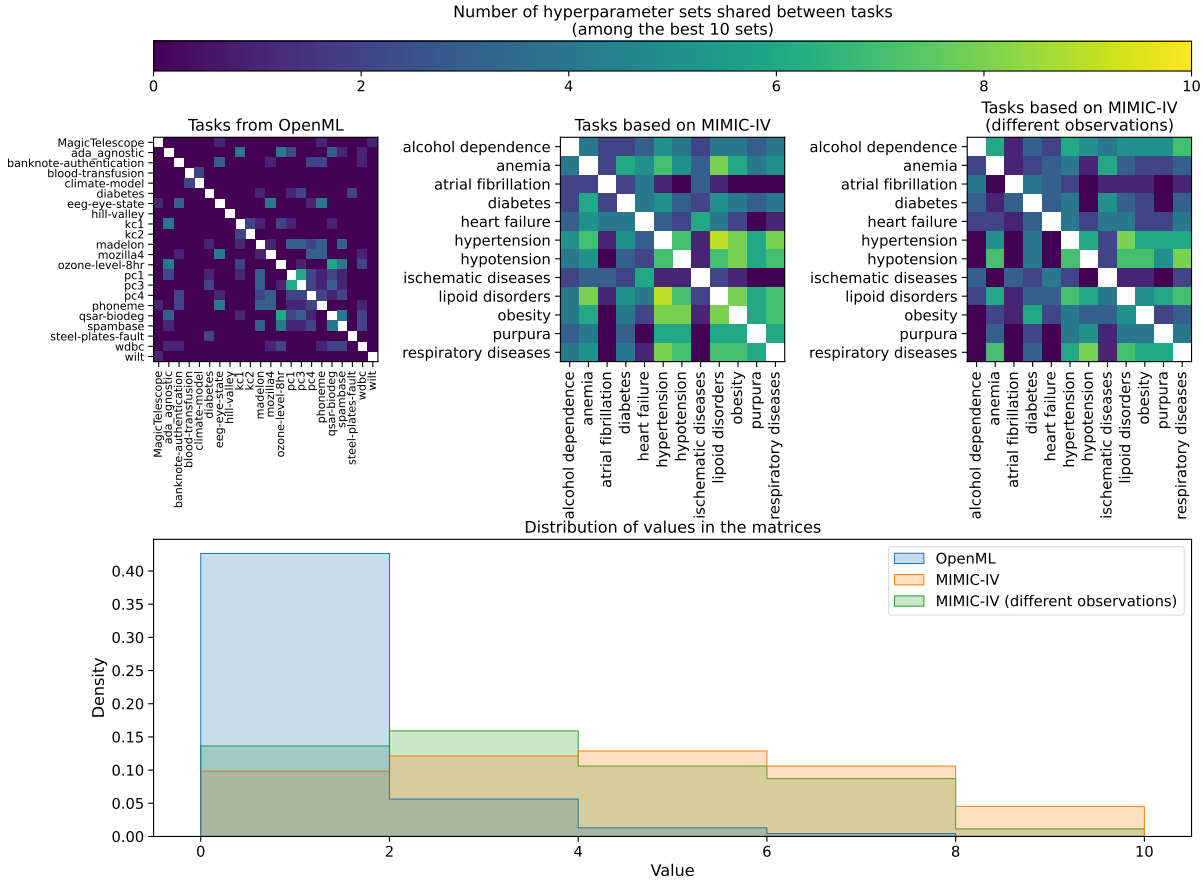

Figure 6.6: Numbers of the best 10 hyperparameter sets (regarding the mean 4-CV ROC AUC measure) shared between tasks from S4, S1, and S2. An individual cell of the matrix corresponds to the number of hyperparameter sets shared between a given pair of tasks. Histograms summarize the distribution of the values of each matrix. White color on the diagonal means that the value is not considered.

To analyze the consistency of the hyperparameter model performance between any pair of tasks, we examine the percentage of overlap in the top 10 best configurations (Figure 6.6). We decide on a threshold of 1% because 10 iterations in tuning is sufficient for strategies S2 and S4. Nevertheless, choosing another threshold value from a reasonable range of 10-100 results in analogous relationships between the distributions of values in the matrices. This fact is also reflected in the mean of Spearman rank correlation

coefficients calculated for individual pairs of full rankings (0.165 ± 0.469 for S4, 0.885 ± 0.072 for S1, and 0.849 ± 0.078 for S2).

Comparing the distributions of values in the presented matrices shows that the number of shared best hyperparameter sets is significantly higher for the S1 than for the S4 scenario representing a meta-learning from unrelated problems. In addition, with the rightmost matrix, it is apparent that considering disjoint subsets of observations (which is often closer to actual use cases) results in only a slight decrease in the average number of shared hyperparameter sets.

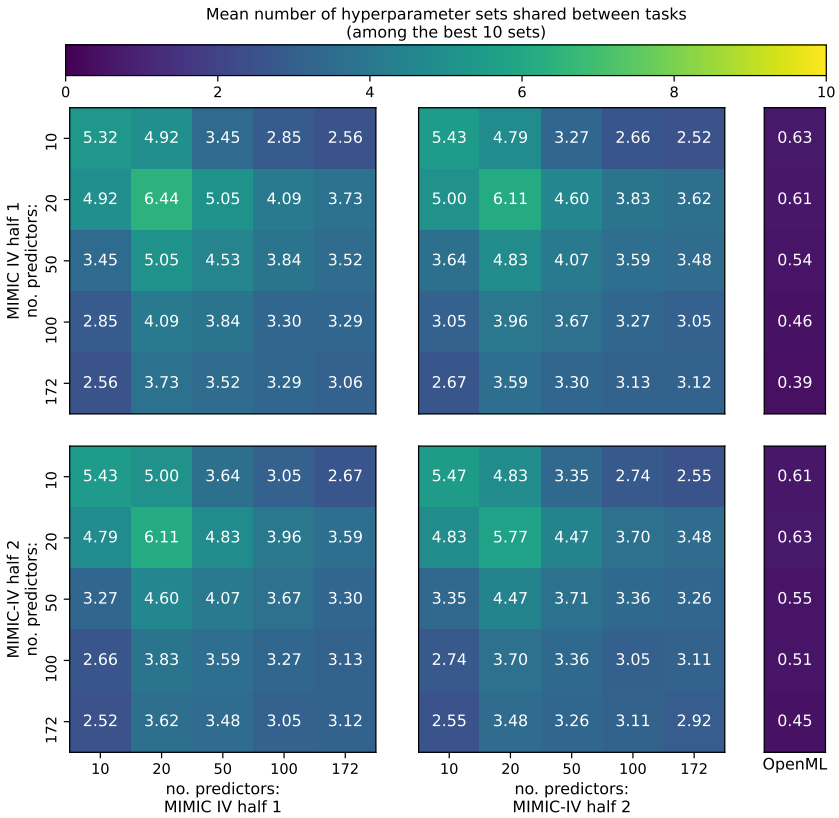

Figure 6.7: Summary of mean numbers of the best ten hyperparameter sets (regarding the mean 4-CV ROC AUC measure) shared between tasks from S3. A single matrix cell represents the average value for tasks with a given number of columns and is based on a given subset of observations. Additionally, the vectors on the right correspond to the same operation for the intersection of MIMIC-IV-based tasks with tasks derived from OpenML.

The analysis of the results scenario S3 required a partial aggregation of the calculated statistics because without this operation, the number of possible combinations would become too large for clear representation in a graph. We decided to perform this aggregation by grouping the tasks based on their source and, for MIMIC-IV, also on the number of predictors and the considered subset of observations (Figure 6.7). Therefore, a single cell corresponds to the average value of a matrix created in the same way as the previous

graph.

As the number of predictors decreases, their diversity between the tasks increases, which is due to the procedure of selecting them described in Section 6.4.3. Despite this, the average number of shared hyperparameter sets for tasks based on a similar number of predictors is consistently high. This suggests that *consolidated learning* is related to the transferability of hyperparameters. Nonetheless, even in the worst case, the average number of shared hyperparameter sets is higher between the pairs of MIMIC-IV-based tasks than when intersecting the MIMIC-IV-based tasks with the tasks derived from the OpenML.

6.8 Conclusions

The results presented in this chapter highlight the importance of selecting meta-train repositories. To our knowledge, this is the first work analyzing the impact of meta-train sets on the optimization power of predefined hyperparameter portfolios. We show that purposefully accumulating results from the prior prediction problems described by similar sets of variables strengthens the optimization strategies. We demonstrate empirically that leveraging datasets from the MIMIC database produces better model performance than using a portfolio determined for a diverse repository. We observe a positive effect of the application of *consolidated learning* both in tuning speed and the consistency of the best hyperparameters. We also analyze the weakening of assumptions in simulated *consolidated learning* - despite smaller constraints in individual *consolidated learning* scenarios, we still show a more significant transfer than for the OpenML datasets.

Using the MIMIC-IV database, we demonstrate how *consolidated meta-train* repositories can be constructed in practice. To our knowledge, this is the first approach to creating a domain-specific repository for meta-learning. Therefore, we consider metaMIMIC as a benchmark for evaluating the quality of the hyperparameters optimization in *consolidated learning* scenario. Creating and sharing a reproducible, unique database corresponding to a *consolidated learning* scenario is a significant resource for future use. It is worth noting that the defined data dependency problem captures the real use of metaMIMIC in both academic work and practical model deployment. What is more, the conducted research uses non-synthetic data from a real-world source but allows us to simulate the different relationships between meta-train and meta-test. Since metaMIMIC

is the first benchmark of its kind, conclusions about, for example, similarities between datasets or the number of datasets in the meta-train necessary for hyperparameter transfer are not universal guidelines but hold valid for this particular experiment.

Our approach enhances the meta-learning effect in hyperparameter optimization while avoiding the problem of defining a representative set of meta-features. This approach attempts to answer whether hyperparameter transfer occurs and whether it can be correlated with some definitions of meta-features. Many papers have posed the question of how to define effective meta-features and whether we can define them a priori [111]. In our study, we were limited to very similar prediction problems in terms of definitions of the described variables, their interactions, and in the context of variable distributions. If there were no transfer of hyperparameters in *consolidated learning* constrained experiment, there would be a serious argument that defining meta-features that affect transfer is impossible. Based on this study, the transfer is more evident within MIMIC-IV-based tasks.

In this study, in the consolidated learning scenario, we focus on the situation where the set of explanatory variables has a non-empty intersection between the sets in the meta-train and the meta-test set. This is possible because the problems were extracted from the same MIMIC-IV database. In real-world applications, meeting these conditions is not trivial. It is possible for a whole series of experiments, e.g., for the QSAR prediction mentioned above, or for unified data from a single source. For other data, determining the similarity of variables and, thus datasets is ambiguous. For data for which we do not have additional domain knowledge, this is a significant limitation of consolidated learning, as it requires inputting information about the semantic meaning of variables. With a non-unified definition of variables, different definitions of the similarity of datasets can be considered. A possible solution to explore these relationships is to use domain ontologies such as SNOMED [223].

In future work, we plan to verify the hypothesis that the consolidated portfolios created for the experiments extracted from MIMIC-IV give better performance for disease prediction problems based on the history collected during hospital admission. This may be the first step leading to domain-specific portfolios for a broader class of problems than defined in this work and transferring hyperparameters between different problems without requiring the datasets to share the same variable definitions partially.

The code needed to reproduce the metaMIMIC and the whole study can be found in this repository: https://github.com/ModelOriented/metaMIMIC.

Chapter 7

SeFNet: Linking Tabular Datasets with Semantic Feature Nets

This chapter corresponds to the article Katarzyna Woźnica, Piotr Wilczyński, and Przemysław Biecek. SeFNet: Linking Tabular Datasets with Semantic Feature Nets. (arXiv:2306.11636), 2023, submitted to Knowledge-based Systems.

AutoDS background in relation to the Hypothesis 5

One of the primary challenges in Data Exploration is the incorporation of domain knowledge. Domain knowledge is essential for understanding the problem context, identifying key variables, and addressing missing information. So far, most developed approaches have been largely universal, primarily due to their greater potential for automation. However, the lack of mechanisms to input additional, domain-specific information into these systems has been identified as a significant barrier to users' broader adoption of AutoDS systems.

It is widely acknowledged that incorporating domain knowledge typically requires the involvement of a machine learning (ML) expert to discern domain-specific nuances. An alternative approach involves utilizing external knowledge sources that systematize domain information, such as ontologies. Incorporating domain knowledge has the potential to provide a more comprehensive perspective on predictive problems, enabling the linkage of current analyses with other experiments within the same domain.

This chapter outlines a methodology for integrating domain knowledge into Data Exploration using ontologies.

7.1 Introduction

Tabular datasets play a significant role in machine learning (ML) applications since they are the most common data type [40]. Their prevalence results in a great diversity of numbers and types of features, understood in this paper as variables encoded in dataset columns. Each dataset can include a different set of variables such as age, gender, income, or education. Thus, considering a broad set of tabular data, we generally refer to a heterogeneous feature space [101]. Because of the features' heterogeneity, most tabular datasets remain unrelated, lacking established relationships to assess their similarity in meaning. This lack of structured semantic information about datasets is a relevant constraint in the development of methods pursuing the paradigm of going beyond one dataset and leveraging other information available in meta-learning [217] and informed machine learning [220]. For example, in Alzheimer's diagnosis, 75% of the works use cognitive tests [124], but their varied description makes it difficult to use transferable ML methods. An analogous situation also happens in other multi-center studies where some tests can be used interchangeably, e.g. creatinine and myoglobin in acute kidney injury diagnosis [226].

To fill this gap, this research article proposes **Semantic Feature Net (SeFNet) methodology** to set up semantic relations between disparate datasets. In SeFNet, variables create the net of relations based on semantic information extracted from ontology. This resource of related features and datasets hold significant potential in incorporating additional information to machine learning pipelines. As of today, it can assist machine learning specialists in collaborating with domain experts, facilitate the exploration of similar experiments, and leverage prior insights about various stages of the data analysis process, such as feature selection, data imputation, or model optimization. It is the missing link to support the exploration of semantically similar experiments and can be an important resource for methods that fit into the broad field of Automated Data Science [47]. Beyond establishing links between features, the integration of ontologies into SeFNet offers an additional external prior information from the perspective of every single dataset.

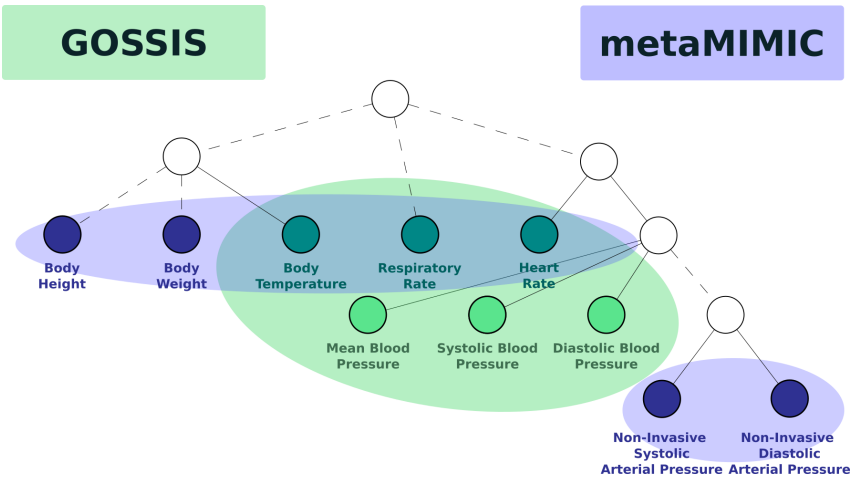

Figure 7.1: An example of subset of SeFNet for the two tabular datasets: *GOSSIS* [173] and *metaMIMIC* [233]. This resource encodes the structure of the relations between their features. Features are mapped on terms from the SNOMED-CT ontology [223]. The diagram shows the tree structure of these features relations. The blue nodes highlight features found in the *metaMIMIC* dataset, and the green ones highlight in the *GOSSIS* dataset. White nodes denote the common ancestors of the presented concepts. A solid line means that the lower node is a direct child of the upper node, and a dashed line means that there are other nodes on the path between them.

This is a significant resource for informed machine learning approaches. In the long term, SeFNet can serve as a foundation to enhance meta-learning methods in Automated Machine Learning [98, 88] that automatically extract information from machine learning experiments [60, 229, 98]. In particular, it can also be an important contribution to the development of methods targeting the heterogeneous feature space [101, 111, 102, 240].

SeFNet is primarily designed for domain-centric applications as it evaluates the semantic similarity of variables given the domain knowledge encoded in ontologies. One notable advantage of the proposed framework is its versatility, as it can be applied in any domain with existing ontology. To demonstrate this methodology, we prepared a prototype for the healthcare domain. That field is an ideal starting point for applying SeFNet. Firstly, medicine requires a holistic perspective when addressing prediction problems, considering the complex interplay between various variables. Additionally, medical datasets often exhibit unique characteristics, such as small sample sizes resulting from rare diseases, emphasizing the need for innovative approaches to extracting meaningful insights [2]. Moreover, the medical domain offers an extensive collection of datasets [74, 109, 204, 53], providing a rich source of information for meta-learning analyses. Finally, medical ontologies and knowledge bases can be further developed more actively due to the new applications provided by SeFNet.

Contributions. In this work, (1) **we introduce a Semantic Feature Net (SeFNet) methodology** that enables the semantic structuring of features found in tabular datasets. Building and applying SeFNet could become a relevant practice in data collection and curation since it can enable sharing of information about features across diverse tasks and potentially improve meta-learning methods. The SeFNet methodology is highly versatile and can be applied to any ontology, making it widely applicable across various domains. (2) **We create and share a comprehensive repository** focused specifically on healthcare datasets used in machine learning. The features within these datasets have been carefully structured and published in SeFNet, resulting in a network of 216 distinct features derived from 16 different datasets. This repository can serve as a valuable resource for researchers and practitioners working in the healthcare field. (3) **We propose a method for quantifying the semantic similarity between any pair of datasets** by utilizing the similarity of features and employing the Dataset Ontology-based Semantic Similarity (DOSS) aggregation. The DOSS representation is a novel approach that can incorporate semantic meaning into meta-learning methods. (4) Finally, **we show benefits for Automated Data Science** resulted from implementation of the SeFNet methodology into machine learning process. Including domain knowledge is necessary to enhance collaboration between domain experts and data scientists, while SeFNet's methodology allows for further development of automation.

7.2 Related work

7.2.1 Collections of datasets

Machine learning researchers exploit open repositories containing a wide spectrum of tasks to properly evaluate their novel methods or algorithms. Consequently, these repositories contain extensive collections of tabular datasets, each representing a separate source of information. Various methods have been introduced to establish links between these datasets, but they lack specificity with regard to domain characteristics. Medical datasets that could be useful in domain-centric approaches are becoming more abundant, yet these datasets remain standalone.

Domain-agnostic collection of tabular data. One of the first repositories of datasets used in benchmarks for tabular data is the UCI repository [52]. Its origins

date back to 1982, and this repository became the foundation for the next one, currently the most widely used repository OpenML [218]. Since any user can upload their dataset there, OpenML is a diverse and very broad collection of datasets. Thus, it is a good reflection of the diversity of data in various fields, but on the other hand, the data is of very different quality. For this reason, benchmarks using a fixed subset of tasks have been created – OpenML100 and OpenML-CC18 [19]. A separate benchmark has also been determined for AutoML research [71]. A parallel initiative was the creation of the Kaggle platform[1], where anyone can create a challenge with a specific prediction problem. Its primary purpose is to allow different teams to compete in preparation of the most accurate solution to a problem, but in addition, Kaggle also became a vast data repository. Another platform aiming at improving collaboration is the Hugging Face [2]. Using this, researchers can share pre-trained models and datasets for a wide range of problems in natural language processing. Some of the uploaded tasks are available as tabular datasets. Also at conferences, such as NeurIPS with Track Dataset and Benchmarks, more and more attention is being paid to expanding dataset repositories and using them to validate new or existing methods. Despite all these efforts to collect more data, many ML researchers use the mentioned repositories to obtain data for benchmarking new machine learning methods, but they are treated mostly as standalone tasks, and this may result in a loss of relevant information.

Similarity between pairs of tabular datasets. One way to find the relations between diverse datasets is through the use of meta-features [217]. Meta-features account for information about the tabular datasets' statistical characteristics. So far, most papers have summarised the distribution of each feature in the dataset. The basic meta-features are based on statistical definitions (such as kurtosis or skewness) [179], the performance of simple machine learning models or features extracted from them [164]. In addition to the prior defined meta-features, automatic feature extraction models [54, 111] are developed to extract noninterpretable meta-features. However, all available representations of tabular datasets focus on reflecting the structure ignoring potential of the specificity of the domain and the semantic meaning of the features [233].

Medical (domain) databases. In medicine, specialization and specificity of datasets are very important and researchers pay much attention to the high quality of data.

[1] https://www.kaggle.com/
[2] https://huggingface.co/

The richness of the available medical domain datasets is very well demonstrated by the PhysioNet platform [74], which was created so that potentially sensitive medical data could be shared responsibly. This allows large datasets such as MIMIC [109], HiRID [100] or GOSSIS [173] to be available for the community. What is more, many country is developing BioBanks, potential sources for the vast amount of data used in machine learning. One of the most popular in the machine learning community is the UK BioBank [204]. Moreover, medical specialities also provide similar data, e.g., UNOS in transplantology or SEER [53] in oncology. Very broad collection of diverse datasets is published also journals such as Scientific Data. Considering the medical category, datasets are divided there into ten different subcategories such as anatomy, healthcare or pathology. Datasets are also collected, as it were, in the course of software development, and interesting datasets can also be found in packages published in Bioconductor [3] where contributors should follow guidelines from ExperimentHub [153]. Despite the great similarity in the information contained in the various data, most of the data is prepared with one specific use in mind rather than integrating it with other sources. Thus,the standards for integrating data from various experiments leave much to be desired.

Data management. Because of the large volume of such diverse databases, there is a real need to create guidelines for proper dataset facilitating. The most known are FAIR principles [227] according to that data should be findable, accessible, interoperable and reusable. These principles are widely recognized as essential prerequisites for data management but also for successful data mining and machine learning applications [188]. Because of this subsequent standards are based on FAIR principles . For instance CARE principles, extending FAIR's, postulate collecitve benefit, authority to control, responsibilty and ethics [31]. CARE principles place additional emphasis on the ethical and societal dimensions of data handling and impose this responsibility on the dataset owners or authors. In the CTSA's National Center for Data to Health (CD2H)[4], guidelines are being developed for the creation, normalization and sharing of meta-data to support reusability. Yet there is still little attention paid to giving semantic meaning to individual features and leveraging this knowledge in meta-learning.

[3] https://www.bioconductor.org/
[4] https://cd2h.org/

7.2.2 Reaching beyond one dataset

Extending machine learning beyond a single dataset involves integrating information from various experiments or incorporating domain knowledge, which falls within the realms of meta-learning and informed machine learning, respectively.

Meta-learning involves extracting insights from multiple experiments to improve the efficiency of creating new machine learning models. Instead of treating each experiment in isolation, meta-learning techniques seek to generalize across experiments, leveraging similarities and differences to inform model design, parameter selection, and optimization strategies. By synthesizing knowledge from diverse experiments, meta-learning facilitates the creation of more robust and adaptable machine learning systems.

Establishing relationships between different datasets is crucial for meta-analysis [148]. Particularly with heterogeneous data originating from different domains, this is often achieved through predefined statistical meta-features described in Section 7.2.1. Despite their simplicity, these meta-features have proven to be effective in transferring hyperparameters. For instance, Feurer et al. [60] introduce a technique to speed up Sequential Model-based Bayesian Optimization (SMBO) by using knowledge from previous optimization runs, while Kim et al. [117] use meta-features to warm-start Bayesian hyperparameter optimization. Feurer et al. [61] further enhanced this approach by developing a scalable meta-learning model for Bayesian optimization, which significantly reduced optimization time.

However, when dealing with datasets related to specific domains, an additional layer of semantic meta-features can be introduced to enrich the relations between datasets. They can be complementary to a set of existing meta-features encoding the mathematical aspects of datasets. What is more, this approach can facilitate informed decision-making and the development of domain-specific machine learning models [47].

Informed machine learning. According to the paradigm of informed machine learning [220], external information is especially important if data-driven methods that rely only on the information contained in a given dataset fail – in terms of model performance or constraints which model needs to meet. Some algorithms employ universal methods that incorporate graph-based knowledge directly into their architecture as a core element [132, 69]. Alternative approaches prioritize generating representations of instances while considering domain-specific knowledge encoded within graph struc-

tures [39, 138, 237]. However, these approaches are not model agnostic – they create representations dedicated to specific input data associated with the problem definition. Incorporating the structure of dependencies between features into feature selection mechanism is more universal [165].

Aside from injecting ontologies directly into the models, Beckh et al. [11] stress the significance of integrating knowledge into explainability methods. Systems like DoctorXAI and FairLens [158, 159] utilize a semantic similarity score derived from the representation of ICD codes for diagnoses, followed by the selection of nearest neighbors based on visit-to-visit distances. Confalonieri and Besold [42] extends the existing algorithm Trepan [44], which extracts surrogate decision trees from black-box models, by incorporating ontologies that model domain knowledge.

7.2.3 Ontologies as bridge between datasets

Domain datasets often describe very similar concepts, and to fully comprehend differences, we need to understand nuances between them. Often, various concepts are described at different levels of generality. In one dataset, we have a more precise definition of a variable, while in another, we don't have full, detailed information and have to use a a broader term. An example is "blood pressure" and "non-invasive systolic blood pressure". Encoding this hierarchy of semantic meaning of individual features requires using an appropriate knowledge base – an ontology.

In simple words, an ontology may be understood as a graph of terms represented as vertices and edges defining relations between them [80]. Usually, ontologies are directed graphs in which a hierarchy of concepts can be specified; from more general concepts to specific ones. Thus, ontology enables the storage of domain-specific vocabulary and provides means to describe phenomena within a particular domain in digital format.

Every domain has its own knowledge bases, with terms and relations reflecting the specificity of the domain. In the medical domain, there are many widely used of ontologies such as Gene Ontology (GO) [8, 70], Human Phenotype Ontology (HPO) [119] or Systematized Nomenclature of Medicine – Clinical Terms (SNOMED-CT) [223]. Ontologies often encode thousands of different terms, so the annotated set of specific concepts forms a sparse subset of all terms. The semantic similarity between terms is defined to assess the semantic proximity of terms considering primarily taxonomic relationships [85].

Semantic similarity allows us to grasp the proximity between concepts by looking at the graph structure of the ontology and the information content. Because this may be differently defined, it is challenging to provide an unambiguous definition of semantic similarity as a formal measure [190, 144, 84, 20]. It is usually assumed that a measure is better the more similar its output would be to the experts' assessment of similarity [190, 84]. Pedersen et al. [160] proposed a benchmark that can be used to evaluate the correlation between the similarities returned by measures and those determined by domain experts. The choice of semantic similarity depends on the application and ontology.

Measures of semantic similarity are successfully used in disciplines such as natural language processing [107], geoinformatics [10] or even neuroscience [118]. Among other fields, wide applications have been found for it in the biomedical domain, where it has been used to compare biological entities by meanings [236, 77]. In this work, we apply this meaning to structure and represent semantic nets of features and datasets.

7.3 Semantic Feature Net (SeFNet)

In this section, we present Semantic Feature Net – a methodology introducing semantic knowledge between tabular datasets. Datasets organized according to SeFNet serves as a system that structures a collection of features originating from the considered domain and used in the machine learning process. This is a universal approach, applicable to many domains [92, 38, 37].

To define SeFNet and adapt it to a specific application, we need to specify three essential components. The first is **a set of tabular datasets** from a selected domain. These datasets serve as the basis for extracting and structuring features. The second is **an ontology** that covers the relevant concepts from the considered domain and the datasets. The choice of ontology is the responsibility of domain experts. The last one is **a semantic similarity measure** consistent with the selected ontology.

After defining the key components, the first step of building resource with SeFNet methodology is feature annotation, which produces the mapping of features found in datasets to terms in the selected ontology. This process can be done manually, preferably with the support of a domain expert, but in the future, it is possible to automate this process. In the SeFNet methodology, we assume that the same ontology is used for annotations of each dataset. In this step, the features gain representation in the domain

Table 7.1: A summary of the annotated datasets and their origins. Each was assigned to one of two categories. We also provide the number of unique variables in each dataset (No.Feat.) and the number of annotated features with terms from SNOMED-CT.

ID	Dataset	Origin	Cat.	No.Feat.	No.Ann.
1	Cardiovascular Study	[Kaggle]	Survey	16	15
2	Diagnosis of COVID-19 (Subset)	[Kaggle]	EHR	19	18
3	Diabetes Health Indicators	[Kaggle]	Survey	22	21
4	Diabetes 130 US	[UCI,OpenML,Kaggle]	EHR	49	38
5	GOSSIS-1-eICU Model Ready	[PhysioNet]	EHR	68	60
6	Stroke Prediction	[Kaggle]	Survey	11	11
7	Heart Disease Indicators	[Kaggle]	Survey	22	21
8	Heart Disease (Comprehensive)	[OpenML]	EHR	12	11
9	HCV data	[UCI,OpenML,Kaggle]	EHR	13	13
10	Hepatitis	[UCI,Kaggle]	EHR	20	19
11	HiRID Preprocessed	[PhysioNet]	EHR	18	17
12	Pima Indians Diabetes	[OpenML,Kaggle]	EHR	9	8
13	ILPD	[UCI,OpenML,Kaggle]	EHR	11	11
14	Breast Cancer	[UCI,OpenML]	EHR	10	9
15	metaMIMIC	[Paper]	EHR	184	175
16	Thyroid Disease	[UCI,OpenML,Kaggle]	EHR	30	27

knowledge graph.

It is worth noting, that SeFNet as the network of semantic links between features and thus between datasets is the next level in assessing similarity between sets of tables. Because of the complex structure of features – in addition to semantic meaning, each feature is a random variable with a specific distribution – it is necessary to consider multiple definitions of similarity between features. To date, the focus has been mainly on statistical descriptions of variables aiming to capture the characteristics of univariate and multivariate random variables. SeFNet adds another layer of systematization of datasets.

Using SeFNet methodology we build a high-level system dependent on the selections made for datasets and ontologies. We present its components using healthcare datasets as an example. We refer to this created repository in the following sections as SeFNet-Healthcare.

7.3.1 Datasets in SeFNet-Healthcare

To set up the prototype of dataset collection built according to SeFNet methodology for the healthcare domain, we need to specify the selected datasets. In Section 7.2.1, we present a diversity of medical datasets, but due to limited resources, we have to limit the scope of the search. Our goal is to systematize the medical datasets used in machine

learning. We target data describing individual cases, enabling the identification of risk factors in medical research. For this reason, we focus mainly on two types of data sources:

1. Kaggle, OpenML, and UCI repository, where medical datasets from not always well-verified sources are available for immediate download. These reflect the resource that an average machine learning researcher works with.

2. PhysioNet platform where high-volume medical datasets are published and available for credentialed usage. These datasets are commonly used in multi-center retrospective studies. For this data, preprocessing is necessary to transform the data into a single plain table. We use preprocessing either prepared by database authors or in other research projects.

We have collected 16 datasets (see Table 7.1), which can be divided into two groups due to tha method of data collection and specificity of features in each dataset: (1) datasets based on survey data (Survey), (2) a group of datasets where predominate electronic health records (EHR).

The resources collected within SeFNet-Healthcare can be expanded, however, we aimed to include representative examples of datasets from various sources. The overview of the presented datasets can be easily accessed on the provided website `https://sefnet.mi2.ai/`.

7.3.2 SNOMED-CT Ontology

We employ the SNOMED-CT ontology to describe medical and demographic concepts encoded in variables. SNOMED-CT is not the only valid choice of ontology, but it is supported by its universality and the numerous works that map other vocabularies onto this ontology [49, 207, 116]. SNOMED-CT ontology is being actively developed and contains more than 350 thousand terms relating to anatomy and demographic data often attached to patient descriptions [198]. The SNOMED-CT terminology is becoming the standard in various countries, such as Canada, France, the USA, and the UK.

The annotation process is demanding. Because of the brief and often inconsistent feature names in the considered data, it is challenging to automate that process without additional resources, so it is done manually at the moment. Annotations are based on feature names but also on descriptions emerging from dictionaries provided with the dataset.

The latter is especially important for PhysioNet data since internal system codes are often used to describe the variables. We establish consistent criteria for annotating ambiguous terms. The SeFNet-Healthcare repository is the first of its kind to be systematized and made available. The entire resource of annotations is also available on the repository[2] and can be easily explored via SeFNet website[1].

SNOMED-CT ontology is well suited to the task of annotating medical data used in machine learning; 216 different features are included in the selected datasets, and up to 92% of them are annotated. Looking at each dataset separately, we also observe a high percentage of variable coverage with terms from the ontology (see Table 7.1). The *Diabetes 130 US* dataset has the lowest percentage of annotated variables. However, the variables that have no equivalent in SNOMED-CT terms were administrative in nature, such as `admission_type_id, discharge_disposition_id`. It is worth noting that for two datasets it was possible to annotate 100% of the variables. On the one hand, all of the data concern medical problems, but on the other hand, they touch on different specificities of diseases, e.g., diabetes or Covid-19. Nevertheless, a significant number of terms occur in more than one dataset. In Table 7.2, we present the most frequently recurring terms and examples of original variable names found in the data. Patient age and gender are the most common, but disease information is also prevalent.

Table 7.2: The most common terms in all datasets, along with examples of how the variables describing these terms were originally named.

SNOMED-CT term ID	No. datasets	No. unique	Names used in datasets
397669002	15	3	*Patient age quantile, age, Age*
263495000	11	4	*Gender, sex, Sex*
73211009	6	5	*diabetes, diabetes_diagnosed, Outcome*
359986008	5	8	*Total_Bilirubin, bilirubin, BIL*
38341003	5	4	*HighBP, prevalentHyp, hypertensive_diagnosed*

7.3.3 Semantic similarity of terms

Annotation of variables allows for the analysis of datasets regarding jointly occurring variables encoded as terms. However, the biggest advantage of using ontologies over dictionaries is using relations between concepts. As mentioned in Section 7.2.3, there is

[2] https://github.com/MI2DataLab/SeFNet-Healthcare
[1] https://sefnet.mi2.ai/

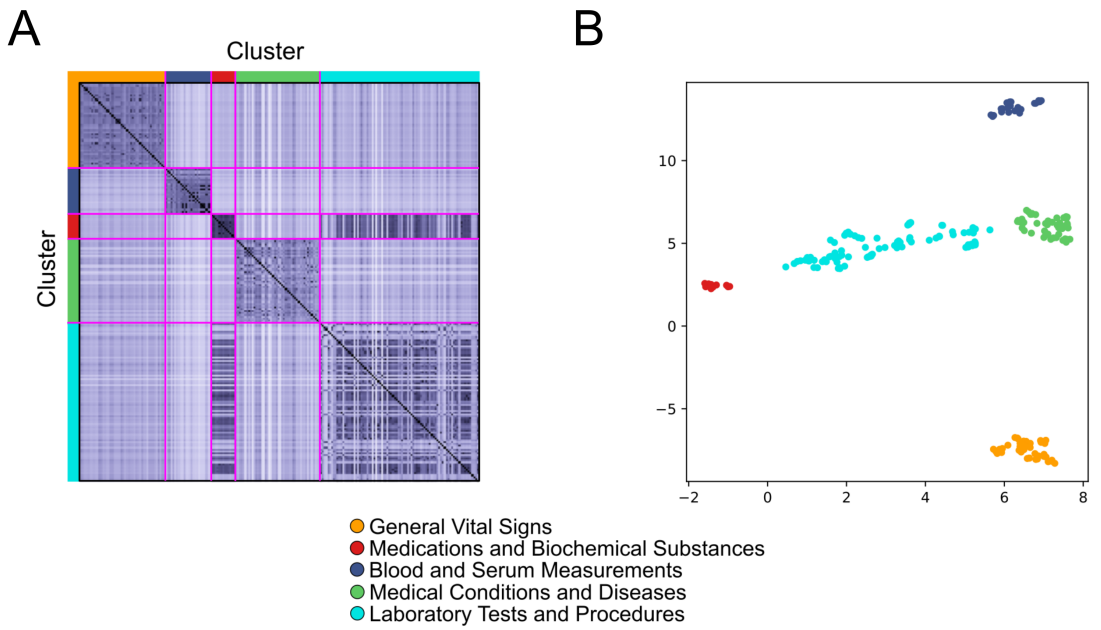

Figure 7.2: Similarity between the annotated features occurring in SeFNet-Healthcare. In panel A, we show the similarity matrix between each pair of features. The features' order corresponds to the clusters' belonging, illustrated in panel B. We can distinguish five groups of features named based on the high-level concepts.

no one universal measure of similarity, but they are application-dependent. In the case of SNOMED-CT, Harispe et al. [84] conducted a benchmark on 29 terms derived from that ontology and found that the most concurrent measure with expert intuition is Tversky's abstract Ratio Model measure [212] with specific values of parameters.

The measure is formulated in the sense of common and distinctive information contained in terms. Let $A(t)$ be a set consisting of a term t and its ancestors in a given ontology, and $\Theta(t) = |A(t)|$ be the cardinality of this set. Common information between terms t_1, t_2 may be encoded as $A(t_1) \cap A(t_2)$, and we assume that this information is measured as $\Psi(t_1, t_2) = |A(t_1) \cap A(t_2)|$. So we use similarity measure (SM_{RM}) between terms t_1, t_2 defined as follows:

$$SM_{RM}(t_1, t_2) = \frac{\Theta(t_1)}{(1+\alpha) \cdot \Theta(t_1) + \beta \cdot \Theta(t_2) - (\alpha + \beta) \cdot \Psi(t_2, t_1)} \tag{7.1}$$

Constants α, β are determined in [84] so we assume that $\alpha = 7.9$ and $\beta = 3.9$. This measure is symmetric, and values of $(1 - SM_{RM})$ can be considered the distance between terms. Whenever we refer to semantic similarity, we calculate it using Equation 7.1, as it seems to be fairly effective in determining the proximity of terms in the SNOMED-CT ontology.

In Figure 7.2A, we show the values of SM_{RM} for each pair of terms occurring within SeFNet-Healthcare. Figure 7.2B visualizes projected vectors of similarity using UMAP [145] technique. To define the groups of features, we use HDBSCAN [29] and then name them based on high-level concepts in the ontology. The determined clusters also form a distinct box structure on the similarity matrix in panel A.

7.4 Dataset Ontology-based Semantic Similarity (DOSS)

In this section, we introduce a Dataset Ontology-based Semantic Similarity (DOSS) measure that aggregates the similarity between two sets of terms in particular between sets of variables contained in two datasets.

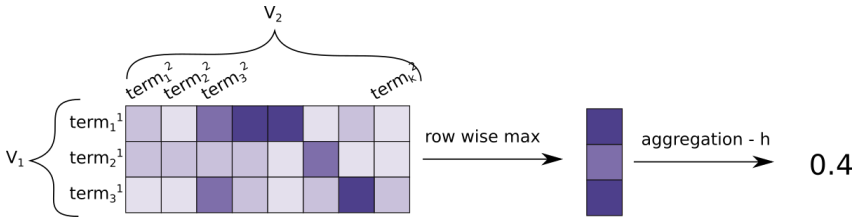

Figure 7.3: An illustrative scheme for calculating DOSS between D_1 and D_2. Sets V_1 and V_2 contains features mapped to $\{term_1^1, \ldots, term_m^1\}$ and $\{term_1^2, \ldots, term_k^2\}$ respectively. Function h is an aggregating function. In individual cells, a higher saturation of purple indicates a higher SM_{RM} value between terms.

Let V_1 and V_2 be two sets of terms included in datasets D_1, D_2 respectively. Let each set consist of a set of terms $V_1 = \{t_1^1, \ldots, t_m^1\}$, $V_2 = \{t_1^2, \ldots, t_n^2\}$. Then we define DOSS as

$$DOSS(D_1|D_2) = h(\{\max_{t_k^2 \in V_2} SM_{RM}(t_i^1, t_k^2) | i \in \{1, \ldots, m\}\}), \tag{7.2}$$

where $h : [0,1]^n \to [0,1]$ is discretionary summarizing function.

An overview of the DOSS calculation scheme is shown in Figure 7.3. For each pair of terms derived from V_1 and V_2, the SM_{RM} measure is determined, and then the maximum SM_{RM} value is determined for each term from the reference dataset. The question remains how to aggregate results across features to guarantee the invariance of size regardless of dataset dimension. The mean function is the most intuitive choice of h since it determines the average similarity of V_1 terms relative to V_2.

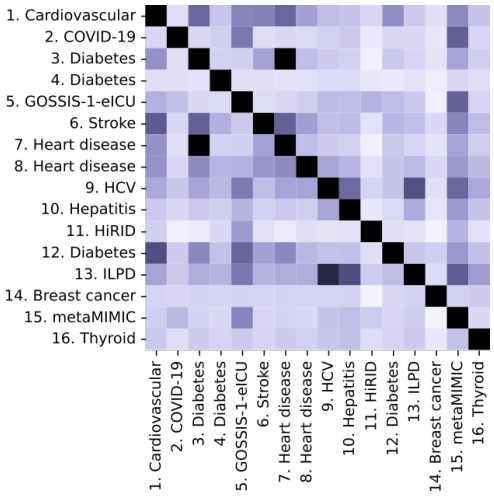

Figure 7.4: Matrix of DOSS values between any two annotated datasets. The matrix is not symmetrical since the DOSS measure does not have this property.

DOSS measure is not symmetric – we define the similarity of one set relative to another. In case of similar cardinality of V_1 and V_2 this measure is close to symmetric, but if one set is much bigger than the second, then the difference is significant – for instance, $V_2 \subseteq V_1$, the $DOSS(D_2|D_1) = 1$ but $DOSS(D_1|D_2) < 1$.

We apply the proposed DOSS measure to the SeFNet-Healthcare database. Figure 7.4 indicates the values for each pair of datasets. This matrix is not symmetric, the similarity of smaller datasets (containing a small number of features) to large datasets is higher than in the opposite case. What is more, the DOSS similarity is highly correlated with a number of shared features between two datasets. However, DOSS is more robust when dealing with terms that are close in meaning but not exactly the same even i.e., if the intersection of a feature set is empty, the DOSS of such datasets is greater than zero.

7.5 Application of SeFNet

In this section, we present the motivation for why SeFNet methodology should be used to systematize tabular data. In the following sections, we trace three potential development directions: the first relates to data scientist's interaction with domain experts. The second shows the potential of using SeFNet in meta-learning. The third application refers to informed machine learning and resource for explainable machine learning techniques.

7.5.1 Assistance for Data Scientists

In today's world, where automation is on the rise, there's a growing need to understand predictive problems. In specific fields, involving experts from that domain and keeping open communication during automation is crucial [140]. SeFNet plays a dual role in that challenge by tapping into domain expertise and using data to broaden both the data scientist's and domain expert's perspectives. By establishing connections among different factors and datasets, SeFNet assists semantic exploration of past experiments, making it simpler to understand their outcomes.

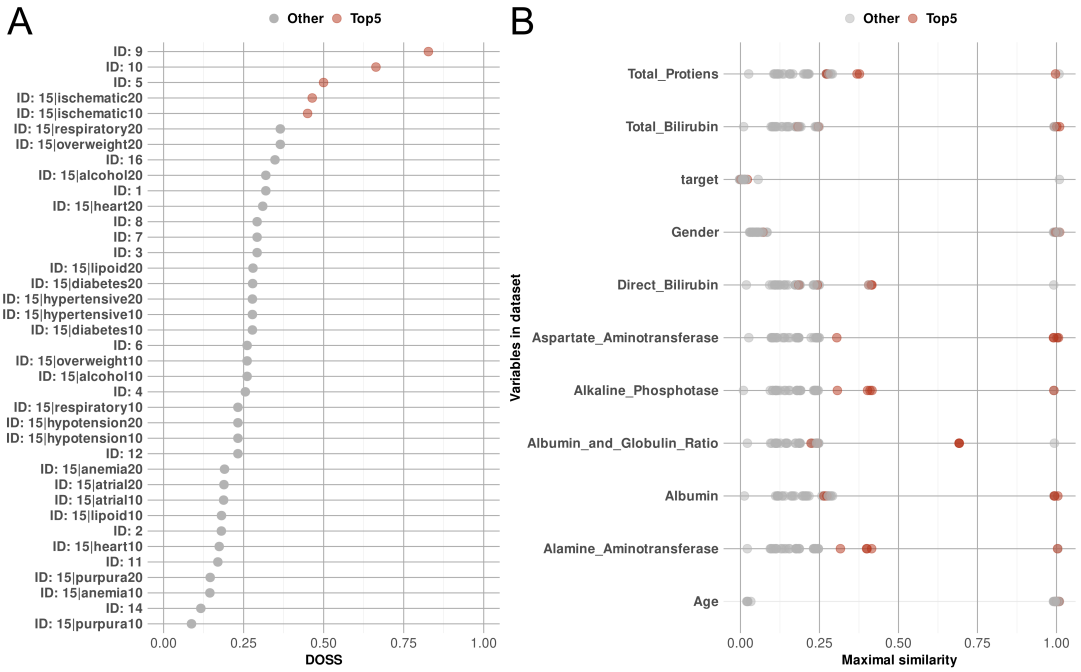

Figure 7.5: Similarity analysis between the **target ILPD dataset (id=13)** and the collections in SeFNet-Healthcare. We start the analysis at the level of whole datasets and then proceed down to the analysis of individual features. In panel A: DOSS measure for target ILPD dataset (id=13) and datasets collected in SeFNet-Healthcare. With dataset metaMIMIC (id=15), 12 tasks are associated because of different target and explanatory variables (see [233]). Among all, data scientist can pay more attention to the first five most similar datasets, indicated with color. In panel B: Going down to the level of analysis of individual variables, we can see for which features in the target datasets in other datasets in SeFNet-Healthcare their exact counterparts or at least correlated features have been found.

Consider an experienced data scientist specializing in developing machine learning models for disease diagnosis. This expert has a track record of utilizing SeFNet-Healthcare to organize and execute previous projects and experiments. When confronted with a new predictive challenge, the initial phase involves grasping the essence of the problem, identifying the most informative features, and unraveling their interactions. Let's assume we have a developed SeFNet-Healthcare repository, and we want to solve a new prediction

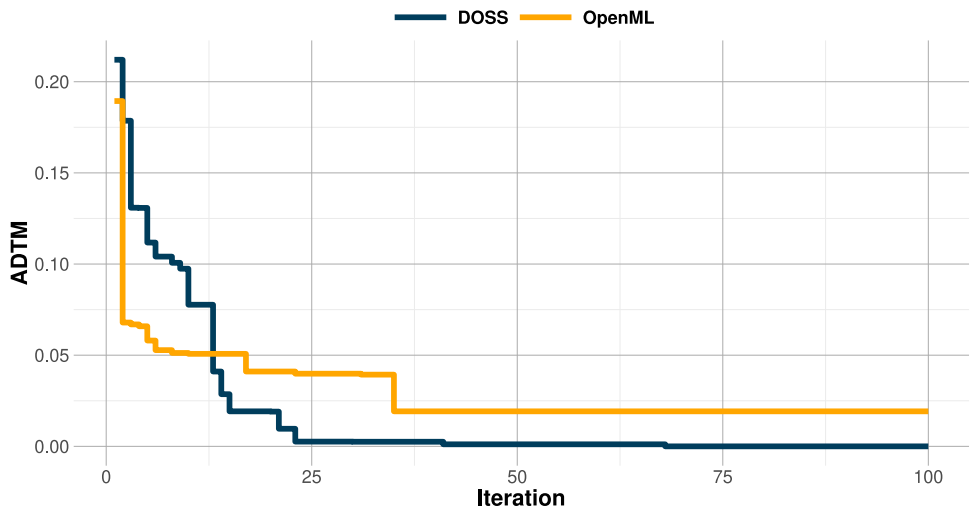

Figure 7.6: Average distance to maximum (ADTM) for two portfolio-based HPO. Navy blue line corresponds to optimization based on DOSS weighting. Yellow one indicates HPO with a portfolio built for OpenML experiments without any weighting.

problem represented as dataset with id 13 - *ILPD dataset*. In Figure 7.5A, we see a comparison of the DOSS measure between the considered dataset and the others. We can clearly see that the 5 datasets are most similar to our new dataset. The DOSS measure for all sets ranges from 0 to 0.8, but for the most similar sets it is above 0.4.

We can then compare similarity to features in datasets whose DOSS is greater than 0.4. In Figure 7.5B, we presented for each variable contained in *ILPD dataset* information about its SM similarity to variables occurring in other datasets. The datasets are divided into two categories – those most similar (DOSS greater than 0.4 – top 5) and the remaining datasets. For some features, such as `Total_Bilirubin` or `Albumin` in a some datasets, the exact equivalent of a given variable is found. The high similarity but slightly below one means that this variable may have been observed in different datasets but comes from other medical procedures. Analysis provided in Figure 7.5B allows us to verify whether the predictive problem is properly defined, and whether variables that were used in another study and were significant in the corresponding models are included in this dataset. This often allows validation of the analysis assumptions at an early stage of the collaboration between the data scientist and the domain expert.

7.5.2 Additional level of dataset similarity in meta-learning

By introducing the structure of features and assessing similarity, SeFNet enables the exploration of semantic representations for tabular datasets. It is important in meta-learning

where we search for descriptive meta-features to assess the potential of transferring information.

The semantic information embedded in DOSS can enhance the pool of existing meta-features (see Section 7.2) or be used as a standalone representation. For example, in hyperparameter optimization, DOSS between datasets can serve as weights for tasks from the meta-train set to pay more attention to experiments containing features close to these in meta-test tasks. In Figure 7.6 we show results of hyperparameter optimization of XGBoost algorithm [36]. Based on the meta-training set and A-SMFO[229] algorithm, we determine portfolios of the potentially best configurations of the ML algorithm's hyperparameters, and their performance is evaluated on the new dataset (meta-test). We compare two strategies. In the first, we build a meta-training set from SeFNet, and in the second, we build a meta-test set from an independent OpenML datasets (not included in SeFNet-Healthcare). In both cases, the meta-test is a dataset from SeFNet-Healthcare, but in the case of the first approach we used a one-dataset-out. In the case of SeFNet, we introduce the DOSS similarity measure of the new dataset to the other datasets. The portfolio determined for the independent OpenML datasets in this use case is a baseline approach to meta-learning portfolio construction. In the first iterations, portfolios determined based on OpenML were more successful, but after 12 iterations, a portfolio determined based on SeFNet-Healthcare using DOSS is better.

7.5.3 Enhance human understandability of domain models

SeFNet injects external knowledge into each dataset independently, which broadens the perspective for explainable machine learning techniques. In the case of highly specialized models based on data containing very specific variables, consideration of ontologies and hierarchical structure of concepts can help in the perception of explainability techniques.Depending on the audience, it may be desirable to use variables that are fairly general, but if we have an expert audience our goal may be to emphasize the impact of specialized factors that are lower in the hierarchy of concepts. This adaptability of the depth of analysis is important in informed machine learning techniques, which, using additional information, are intended to improve the performance of models but at the same time preserve their interpretability.

A very good example is the modified Trepan algorithm [44] – used to explain neural

networks through decision tree splitting rules. In [42], the authors used the semantic meaning of individual variables and modified their weights in the selection of variables used to create an explanation (see Figure 7.7). Through an empirical study, they showed that this way of creating an explanation is better received by users. Hierarchy and links between features may be also informative while creating counterfactual explanations [166].

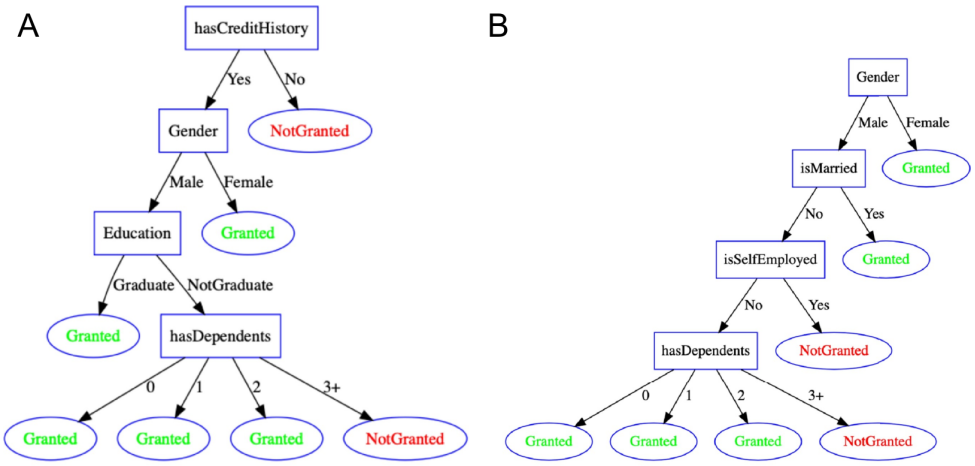

Figure 7.7: Explanation prepared without and without domain knowledge by Trepan technique [42]. Hierarchical structure of features in single datasets inject new information about semantics and can be applied in XAI methods.

7.6 Discussion

Automated Data Science. In the preceding section, we illustrate potential applications of heterogeneous tabular data systematized with SeFNet. Given SeFNet's versatility across various stages of the machine learning pipeline, it aligns seamlessly with the Automated Data Science [47]. Considering the breadth of this field, De Bie et al. [47] segmented the entire process into four stages and delineated two dimensions for assessing their complexity. The first dimension pertains to the incorporation of domain knowledge, while the second evaluates the extent of qualitative assessment feasible for the methods presented, delineating how open-ended the process is. The SeFNet touches on both dimensions. Through the use of ontologies, it makes possible to inject additional knowledge into a single dataset, and through connections to other data as new transfer learning techniques develop, it is possible that more advanced techniques may become available.

So far, the aspect of semantic similarities has been overlooked in automated machine learning, but SeFNet could be the foundation for new data-driven methods. Based on

similar features we can enhance existing methods of specific preprocessing techniques, e.g. in the imputation of missing data. Using feature similarity comparison, data can be selected to compare ranges of values or entire distributions from which the considered feature should be imputed.

Semantic similarity may also be incorporated into tabular dataset embeddings. Because of the new level of similarity based on semantic relations, representations can be validated in terms of the expressivity of the semantic features. Existing representations were not provided with this additional information but users expect that some similarity between results may come from the similarity in the definition of prediction problem [2, 111]. This is a place to benefit from SeFNet which can be used as a resource to take into account the creation of representations but also to verify what features encode the obtained representations.

SeFNet also provides the missing layer of relationships between different experiments. Filling this gap makes navigation between available experiments more accessible and intuitive for domain experts.

Semantic layer in Bussiness Intelligence. SeFNet mirrors the concept of Semantic Layer in Business Intelligence (BI) in Automating Data Science, aiming to integrate knowledge and provide users with a comprehensive understanding of the problem at hand. While the business community has long recognized the importance of semantic layers in enhancing data analysis and decision-making processes, the machine learning community has largely overlooked this aspect. SeFNet, however, serves as a pivotal foundation for bridging this gap and fostering the development of techniques that are more attuned to the needs and preferences of end users. SeFNet also addresses the challenge of how to store domain meta-data effectively [175].

Limitations. In order to ensure the proper use of the individual datasets, we strongly advise researchers to refer to the official documentation and resources. We must emphasize that we are not the creators of the datasets and, therefore cannot offer any endorsement or guarantees regarding their accuracy. Due to SNOMED-CT policy, we cannot share the entire structure of this ontology, but we can share information about the terms used. To gain insight into the graph structure, it is necessary to acquire access to SNOMED-CT.

7.7 Conclusions

In this paper, we introduced the SeFNet methodology reflecting the semantics of features found in tabular datasets. To our knowledge, this is the first work incorporating semantic feature information in the representation of tasks based on tabular data. This additional layer of information allows us to gain insights into the semantic structure and may lead to more informed decision-making in designing machine learning solutions for specific tasks. It aligns well with the concept of Automated Data Science as it facilitates the leveraging of external knowledge and provides users with a comprehensive understanding of the problem at hand.

A very important aspect of future work is the integration of SeFNet into end-to-end machine learning pipelines. Investigating how SeFNet can be seamlessly incorporated into various stages of the machine learning processes, such as feature selection, data imputation, or model optimization, would enhance its practical utility and adoption in real-world applications. The next point that can be developed is refining the methods for quantifying the semantic similarity between datasets using SeFNet. The current approach utilizes the Dataset Ontology-based Semantic Similarity (DOSS) aggregation, but further research can explore alternative techniques.

This project lays the foundation for feature-centric approaches in machine learning. Creating semantic links between stand-alone datasets provides new possibilities in the realm of tabular data. It transforms groups of singular machine learning algorithms into manifolds of interconnected tasks from which we can derive information to achieve new, previously unattainable results.

Chapter 8

Summary

The primary objective of automated data science (AutoDS) is to assist users in creating decision-making systems through machine learning models. The developers of AutoDS methods aim to cater to a diverse audience with varying needs. Ensuring that AutoDS systems are user-centric rather than merely technical support is critical for their practical adoption and success. A black-box approach to AutoDS is insufficient, as users need transparency and understanding of how these systems operate. The effectiveness of machine learning models alone does not justify their deployment. Users must comprehend the decision-making processes and rationale behind the selection of operations within the AutoDS pipeline to trust and effectively utilize these systems.

This thesis focuses on evaluating and incorporating domain knowledge into AutoDS systems. In the first part, the thesis emphasizes the importance of going beyond numerical evaluation to provide deeper insights into why certain operations are applied within AutoDS.

The first thesis contribution is the independent validation of data imputation methods during Data Engineering. Given the prevalence of missing data, benchmarking these methods narrows down the techniques for AutoDS, reducing the search space and simplifying the process for users. Also, from the perspective of users building AutoDS systems, such an experiment gives a conclusion about the possibility of just simplifying this space.

The second hypothesis explores the potential of explanatory machine learning techniques in validating the usefulness of meta-features in hyperparameter optimization for tabular datasets. Traditionally, XAI methods have been used to analyze the impact of hyperparameters on model performance. This thesis introduces a methodology to explain meta-models and evaluate the effectiveness of meta-features, addressing a crucial issue in

the field.

The third hypothesis aims to enhance the explainability of machine learning model benchmarks using the EPP score. EPP's probabilistic interpretation of model quality differences aids users, especially non-experts, in drawing conclusions. This simple interpretation improves communication between data science experts and end-users, facilitating a better understanding of model selection.

The fourth contribution focuses on applying domain knowledge in AutoDS systems by proposing a consolidated learning methodology. This approach builds a portfolio of datasets for meta-learning in hyperparameter transfer, leveraging preliminary knowledge of new datasets. Incorporating results from experiments on similar data can enhance hyperparameter transfer and optimization, boosting user confidence and ensuring high-quality meta-learning.

The final original contribution is a methodology for describing tabular datasets through their semantic meaning using a domain ontology. So far, tabular datasets are treated independently. The SeFNet methodology structures datasets based on the semantic meaning of their features, improving communication with domain experts and enhancing the exploration of experiments and datasets.

A limitation of this work is its focus on medical datasets, necessitating further evaluation and validation on other datasets. Despite this, the methodologies and resources provided, such as the novel metaMIMIC and SeFNet-Healthcare datasets, offer valuable benchmarks for new AutoDS approaches. Future work should extend these methodologies to diverse datasets and explore additional domains to validate their generalizability and effectiveness. Additionally, further research could focus on refining these methodologies to enhance their robustness and applicability in various real-world scenarios.

This thesis demonstrates that addressing user needs, particularly the integration of domain knowledge, can overcome significant barriers to developing more interactive AutoDS methods. By incorporating methodologies that bring domain knowledge into the system, the work highlights the potential for advancing AutoDS to better meet user requirements and enhance system interactivity.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. TensorFlow: a system for Large-Scale machine learning. In *Proceedings of the USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, 2016.

[2] A. Alaa and M. Schaar. AutoPrognosis: Automated Clinical Prognostic Modeling via Bayesian Optimization with Structured Kernel Learning. In *Proceedings of the International Conference on Machine Learning*, pages 139–148, 2018.

[3] E. Alcobaça, F. Siqueira, A. Rivolli, L. P. Garcia, J. T. Oliva, and A. C. De Carvalho. MFE: Towards reproducible meta-feature extraction. *The Journal of Machine Learning Research*, 21(1):4503–4507, 2020.

[4] H. Alibrahim and S. A. Ludwig. Hyperparameter optimization: comparing genetic algorithm against grid search and bayesian optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1551–1559, 2021.

[5] E. Alpaydin. Combined 5 &Times; 2 Cv F Test for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 11(8):1885–1892, 1999. doi: 10.1162/089976699300016007.

[6] Anaconda. 2022 State of Data Science, 2022. URL `https://www.anaconda.com/resources/whitepapers/state-of-data-science-report-2022`. Accessed on 20.04.2024.

[7] R. R. Andridge and R. J. A. Little. A Review of Hot Deck Imputation for Survey Non-response. 78(1):40–64, 2010. doi: 10.1111/j.1751-5823.2010.00103.x.

[8] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene ontology:

tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000. doi: 10.1038/s41597-019-0206-3.

[9] P. K. Ashvini Balte, Nitin Pise. Meta-Learning with Landmarking: A Survey. *International Journal of Computer Applications*, 105(8):47–51, 2014. doi: 10.5120/18401-9666.

[10] A. Ballatore, M. Bertolotto, and D. C. Wilson. Geographic knowledge extraction and semantic similarity in OpenStreetMap. *Knowledge and Information Systems*, 37(1):61–81, 2013. doi: 10.1007/s10115-012-0571-0.

[11] K. Beckh, S. Müller, M. Jakobs, V. Toborek, H. Tan, R. Fischer, P. Welke, S. Houben, and L. von Rueden. Harnessing Prior Knowledge for Explainable Machine Learning: An Overview. In *Proceedings of the IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 450–463, 2023.

[12] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

[13] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

[14] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science and Discovery*, 8(1):13–19, 2015. doi: 10.1088/1749-4699/8/1/014008.

[15] P. Biecek. DALEX: explainers for complex predictive models in R. *The Journal of Machine Learning Research*, 19(1):3245–3249, 2018.

[16] P. Biecek and T. Burzykowski. *Explanatory Model Analysis. Explore, Explain and Examine Predictive Models.* Chapman and Hall/CRC, New York, 2021.

[17] B. Bilalli, A. Abelló, and T. Aluja-Banet. On the predictive power of meta-features in OpenML. *International Journal of Applied Mathematics and Computer Science*, 27(4):697–712, 2017. doi: 10.1515/amcs-2017-0048.

[18] B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine Learning in R. *Journal of Machine Learning Research*, 17(170):1–5, 2016.

[19] B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren. OpenML benchmarking suites and the OpenML100. *stat*, 1050:11, 2017.

[20] E. Blanchard, M. Harzallah, and P. Kuntz. A generic framework for comparing semantic similarities on a subsumption hierarchy. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 20–24, 2008.

[21] R. R. Bouckaert. Choosing Between Two Learning Algorithms Based on Calibrated Tests. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 51–58, 2003.

[22] X. Bouthillier and G. Varoquaux. *Survey of machine-learning experimental methods at NeurIPS2019 and ICLR2020*. PhD thesis, Inria Saclay Ile de France, 2020.

[23] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[24] P. Brazdil, C. G. Carrier, C. Soares, and R. Vilalta. *Metalearning*. Springer Berlin, Heidelberg, 2008.

[25] P. B. Brazdil and C. Soares. A comparison of ranking methods for classification algorithm selection. In *Proceedings of the European Conference on Machine Learning*, pages 63–75, Berlin Heidelberg, 2000. doi: 10.1007/3-540-45164-1_8.

[26] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[27] J. Brierley, B. O'Sullivan, H. Asamura, D. Byrd, S. H. Huang, A. Lee, M. Piñeros, M. Mason, F. Y. Moraes, W. Rösler, et al. Global Consultation on Cancer Staging: promoting consistent understanding and use. *Nature Reviews Clinical Oncology*, 16 (12):763–771, 2019.

[28] M. L. Brown and J. F. Kros. Data mining and the impact of missing data. *Industrial Management and Data Systems*, 103(8-9):611–621, 2003. doi: 10.1108/02635570310497657.

[29] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proceedings of the Pacific-Asia Conference in Knowledge Discovery and Data Mining (PAKDD)*, pages 160–172. Springer, 2013.

[30] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.

[31] S. R. Carroll, I. Garba, R. Plevel, D. Small-Rodriguez, V. Y. Hiratsuka, M. Hudson, and N. Garrison. Using indigenous standards to implement the CARE principles: Setting expectations through tribal research codes. *Frontiers in Genetics*, 13:823309, 2022.

[32] J. A. Casey, B. S. Schwartz, W. F. Stewart, and N. E. Adler. Using electronic health records for population health research: a review of methods and applications. *Annual review of public health*, 37:61–81, 2016.

[33] B. Celik and J. Vanschoren. Adaptation strategies for automated machine learning on evolving data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3067–3078, 2021.

[34] O. Chapelle and L. Li. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems*, volume 24.

[35] J. Chen, P. Hu, E. Jimenez-Ruiz, O. M. Holter, D. Antonyrajah, and I. Horrocks. Owl2vec*: Embedding of owl ontologies. *Machine Learning*, 110(7):1813–1845, 2021.

[36] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[37] X. Chen, M. M. Singh, and P. Geyer. Utilizing domain knowledge: robust machine learning for building energy performance prediction with small, inconsistent datasets. *Knowledge-Based Systems*, page 111774, 2024. doi: 10.1016/j.knosys.2024.111774.

[38] C. M. Childs and N. R. Washburn. Embedding domain knowledge for machine learning of complex material systems. *MRS Communications*, 9(3):806–820, 2019.

[39] E. Choi, M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun. GRAM: graph-based attention model for healthcare representation learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 787–795, 2017.

[40] M. Chui, J. Manyika, M. Miremadi, N. Henke, R. Chung, P. Nel, and S. Malhotra. Notes from the AI frontier: insights from hundreds of use cases, 2018.

[41] A. P. Clark, K. L. Howard, A. T. Woods, I. S. Penton-Voak, and C. Neumann. Why rate when you could compare? Using the "EloChoice" package to assess pairwise comparisons of perceived physical strength. *PLOS ONE*, 13:1–16, 2018. doi: 10.1371/journal.pone.0190393.

[42] R. Confalonieri and T. R. Besold. TREPAN Reloaded: A Knowledge-Driven Approach to Explaining Black-Box Models. In *Proceedings of the European Conference on Artificial Intelligence*, 2019.

[43] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[44] M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, NIPS'95, page 24–30, Cambridge, MA, USA, 1995.

[45] A. Crisan and B. Fiore-Gartland. Fits and Starts: Enterprise Use of AutoML and the Role of Humans in the Loop. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021. doi: 10.1145/3411764.3445775.

[46] C. Davis and C. Giraud-Carrier. Annotative experts for hyperparameter selection. In *AutoML Workshop at ICML*, 2018.

[47] T. De Bie, L. De Raedt, J. Hernández-Orallo, H. H. Hoos, P. Smyth, and C. K. I. Williams. Automating Data Science: Prospects and Challenges. *Communications of the ACM*, 65(3):76–87, 2022. doi: 10.1145/3495256.

[48] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[49] F. Dhombres and O. Bodenreider. Interoperability between phenotypes in research and healthcare terminologies—Investigating partial mappings between HPO and SNOMED CT. *Journal of Biomedical Semantics*, 7:1–13, 2016. doi: 10.1186/s13326-016-0047-3.

[50] T. G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998. doi: 10.1162/089976698300017197.

[51] J. Drozdal, J. Weisz, D. Wang, G. Dass, B. Yao, C. Zhao, M. Muller, L. Ju, and H. Su. Trust in AutoML: Exploring Information Needs for Establishing Trust in Automated Machine Learning Systems. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 297–307, 2020. doi: 10.1145/3377325.3377501.

[52] D. Dua and C. Graff. UCI Machine Learning Repository, 2017.

[53] M. A. Duggan, W. F. Anderson, S. Altekruse, L. Penberthy, and M. E. Sherman. The surveillance, epidemiology and end results (SEER) program and pathology: towards strengthening the critical relationship. *The American Journal of Surgical Pathology*, 40(12):e94–e102, 2016. doi: 10.1097/PAS.0000000000000749.

[54] H. Edwards and A. Storkey. Towards a neural statistician. In *Proceedings of the International Conference on Learning Representations*, pages 1–13, 2016.

[55] A. Elo and S. Sloan. *The Rating of Chess Players, Past and Present.* Ishi Press International, 2008. ISBN 9780923891275.

[56] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[57] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv: 2003.06505*, 2020. doi: 10.48550/arXiv.2003.06505.

[58] H. J. Escalante, M. Montes, and E. Sucar. Ensemble particle swarm model selection. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[59] S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1437–1446, 2018.

[60] M. Feurer, J. T. Springenberg, and F. Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, volume 29, pages 1128—-1135, 2015. doi: https://doi.org/10.1609/aaai.v29i1.9354.

[61] M. Feurer, B. Letham, and E. Bakshy. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML Workshop at ICML*, volume 7, page 5, 2018.

[62] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. *Auto-sklearn: Efficient and Robust Automated Machine Learning*, pages 113–134. Cham, 2019. doi: 10.1007/978-3-030-05318-5_6.

[63] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, and F. Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*, 23(261):1–61, 2022.

[64] A. Fisher, C. Rudin, and F. Dominici. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

[65] E. Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. 1985.

[66] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. In *Annals of Statistics*, volume 28, pages 337–407. 2000. doi: 10.1214/aos/1016218223.

[67] J. H. Friedman, B. E. Popescu, et al. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

[68] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, et al. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(1), 2012.

[69] M. Gaur, K. Faldu, and A. Sheth. Semantics of the black-box: Can knowledge graphs help make deep learning systems more interpretable and explainable? *IEEE Internet Computing*, 25(1):51–59, 2021.

[70] Gene Ontology Consortium. The Gene Ontology resource: enriching a GOld mine. *Nucleic Acids Research*, 49(D1):D325–D334, 2021. doi: 10.1093/nar/gkaa1113.

[71] P. Gijsbers, M. L. P. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024.

[72] Y. Gil, J. Honaker, S. Gupta, Y. Ma, V. D'Orazio, D. Garijo, S. Gadewar, Q. Yang, and N. Jahanshad. Towards Human-Guided Machine Learning. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 614–624, 2019. doi: 10.1145/3301275.3302324.

[73] C. Giraud-Carrier et al. A meta-learning assistant for providing user support in data mining and machine learning, 1999-2001.

[74] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, 101(23):215–220, 2000.

[75] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015. doi: 10.1080/10618600.2014.907095.

[76] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google Vizier: A Service for Black-Box Optimization. 2017. doi: 10.1145/3097983.3098043.

[77] A. Gottlieb, G. Y. Stein, E. Ruppin, and R. Sharan. PREDICT: a method for inferring novel drug indications with application to personalized medicine. *Molecular Systems Biology*, 7(1):496, 2011. doi: 10.1038/msb.2011.26.

[78] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 6(4):309–322, 2008.

[79] B. Greenwell, B. Boehmke, J. Cunningham, and G. Developers. *gbm: Generalized Boosted Regression Models*, 2020. R package version 2.1.8.

[80] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993. doi: 10.1006/KNAC.1993.1008.

[81] E. Guerrero Vázquez, A. Yañez Escolano, P. Galindo Riaño, and J. Pizarro Junquera. Repeated Measures Multiple Comparison Procedures Applied to Model Selection in Neural Networks. In *Bio-Inspired Applications of Connectionism*, pages 88–95, Berlin, Heidelberg, 2001.

[82] I. Guyon, L. Sun-Hosoya, M. Boullé, H. J. Escalante, S. Escalera, Z. Liu, D. Jajetic, B. Ray, M. Saeed, M. Sebag, A. Statnikov, W.-W. Tu, and E. Viegas. *Analysis of the AutoML Challenge Series 2015–2018*, pages 177–219. Cham, 2019. doi: 10.1007/978-3-030-05318-5_10.

[83] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[84] S. Harispe, D. Sánchez, S. Ranwez, S. Janaqi, and J. Montmain. A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain. *Journal of Biomedical Informatics*, 48:38–53, 2014. doi: 10.1016/j.jbi.2013.11.006.

[85] S. Harispe, S. Ranwez, S. Janaqi, and J. Montmain. Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies*, 8(1):1–254, 2015. doi: 10.2200/S00639ED1V01Y201504HLT027.

[86] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown, D. Botstein, et al. Imputing missing data for gene expression arrays. *Stanford University Statistics Department Technical report*, 1999.

[87] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh. Matrix completion and low-rank SVD via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015.

[88] X. He, K. Zhao, and X. Chu. AutoML: A survey of the state-of-the-art. *Knowledge-based Systems*, 212:106622, 2021. doi: 10.1016/j.knosys.2020.106622.

[89] R. Herbrich, T. Minka, and T. Graepel. TrueSkill(TM): A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems*, volume 19, 2006.

[90] L. B. Hewitt, M. I. Nye, A. Gane, T. Jaakkola, and J. B. Tenenbaum. The variational homoencoder: Learning to learn high capacity generative models from few examples. 2018.

[91] J. U. Hibbard, I. Wilkins, L. Sun, K. Gregory, S. Haberman, M. Hoffman, M. A. Kominiarek, U. Reddy, J. Bailit, D. W. Branch, et al. Respiratory morbidity in late preterm births. *JAMA: the journal of the American Medical Association*, 304(4): 419, 2010.

[92] K. Hippalgaonkar, Q. Li, X. Wang, J. W. Fisher III, J. Kirkpatrick, and T. Buonassisi. Knowledge-integrated machine learning for materials: lessons from gameplaying and robotics. *Nature Reviews Materials*, 8(4):241–260, 2023.

[93] J. Honaker, G. King, and M. Blackwell. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011. doi: 10.18637/jss.v045.i07.

[94] G. Hooker and L. Mentch. Please Stop Permuting Features An Explanation and Alternatives. *arXiv preprint arXiv:1905.03151*, 2019.

[95] D. Hull. *Information retrieval using statistical classification*. Phd thesis, Stanford University, 1994. Available at `https://user.eng.umd.edu/~oard/pdf/hull.pdf`.

[96] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *Proceedings of the International Conference*

on Learning and Intelligent Optimization, pages 507–523. Springer, 2011. doi: 10.1007/978-3-642-25566-3_40.

[97] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the International Conference on Machine Learning*, pages 754–762. PMLR, 2014.

[98] F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. 2019.

[99] C. Hvarfner, D. Stoll, A. Souza, M. Lindauer, F. Hutter, and L. Nardi. π BO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization. In *Proceeding of the International Conference of Learning Representations*, 2022.

[100] S. L. Hyland, M. Faltys, M. Hüser, X. Lyu, T. Gumbsch, C. Esteban, C. Bock, M. Horn, M. Moor, B. Rieck, et al. Early prediction of circulatory failure in the intensive care unit using machine learning. *Nature Medicine*, 26(3):364–373, 2020. doi: 10.1038/s41591-020-0789-4.

[101] T. Iwata and A. Kumagai. Meta-learning from tasks with heterogeneous attribute spaces. *Advances in Neural Information Processing Systems*, 33, 2020.

[102] T. Iwata and A. Kumagai. Sharing Knowledge for Meta-learning with Feature Descriptions. *Advances in Neural Information Processing Systems*, 35, 2022.

[103] A. Jadhav, D. Pramod, and K. Ramanathan. Comparison of Performance of Data Imputation Methods for Numeric Dataset. *Applied Artificial Intelligence*, 33(10): 913–933, 2019. doi: 10.1080/08839514.2019.1637138.

[104] K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 240–248, 2016.

[105] A. D. Jesus, A. Liefooghe, B. Derbel, and L. Paquete. Algorithm selection of anytime algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 850–858, 2020.

[106] J. J. Jiang and D. W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proceedings of International Conference Research on Computational Linguistics*, 1997.

[107] S. Jiang, W. Wu, N. Tomita, C. Ganoe, and S. Hassanpour. Multi-Ontology Refined Embeddings (MORE): A hybrid multi-ontology and corpus-based semantic representation model for biomedical concepts. *Journal of Biomedical Informatics*, 111: 103581, 2020. doi: 10.1016/j.jbi.2020.103581.

[108] H. Jin, F. Chollet, Q. Song, and X. Hu. AutoKeras: An AutoML Library for Deep Learning. *Journal of Machine Learning Research*, 24(6):1–6, 2023.

[109] A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. Celi, and R. Mark. MIMIC-IV (version 1.0), 2020.

[110] A. E. W. Johnson, T. J. Pollard, and R. G. Mark. Reproducibility in critical care: a mortality prediction case study. In *Proceedings of the Machine Learning for Health Care Conference (MLHC)*, pages 361–376, 2017.

[111] H. S. Jomaa, L. Schmidt-Thieme, and J. Grabocka. Dataset2vec: Learning dataset meta-features. *Data Mining and Knowledge Discovery*, 35(3):964–985, 2021. doi: 10.1007/s10618-021-00737-9.

[112] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998. doi: 10.1023/A:1008306431147.

[113] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. 13(4):455–492, 1998. doi: 10.1023/A:1008306431147.

[114] J. Josse and F. Husson. missMDA: A package for handling missing values in multivariate data analysis. *Journal of Statistical Software*, 70(1):1–31, 2016. doi: 10.18637/jss.v070.i01.

[115] M. Karelson, V. S. Lobanov, and A. R. Katritzky. Quantum-chemical descriptors in QSAR/QSPR studies. *Chemical reviews*, 96(3):1027–1044, 1996.

[116] R. Kieft, E. M. Vreeke, E. De Groot, H. de Graaf-Waar, C. H. van Gool, N. Koster, H. ten Napel, A. L. Francke, and D. M. Delnoij. Mapping the Dutch SNOMED CT subset to Omaha system, NANDA international and international classification of functioning, disability and health. *International Journal of Medical Informatics*, 111:77–82, 2018. doi: 10.1016/j.ijmedinf.2017.12.025.

[117] J. Kim, S. Kim, and S. Choi. Learning to warm-start Bayesian hyperparameter optimization. *arXiv preprint arXiv:1710.06219*, 2017.

[118] J. Kocoń and M. Maziarz. Mapping WordNet onto human brain connectome in emotion processing and semantic similarity recognition. *Information Processing & Management*, 58(3):102530, 2021. doi: 10.1016/j.ipm.2021.102530.

[119] S. Köhler, M. Gargano, N. Matentzoglu, L. C. Carmody, D. Lewis-Smith, N. A. Vasilevsky, D. Danis, G. Balagura, G. Baynam, A. M. Brower, et al. The Human Phenotype Ontology in 2021. *Nucleic Acids Research*, 49(D1):D1207–D1217, 2021. doi: https://doi.org/10.1093/nar/gkaa1043.

[120] A. Kowarik and M. Templ. Imputation with the R Package VIM. *JSS Journal of Statistical Software*, 74(7), 2016. doi: 10.18637/jss.v074.i07.

[121] J. L. Koyner, K. A. Carey, D. P. Edelson, and M. M. Churpek. The development of a machine learning inpatient acute kidney injury prediction model. *Critical Care Medicine*, 46(7):1070–1077, 2018. doi: 10.1097/CCM.0000000000003123.

[122] W. Kretowicz and P. Biecek. MementoML: Performance of selected machine learning algorithm configurations on OpenML100 datasets. *arXiv preprint arXiv:2008.13162*, 2020.

[123] A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, and J. Moult. Critical assessment of methods of protein structure prediction (CASP) - Round XIII. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1011–1020, 2019. doi: 10.1002/prot.25823.

[124] S. Kumar, I. Oh, S. Schindler, A. M. Lai, P. R. Payne, and A. Gupta. Machine learning for modeling the progression of Alzheimer disease dementia using clinical

data: a systematic literature review. *JAMIA Open*, 4(3):ooab052, 2021. doi: 10.1093/jamiaopen/ooab052.

[125] G. Kyureghian, O. Capps, and R. M. Nayga. A missing variable imputation methodology with an empirical application. *Advances in Econometrics*, 27 A:313–337, 2011. doi: 10.1108/S0731-9053(2011)000027A015.

[126] N. Lavesson and P. Davidsson. Quantifying the impact of learning algorithm parameter tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 6, pages 395–400, 2006.

[127] V. Le and S. Gulwani. FlashExtract: A Framework for Data Extraction by Examples. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 542–553, 2014. doi: 10.1145/2594291.2594333.

[128] E. LeDell and S. Poirier. H2O AutoML: Scalable Automatic Machine Learning. *7th ICML Workshop on Automated Machine Learning (AutoML)*, 2020.

[129] M. F. Lensink, N. Nadzirin, S. Velankar, and S. J. Wodak. Modeling protein-protein, protein-peptide, and protein-oligosaccharide complexes: CAPRI 7th edition. *Proteins: Structure, Function, and Bioinformatics*, 88(8):916–938, 2020. doi: 10.1002/prot.25870.

[130] K. Li and J. Malik. Learning to Optimize. In *Proceeding of the International Conference on Learning Representations*, 2017.

[131] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *Proceedings of the International Conference on Learning Representations*, 2017.

[132] X. Liang, Z. Hu, H. Zhang, L. Lin, and E. P. Xing. Symbolic graph reasoning meets convolutions. volume 31, 2018.

[133] D. Lin et al. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning*, volume 98, pages 296–304, 1998.

[134] M. Lindauer, M. Feurer, K. Eggensperger, A. Biedenkapp, and F. Hutter. Towards Assessing the Impact of Bayesian Optimization's Own Hyperparameters. *arXiv preprint arXiv:1908.06674*, 2019.

[135] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[136] T. Liu, Q. Zhao, and B. Du. Effects of high-flow oxygen therapy on patients with hypoxemia after extubation and predictors of reintubation: a retrospective study based on the MIMIC-IV database. *BMC Pulmonary Medicine*, 21(1):1–15, 2021. doi: 10.1186/s12890-021-01526-2.

[137] S. M. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[138] F. Ma, Q. You, H. Xiao, R. Chitta, J. Zhou, and J. Gao. Kame: Knowledge-based attention model for diagnosis prediction in healthcare. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 743–752, 2018.

[139] R. G. Mantovani, A. L. D. Rossi, E. Alcobaça, J. C. Gertrudes, S. B. Junior, and A. C. P. de Leon Ferreira de Carvalho. Rethinking default values: a low cost and efficient strategy to define hyperparameters. arXiv:2008.00025, 2020.

[140] Y. Mao, D. Wang, M. Muller, K. R. Varshney, I. Baldini, C. Dugan, and A. Mojsilović. How data scientists work together with domain experts in scientific collaborations: To find the right answer or to ask the right question? *Proceedings of the ACM on Human-Computer Interaction*, 3:1–23, 2019. doi: 10.1145/3361118.

[141] F. Martínez-Plumed, P. Barredo, S. Ó. hÉigeartaigh, and J. Hernández-Orallo. Research community dynamics behind popular AI benchmarks. *Nature Machine Intelligence*, 2021. doi: 10.1038/s42256-021-00339-6.

[142] I. Mayer, A. Sportisse, J. Josse, N. Tierney, and N. Vialaneix. R-miss-tastic: a unified platform for missing values methods and workflows. *The R Journal*, 14: 244–266, 2022. doi: 10.32614/RJ-2022-040.

[143] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall / CRC, London, 1989.

[144] B. T. McInnes and T. Pedersen. Evaluating semantic similarity and relatedness over the semantic grouping of clinical term pairs. *Journal of Biomedical Informatics*, 54: 329–336, 2015. doi: 10.1016/j.jbi.2014.11.014.

[145] L. McInnes, J. Healy, N. Saul, and L. Großberger. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861.

[146] Y. Mehta, C. White, A. Zela, A. Krishnakumar, G. Zabergja, S. Moradian, M. Safari, K. Yu, and F. Hutter. NAS-Bench-Suite: NAS Evaluation is (Now) Surprisingly Easy. In *Proceedings of the International Conference on Learning Representations*, 2022.

[147] C. Meng, L. Trinh, N. Xu, J. Enouen, and Y. Liu. Interpretability and fairness evaluation of deep learning models on MIMIC-IV dataset. *Scientific Reports*, 12(1): 7166, 2022. doi: 10.1038/s41598-022-11012-2.

[148] N. Mikolajewicz and S. V. Komarova. Meta-analytic methodology for basic research: a practical guide. *Frontiers in physiology*, 10:344709, 2019.

[149] C. Molnar. *Interpretable Machine Learning*. 2019.

[150] C. Molnar, G. Casalicchio, and B. Bischl. iml: An R package for interpretable machine learning. *Journal of Open Source Software*, 3(26):786, 2018.

[151] A. C. Moorman, S. C. Gordon, L. B. Rupp, P. R. Spradling, E. H. Teshale, M. Lu, D. R. Nerenz, C. C. Nakasato, J. A. Boscarino, E. M. Henkle, et al. Baseline characteristics and mortality among people in care for chronic viral hepatitis: the chronic hepatitis cohort study. *Clinical Infectious Diseases*, 56(1):40–50, 2013.

[152] J. Moosbauer, J. Herbinger, G. Casalicchio, M. Lindauer, and B. Bischl. Explaining hyperparameter optimization via partial dependence plots. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[153] M. Morgan and L. Shepherd. *ExperimentHub: Client to access ExperimentHub resources*, 2023. R package version 2.10.0.

[154] S. Nemati, A. Holder, F. Razmi, M. D. Stanley, G. D. Clifford, and T. G. Buchman. An interpretable machine learning model for accurate prediction of sepsis in the ICU. *Critical Care Medicine*, 46(4):547–553, 2018. doi: 10.1097/CCM.0000000000002936.

[155] C. Oh, E. Gavves, and M. Welling. BOCK: Bayesian optimization with cylindrical kernels. In *Proceedings of the International Conference on Machine Learning*, pages 3868–3877, 2018.

[156] I. Olier, N. Sadawi, G. R. Bickerton, J. Vanschoren, C. Grosan, L. Soldatova, and R. D. King. Meta-QSAR: a large-scale application of meta-learning to drug design and discovery. *Machine Learning*, 107(1):285–311, 2018.

[157] R. S. Olson and J. H. Moore. *TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning*, pages 151–160. 2019. doi: 10.1007/978-3-030-05318-5_8.

[158] C. Panigutti, C. Panigutti, A. Perotti, A. Perotti, A. Perotti, D. Pedreschi, and D. Pedreschi. Doctor XAI: an ontology-based approach to black-box sequential data classification explanations. *FAT**, 2020. doi: 10.1145/3351095.3372855.

[159] C. Panigutti, A. Perotti, A. Panisson, P. Bajardi, and D. Pedreschi. FairLens: Auditing black-box clinical decision support systems. *Information Processing & Management*, 58(5):102657, 2021. doi: 10.1016/j.ipm.2021.102657.

[160] T. Pedersen, S. V. Pakhomov, S. Patwardhan, and C. G. Chute. Measures of semantic similarity and relatedness in the biomedical domain. *Journal of Biomedical Informatics*, 40(3):288–299, 2007. doi: 10.1016/j.jbi.2006.06.004.

[161] 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.

[162] K. Pekala, K. Woznica, and P. Biecek. Triplot: model agnostic measures and visualisations for variable importance in predictive models that take into account the hierarchical correlation structure. arXiv:2104.03403, 2021.

[163] V. Perrone, H. Shen, M. W. Seeger, C. Archambeau, and R. Jenatton. Learning search spaces for Bayesian optimization: Another view of hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[164] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-Learning by Landmarking Various Learning Algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 743–750, 2000.

[165] B. Pfeifer, H. Baniecki, A. Saranti, P. Biecek, and A. Holzinger. Multi-omics disease module detection with an explainable Greedy Decision Forest. *Scientific Reports*, 12(1):16857, 2022. doi: 10.1038/s41598-022-21417-8.

[166] B. Pfeifer, M. Krzyziński, H. Baniecki, A. Saranti, A. Holzinger, and P. Biecek. Explaining and visualizing black-box models through counterfactual paths. *arXiv preprint arXiv:2307.07764*, 2023.

[167] F. Pfisterer, J. N. van Rijn, P. Probst, A. Müller, and B. Bischl. Learning Multiple Defaults for Machine Learning Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 241–242, 2018. doi: 10.1145/3449726.3459523.

[168] J. Pizarro, E. Guerrero, and P. L. Galindo. Multiple comparison procedures applied to model selection. *Neurocomputing*, 48(1):155–173, 2002. doi: 10.1016/S0925-2312(01)00653-1.

[169] D. Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technology*, 2, 2008.

[170] P. Probst, A. L. Boulesteix, and B. Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20: 1–22, 2019.

[171] S. Purushotham, C. Meng, Z. Che, and Y. Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of Biomedical Informatics*, 83:112–134, 2018.

[172] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE transactions on systems, man, and cybernetics*, 19 (1):17–30, 1989.

[173] J. D. Raffa, A. E. Johnson, Z. O'Brien, T. J. Pollard, R. G. Mark, L. A. Celi, D. Pilcher, and O. Badawi. The global open source severity of illness score (GOSSIS). *Critical Care Medicine*, 50(7):1040–1050, 2022. doi: 10.1097/CCM.0000000000005518.

[174] C. E. Rasmussen. Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning: ML Summer Schools 2003*, pages 63–71. Springer, 2004. doi: 10.1007/978-3-540-28650-9_4.

[175] S. Redyuk, Z. Kaoudi, S. Schelter, and V. Markl. DORIAN in action: assisted design of data science pipelines. *Proceedings of VLDB Endowment*, 15(12):3714–3717, 2022. doi: 10.14778/3554821.3554882.

[176] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel. Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, 17(1):83–96, 2014. doi: 10.1007/s10044-012-0280-z.

[177] P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[178] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *International Conference on Knowledge Discovery and Data Mining*, 2016.

[179] A. Rivolli, L. P. Garcia, C. Soares, J. Vanschoren, and A. C. de Carvalho. Meta-features for meta-learning. *Knowledge-Based Systems*, 240:108101, 2022. doi: 10.1016/j.knosys.2021.108101.

[180] C. Rosse and J. L. Mejino Jr. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. *Journal of Biomedical Informatics*, 36(6):478–500, 2003.

[181] C. Roth, R. E. Foraker, P. R. Payne, and P. J. Embi. Community-level determinants of obesity: harnessing the power of electronic health records for retrospective data analysis. *BMC Medical Informatics and Decision Making*, 14(1):1–8, 2014.

[182] D. Rubin. *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, 1987. doi: 10.1002/9780470316696.

[183] D. B. Rubin. Inference and Missing Data. *Biometrika*, 63(3):581, 1976. doi: 10.2307/2335739.

[184] T. Ruotsalo, G. Jacucci, P. Myllymäki, and S. Kaski. Interactive Intent Modeling: Information Discovery beyond Search. *Communications of the ACM*, 58(1):86–92, 2014. doi: 10.1145/2656334.

[185] R. Sadeghi, T. Banerjee, and W. Romine. Early hospital mortality prediction using vital signals. *Smart Health*, 9:265–274, 2018.

[186] S. L. Salzberg. On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. *Data Mining and Knowledge Discovery*, 1(3):317–328, 1997. doi: 10.1023/A:1009752403260.

[187] R. Sass, E. Bergman, A. Biedenkapp, F. Hutter, and M. Lindauer. DeepCAVE: An interactive analysis tool for automated machine learning. *arXiv preprint arXiv:2206.03493*, 2022.

[188] M. Scheffler, M. Aeschlimann, M. Albrecht, T. Bereau, H.-J. Bungartz, C. Felser, M. Greiner, A. Groß, C. T. Koch, K. Kremer, et al. FAIR data enabling new horizons for materials research. *Nature*, 604(7907):635–642, 2022. doi: 10.1038/s41586-022-04501-x.

[189] S. Segrera, J. Pinho, and M. N. Moreno. Information-theoretic measures for meta-learning. In *Hybrid Artificial Intelligence Systems*, pages 458–465, 2008.

[190] K. Seki and J. Mostafa. Gene ontology annotation as text categorization: An empirical study. *Information Processing & Management*, 44(5):1754–1770, 2008. doi: 10.1016/j.ipm.2008.05.003.

[191] J. V. Selby. Linking automated databases for research in managed care settings. *Annals of Internal Medicine*, 127(8_Part_2):719–724, 1997.

[192] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[193] H. Shimodaira. An Approximately Unbiased Test of Phylogenetic Tree Selection. *Systematic Biology*, 51(3):492–508, 2002. doi: 10.1080/10635150290069913.

[194] H. Shimodaira. Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling. *The Annals of Statistics*, 32(6):2616–2641, 2004. doi: 10.1214/009053604000000823.

[195] R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[196] M. Simonov, U. Ugwuowo, E. Moreira, Y. Yamamoto, A. Biswas, M. Martin, J. Testani, and F. P. Wilson. A simple real-time model for predicting acute kidney injury in hospitalized patients in the US: A descriptive modeling study. *PLoS medicine*, 16(7):e1002861, 2019.

[197] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[198] SNOMED International. SNOMED: Executive Summary, 2021. URL `https://www.snomed.org/_files/ugd/900274_8a849a3565054d14a4c94cf1062331a3.pdf`.

[199] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. doi: 10.1016/j.ipm.2009.03.002.

[200] A. Souza, L. Nardi, L. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter. *Bayesian Optimization with a Prior for the Optimum*, pages 265–296. 2021. doi: 10.1007/978-3-030-86523-8_17.

[201] D. J. Stekhoven and P. Bühlmann. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 10 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr597.

[202] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36(2):111–133, 1974.

[203] S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K.-R. Müller. Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology. *Machine Learning and Knowledge Extraction*, 3(2):392–413, 2021.

[204] C. Sudlow, J. Gallacher, N. Allen, V. Beral, P. Burton, J. Danesh, P. Downey, P. Elliott, J. Green, M. Landray, et al. UK biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12(3):e1001779, 2015. doi: 10.1371/journal.pmed.1001779.

[205] C. Sutton, T. Hobson, J. Geddes, and R. Caruana. Data Diff: Interpretable, Executable Summaries of Changes in Distributions for Data Wrangling. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2279–2288, 2018. doi: 10.1145/3219819.3220057.

[206] R. Suzuki and H. Shimodaira. pvclust: an R package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, 22(12):1540–1542, 2006. doi: 10.1093/bioinformatics/btl117.

[207] M. Thandi, S. Brown, and S. T. Wong. Mapping frailty concepts to SNOMED CT. *International Journal of Medical Informatics*, 149:104409, 2021. doi: 10.1016/j.ijmedinf.2021.104409.

[208] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 847–855, 2013. doi: 10.1145/2487575.2487629.

[209] K. Thulasiraman and M. N. Swamy. *Directed Graphs*, chapter 5, pages 97–125. 1992. doi: 10.1002/9781118033104.ch5.

[210] J. W. Tukey et al. *Exploratory data analysis*, volume 2. Pearson, 1977.

[211] L. Turgeman, J. H. May, and R. Sciulli. Insights from a machine learning model for predicting the hospital Length of Stay (LOS) at the time of admission. *Expert Systems with Applications*, 78:376–385, 2017.

[212] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977. doi: 10.1037/0033-295X.84.4.327.

[213] A. Vakhrushev, A. Ryzhkov, M. Savchenko, D. Simakov, R. Damdinov, and A. Tuzhilin. LightAutoML: AutoML Solution for a Large Financial Services Ecosystem. *preprint arXiv:2109.01528*, 2021.

[214] S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011. doi: 10.18637/jss.v045.i03.

[215] M. van Leeuwen. *Interactive Data Exploration Using Pattern Mining*, pages 169–182. Berlin, Heidelberg, 2014. doi: 10.1007/978-3-662-43968-5_9.

[216] J. N. Van Rijn and F. Hutter. Hyperparameter importance across datasets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2367–2376, 2018. doi: 10.1145/3219819.3220058.

[217] J. Vanschoren. *Meta-Learning*, pages 35–61. Cham, 2019. doi: 10.1007/978-3-030-05318-5_6.

[218] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198.

[219] R. Vilalta, C. G. Giraud-Carrier, P. Brazdil, and C. Soares. Using Meta-Learning to Support Data Mining. *International Journal of Computer Science and Applications*, 1(1):31–45, 2004.

[220] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, M. Walczak, J. Garcke, C. Bauckhage, and J. Schuecker. Informed Machine Learning – A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2023. doi: 10.1109/TKDE.2021.3079836.

[221] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018. doi: 10.18653/v1/W18-5446.

[222] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. SuperGLUE: a stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[223] A. Y. Wang, J. H. Sable, and K. A. Spackman. The SNOMED clinical terms development process: refinement and analysis of content. In *Proceedings of the AMIA Symposium*. American Medical Informatics Association, 2002.

[224] D. Wang, J. Andres, J. Weisz, E. Oduor, and C. Dugan. AutoDS: Towards Human-Centered Automation of Data Science. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1—-12. doi: 10.1145/3411764.3445526.

[225] A. Wasay, M. Athanassoulis, and S. Idreos. Queriosity: Automated Data Exploration. In *Proceedings of the IEEE International Congress on Big Data*, pages 716–719, 2015. doi: 10.1109/BigDataCongress.2015.116.

[226] J. Wiens, J. Guttag, and E. Horvitz. A study in transfer learning: leveraging data from multiple hospitals to enhance hospital-specific predictions. *Journal of the American Medical Informatics Association*, 21(4):699–706, 2014. doi: 10.1136/amiajnl-2013-002162.

[227] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):1–9, 2016. doi: 10.1038/sdata.2016.18.

[228] F. Winkelmolen, N. Ivkin, H. F. Bozkurt, and Z. Karnin. Practical and sample efficient zero-shot HPO. *arXiv preprint arXiv:2007.13382*, 2020.

[229] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. Sequential model-free hyperparameter tuning. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1033–1038, 2015. doi: 10.1109/ICDM.2015.20.

[230] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. Learning hyperparameter optimization initializations. In *Proceedings of the IEEE International Conference on*

Data Science and Advanced Analytics, pages 1–10, 2015. doi: 10.1109/dsaa.2015.7344817.

[231] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1): 43–78, 2018. doi: 10.1007/s10994-017-5684-y.

[232] K. Woźnica and P. Biecek. Towards Explainable Meta-learning. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, volume 1524, pages 505–520. 2021. doi: 10.1007/978-3-030-93736-2_38.

[233] K. Woźnica, M. Grzyb, Z. Trafas, and P. Biecek. Consolidated learning: A domain-specific model-free optimization strategy with validation on metaMIMIC benchmarks. *Machine Learning*, 113:4925–4949, 2023. doi: 10.1007/s10994-023-06359-0.

[234] D. Yogatama and G. S. Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 1077–1085, 2014.

[235] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen, M. Michalski, O. Bousquet, S. Gelly, and N. Houlsby. A Large-scale Study of Representation Learning with the Visual Task Adaptation Benchmark. *arXiv preprint arXiv:1910.04867*, 2020.

[236] S.-B. Zhang and Q.-R. Tang. Protein–protein interaction inference based on semantic similarity of gene ontology terms. *Journal of Theoretical Biology*, 401:30–37, 2016.

[237] X. Zhang, B. Qian, Y. Li, C. Yin, X. Wang, and Q. Zheng. KnowRisk: an interpretable knowledge-guided model for disease risk prediction. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1492–1497. IEEE, 2019.

[238] Z. Zhang, K. M. Ho, and Y. Hong. Machine learning for the prediction of volume responsiveness in patients with oliguric acute kidney injury in critical care. *Critical Care*, 23(1):1–10, 2019. doi: 10.1186/s13054-019-2411-z.

[239] N. Zhou, Y. Jiang, T. R. Bergquist, A. J. Lee, B. Z. Kacsoh, A. W. Crocker, K. A. Lewis, G. Georghiou, H. N. Nguyen, M. N. Hamid, et al. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome biology*, 20(1):1–23, 2019. doi: 10.1186/s13059-019-1835-8.

[240] M. Zhu, K. Kobalczyk, A. Petrovic, M. Nikolic, M. van der Schaar, B. Delibasic, and P. Lio. Tabular Few-Shot Generalization Across Heterogeneous Feature Spaces. *arXiv preprint arXiv:2311.10051*, 2023.

[241] L. Zimmer, M. Lindauer, and F. Hutter. Auto-PyTorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021. doi: 10.1109/TPAMI.2021.3067763.