



# Consolidated learning: a domain-specific model-free optimization strategy with validation on metaMIMIC benchmarks

Katarzyna Woźnica<sup>1</sup> · Mateusz Grzyb<sup>1</sup> · Zuzanna Trafas<sup>2</sup> · Przemysław Biecek<sup>1,3</sup>

Received: 30 January 2022 / Revised: 2 March 2023 / Accepted: 9 June 2023 /  
Published online: 7 August 2023  
© The Author(s) 2023

## Abstract

For many machine learning models, a choice of hyperparameters is a crucial step towards achieving high performance. Prevalent meta-learning approaches focus on obtaining good hyperparameter configurations with a limited computational budget for a completely new task based on the results obtained from the prior tasks. This paper proposes a new formulation of the tuning problem, called *consolidated learning*, more suited to practical challenges faced by model developers, in which a large number of predictive models are created on similar datasets. In such settings, we are interested in the total optimization time rather than tuning for a single task. We show that a carefully selected static portfolio of hyperparameter configurations yields good results for anytime optimization, while maintaining the ease of use and implementation. Moreover, we point out how to construct such a portfolio for specific domains. The improvement in the optimization is possible due to the more efficient transfer of hyperparameter configurations between similar tasks. We demonstrate the effectiveness of this approach through an empirical study for the XGBoost algorithm and the newly created metaMIMIC benchmarks of predictive tasks extracted from the MIMIC-IV medical database. In the paper, we show that the potential of *consolidated learning* is considerably greater due to its compatibility with many machine learning application scenarios.

**Keywords** Meta-learning · Hyperparameter optimization · Consolidated learning · Portfolio of hyperparameters

## 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

---

Editors: Tijl De Bie, Jose Hernandez-Orallo, Joaquin Vanschoren, Gaël Varoquaux, Chris Williams.

---

Extended author information available on the last page of the article

quality (Lavesson & Davidsson, 2006; Probst et al., 2019). 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 (Thornton et al., 2013; Bergstra et al., 2015; Olson & Moore, 2019; Feurer et al., 2019) 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 (Bergstra & Bengio, 2012). 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 regions against sampling from the known regions with good performance (Hutter et al., 2011; Bergstra et al., 2011; Snoek et al., 2012). 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 (Escalante et al., 2010; Alibrahim & Ludwig, 2021) or reinforcement learning optimization (Li & Malik, 2017) 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 (Bouthillier & Varoquaux, 2020).

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 (Wistuba et al., 2016; Winkelmolen et al., 2020; Feurer et al., 2019, 2022; Pfisterer et al., 2021; Mantovani et al., 2020). 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.

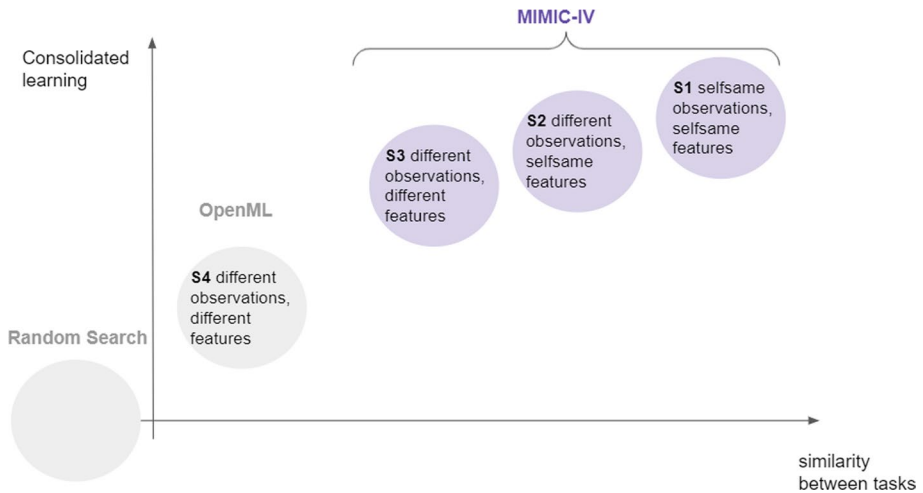
A predefined, limited set of hyperparameter configurations optimized for a wide range of datasets has been shown to give comparable results to Bayesian optimization (Wistuba et al., 2016; Pfisterer et al., 2021) and proves to be even better when considering anytime performance. Moreover, the portfolio 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. (2019) 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 (Feurer et al., 2022). 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 (Vanschoren, 2019; Rivoli et al., 2022), landmarks (Pfahring et al., 2000), or perhaps automatically trained extractors based on neural networks (Edwards & Storkey, 2017; Hewitt et al., 2018; Jomaa et al., 2021). 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 (Woznica & Biecek, 2021). Likewise, the definition of distance and similarity between datasets is underdetermined (Wistuba et al., 2015; Feuer et al., 2015).

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 (Bischl et al., 2017). 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 (Alaa & Schaar, 2018; Guyon et al., 2019; Vakhruhev et al., 2021; Olier et al., 2018) 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 (Johnson et al., 2020). Our experiments reflect various levels of consolidation between the meta-train and the meta-test (see Fig. 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 (Chen & Guestrin, 2016) 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



**Fig. 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

method does not require any additional optimization, is parallelizable, and has strong any-time 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.

## 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 (Jamieson & Talwalkar, 2016) or Hyperband (Li et al., 2017) 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) (Jones et al., 1998) such as Sequential Model-Based Algorithm Configuration (SMAC) (Hutter et al., 2011), Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) or Spearmint (Snoek et al., 2012). However, all of them are computationally demanding, difficult to parallelize and depend on the choice of a starting point (Wistuba et al., 2015). 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 (Falkner et al., 2018). 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 (Oh et al., 2018; Souza et al., 2021; Perrone et al., 2019).

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 (Feurer et al., 2015, 2019; Wistuba et al., 2018). 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 (Vilalta et al., 2004; Reif et al., 2014; Davis & Giraud-Carrier, 2018; Probst et al., 2019).

### 3 Problem definition

#### 3.1 Hyperparameter optimization

Most machine learning algorithms are dependent on the user-specified hyperparameters. An algorithm is trained, i.e., values of internal model parameters are updated iteratively, in accordance with the chosen algorithm and the data provided. So most machine learning algorithms  $\mathcal{A}$  can be parametrized with dataset  $D$  and hyperparameter configuration  $\lambda \in \Lambda \subset \mathbb{R}^d$ . Dataset  $D$  is a finite sample from joint distribution  $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ , where  $\mathcal{X} \subset \mathbb{R}^p$  is the  $p$ -dimensional feature space and  $\mathcal{Y}$  is the target variable space, categorical or numerical. To evaluate the quality of the trained model  $\mathcal{A}(D, \lambda)$  we use the quality function  $F : \mathbb{D} \times \Lambda \rightarrow \mathbb{R}$ , mapping a dataset and hyperparameters to model performance, for instance, accuracy or area under curve (AUC). For every hyperparameter, we attempt to estimate the expected value of the performance for a random sample of observations given as

$$G(D, \lambda) = \mathbb{E}_{D \sim \mathcal{D}} F(D, \lambda). \quad (1)$$

Since we observe only a finite sample from  $\mathcal{D}$ , we estimate Eq. 1 using cross-validation or a holdout subset of data. The main objective of hyperparameter optimization for a prediction

problem  $\mathcal{D}$ , is to find  $\lambda^*$  configuration, optimal with respect to the expected value of model performance

$$\lambda^* = \arg \max_{\lambda \in \Lambda} G(\mathcal{D}, \lambda). \quad (2)$$

We consider different hyperparameter tuning strategies, such as the previously mentioned random search or Bayesian optimization to find this configuration. However, these cannot guarantee that we will find the global maximum and the configuration providing this, so we are interested in finding a configuration that gives a decent model performance, preferably after only a few iterations.

Most optimization strategies are based on a trial consisting of a finite sequence of hyperparameter values  $\Lambda_T = (\lambda_1, \dots, \lambda_T)$ , where  $T$  the number of iterations depending on the available budget. In random search, the set  $\Lambda_T$  is predefined and independent of the prior experiments, and every component  $\lambda_i$  is sampled independently of each other. In Bayesian-based optimization methods  $\Lambda_T$  is selected at runtime and  $\lambda_i$  is determined by a surrogate model which is based on the model performance for the preceding values  $\lambda_1, \dots, \lambda_{i-1}$ .

### 3.2 Meta-learning in hyperparameter optimization

One of the applications of meta-learning in hyperparameter optimization is transferring a hyperparameter portfolio from a set of previously performed experiments to a new dataset. Similar to random search, a predefined meta-learning portfolio is actually a finite sequence of configurations completely defined before tuning for the considered new dataset. However, configurations  $\lambda_i$  are not sampled independently but selected considering the model performance from the previously collected experiments, optimized for these experiments, and possibly extended to the randomly chosen new task. Herein lies the primary assumption of meta-learning and hyperparameter transfer; we assume that configurations that worked for a set of previous tasks will also give a decent performance for new, unknown data. The algorithm by which the portfolio is composed can vary, the most commonly used is greedy search (Wistuba et al., 2015), but average ranking (Brazdil & Soares, 2000) can also be applied.

In this article, the main contribution is the method of determining the family of datasets for which the optimization is performed, not the procedure of completing the transferred portfolio of hyperparameters. That is why we have to highlight two sets of tasks. Firstly, we determine a repository of the already tuned  $N$  tasks and call it the meta-train set  $\mathbf{D}_{meta-train}$ . Every dataset is associated with the distribution  $\mathcal{D}_{meta-train}^i = (\mathcal{X}_{meta-train}^i, \mathcal{Y}_{meta-train}^i)$  for  $i = 1, \dots, N$ , where  $\mathcal{X}_{meta-train}^i \subset \mathbb{R}^{p_i}$  is a  $p_i$  dimensional feature space. Using  $\mathbf{D}_{meta-train}$  we will define the meta-learned portfolio  $\Lambda_T$ . Next to the meta-train, we get a meta-test task, the dataset not known before for which we want to solve the Eq. 2. The meta-test task is sampled from the distribution  $\mathcal{D}_{meta-test} = (\mathcal{X}_{meta-test}, \mathcal{Y}_{meta-test})$ , which differs from the meta-train distribution. Let  $P(\mathcal{X}_{meta-train}^i), P(\mathcal{X}_{meta-test})$  be marginal probability distributions for  $i$ -th meta-train prediction problem and meta-test, respectively. Generally, every meta-train distribution  $\mathcal{X}_{meta-train}^i$  and  $\mathcal{X}_{meta-test}$  is defined independently, and we do not assume any relationship between them.

As OpenML is a major source of prediction tasks, various unrelated datasets are used. For instance, one of the meta-train tasks was wdbc describing the prediction of breast cancer. It consists of numerical features extracted from imaging diagnostics, and the meta-test is spambase which uses word frequency statistics to assess whether a mail belongs to

spam. Every feature may come from markedly different sources and distributions. However, even this kind of meta-train selection ensures an improvement in the optimization strategy, especially in terms of anytime performance. To benefit from the advantages of such a guided portfolio of hyperparameters, in this work, we focus more on the relationships between meta-train and meta-test.

### 3.3 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.

#### 3.3.1 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 (Selby, 1997; Hibbard et al., 2010; Moorman et al., 2013; Roth et al., 2014; Casey et al., 2016). 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 Kumar et al. (2021), 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 (Brierley et al., 2019). 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 (Karelson et al., 1996); 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 (Gaulton et al., 2012), we can find collections of diverse tasks used in the meta-learning approach in Olier et al. (2018). On the other hand, in high-energy physics, machine learning is used in particle detection from collected data at the Large Hadron Collider (LHC) (Carleo et al., 2019).

- *Related targets.* Many prediction problems do not have a clearly defined endpoint, and several different correlated target definitions are examined. In Simonov et al. (2019), in addition to the occurrence of acute kidney injury (AKI) within 24 h 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 (Purushotham et al., 2018; Sadeghi et al., 2018). From a logistical point of view, predicting the length of stay in hospital is also an important issue (Purushotham et al., 2018; Turgeman et al., 2017). 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 Koyner et al. (2018) 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 (Celik & Vanschoren, 2021).

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.

### 3.3.2 Formalization

We formalize the *consolidated meta-train* and *consolidated learning* using the terminology from Sect. 3.2. Restricting meta-train tasks to the representative for specific domain results in dependency between  $\mathcal{X}_{meta-train}^i$  and  $\mathcal{X}_{meta-test}$  for some  $i = 1, \dots, N$ . 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. We denote this situation as  $P(\mathcal{X}_{meta-train}^i) \cap P(\mathcal{X}_{meta-train}^j) \neq \emptyset$  for  $i \neq j$  or  $P(\mathcal{X}_{meta-train}^i) \cap P(\mathcal{X}_{meta-test}) \neq \emptyset$ . If the features set is identical we denote this by  $P(\mathcal{X}_{meta-train}^i) \equiv P(\mathcal{X}_{meta-test})$ . 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.

## 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 (Johnson et al., 2020) 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 (Nemati et al., 2018; Zhang et al., 2019; Meng et al., 2022; Liu et al., 2021). We employ this collection as a benchmark for evaluating hyperparameter transfer in *consolidated learning* and assessing the improvement in tuning.

### 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 (Johnson et al., 2017; Meng et al., 2022; Purushotham et al., 2018). 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, 34,925 unique patients met all the above conditions.

### 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 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).

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

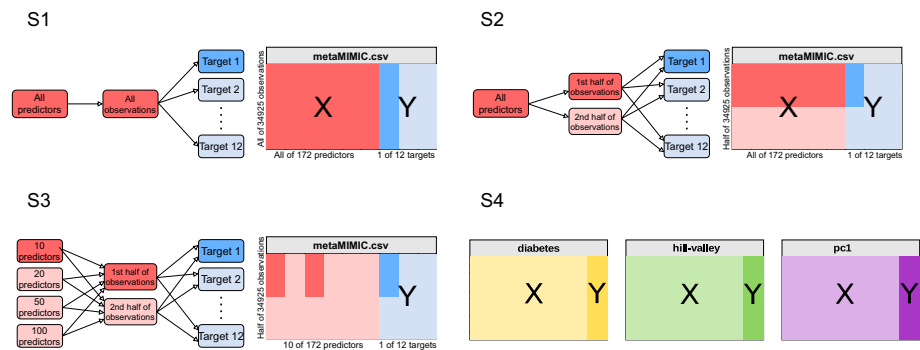
**Table 1** Selected targets with corresponding ICD codes and frequency in the considered cohort

Condition	ICD-9	ICD-10	Frequency (%)
Hypertensive diseases	401–405	I10–I16	59.8
Disorders of lipid 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 hyperalimentionation	278	E65–E68	10.5
Alcohol dependence	303	F10	7.7

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.

### 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.



**Fig. 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

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 Fig. 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.

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  $\mathcal{X}_{meta-train}^i \equiv \mathcal{X}_{meta-test}$  for every  $i = 1, \dots, N$ . Therefore, the only real choice to make is to select which target to predict (Fig. 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 samples observations but using the same set of variables. To avoid leakage of information occurring in S1, we consider two random, disjoint samples of observations (Fig. 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 (Fig. 2 S3). The choice of predictor set was realized by selecting top  $n$  variables with the highest permutation variable importance value (Breiman, 2001; Fisher et al., 2019), 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.

## 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 (Shwartz-Ziv & Armon, 2022). 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 (Probst et al., 2019), we validate the potential of *consolidated learning* for this algorithm.

## 5.1 Hyperparameter grid

As the hyperparameter search space, we use the grid from the MementoML study (Kretowicz & Biecek, 2020) 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 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 (Wistuba et al., 2015, 2016) 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.

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 MementoML. 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.

## 5.2 Tuning strategies

We test four hyperparameter tuning methods for the XGBoost algorithm. Two of them are meta-learning based, so they use the model performance results for other meta-train sets.

**Table 2** Hyperparameters and their underlying distributions

	Hyperparameter	Type	Lower	Upper	Distribution
*	<code>n_estimators</code>	Integer	1	1000	U
*	<code>learning_rate</code>	Float	0.031	1	$2^U$
*	<code>booster</code>	Discrete	–	–	{ <code>gblinear</code> , <code>gbtree</code> }
	<code>subsample</code>	Float	0.5	1	U
	<code>max_depth</code>	Integer	6	15	U
	<code>min_child_weight</code>	Float	1	8	$2^U$
	<code>colsample_bytree</code>	Float	0.2	1	U
	<code>colsample_bylevel</code>	Float	0.2	1	U

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`

For each scenario, the following strategies are tested: the transfer is performed with meta-MIMIC (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) (Wistuba et al., 2016) 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) (Brazdil & Soares, 2000) 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 Sect. 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.

### 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 (Wistuba et al., 2015).

Let  $\mathbf{D}_{meta-test}$  be the collection of meta-test datasets for which tuning strategy is evaluated, and  $f_i$  be AUC measure for  $i$ -th dataset from  $\mathbf{D}_{meta-test}$ . The portfolio of hyperparameter configuration proposed by strategy in iteration  $T$  is  $\Lambda_T$ . Then ADTM is defined as

$$ADTM(\mathbf{D}_{meta-test}, \Lambda_T) = \frac{1}{|\mathbf{D}_{meta-test}|} \sum_{i \in \mathbf{D}_{meta-test}} \min_{\lambda \in \Lambda_T} \frac{f_i^{max} - f_i(\lambda)}{f_i^{max} - f_i^{min}}, \quad (3)$$

where  $|\mathbf{D}_{meta-test}|$  is the cardinality of meta-test set. 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.

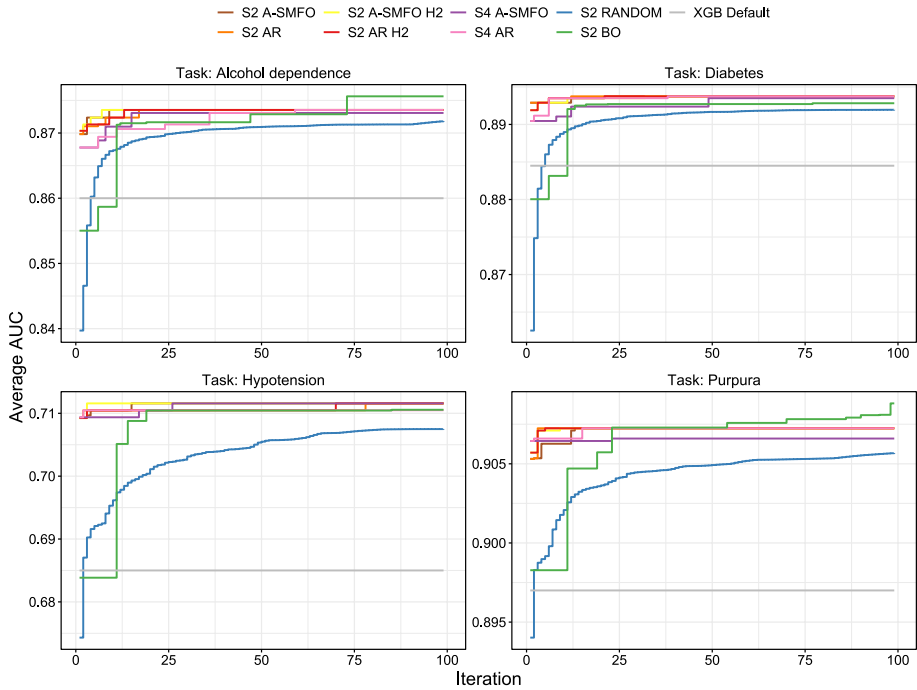
## 6 Effectiveness of consolidated learning

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.

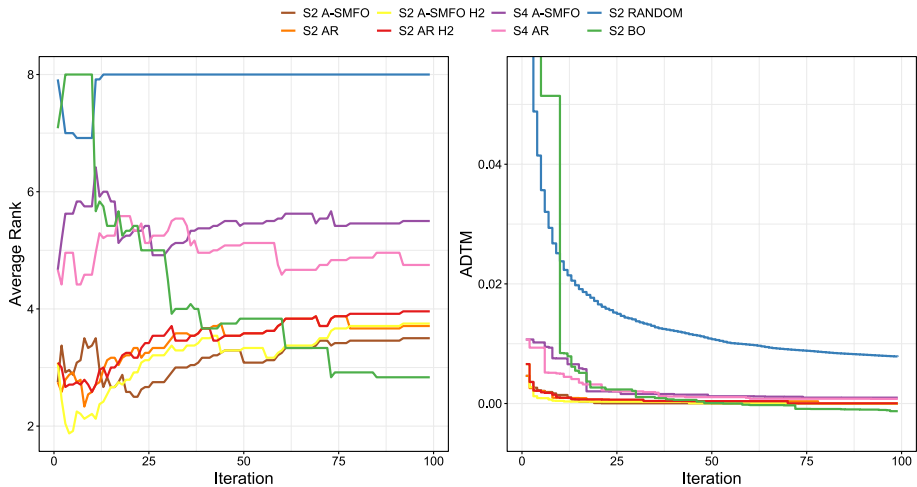
Figure 3 shows the results for four selected meta-test targets. Looking at the model performance during hyperparameter tuning, we see that methods based on meta-learning provide configurations close to the observed maximum already in the first iteration. The distance between the learning curves for scenarios 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 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 (Fig. 4). We see that



**Fig. 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



**Fig. 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)

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 subsample 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 Fig. 4 we see how fast the tuning strategies converge 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.

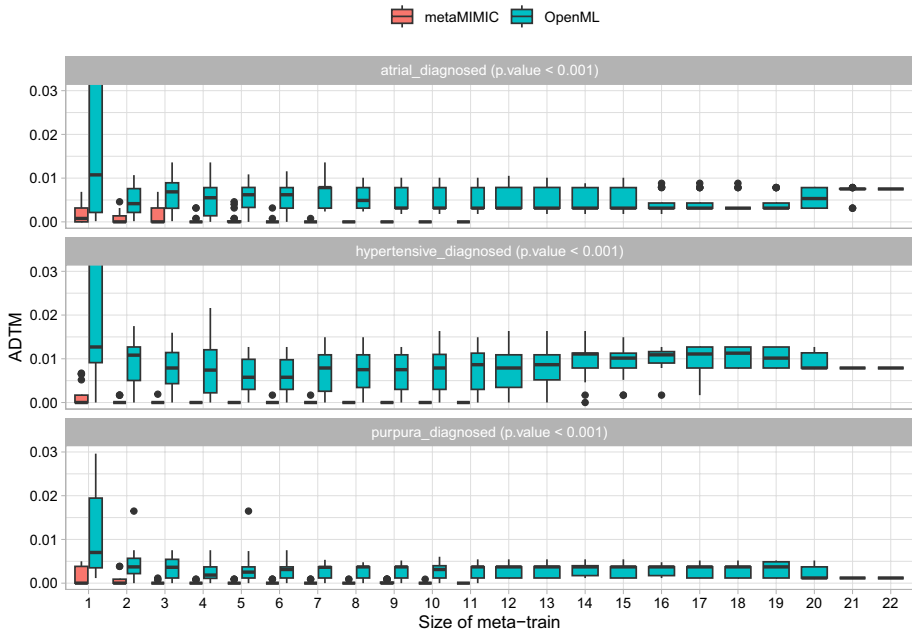
Thus, we can conclude that meta-learning is effective and even using unrelated datasets allows us to reject unsatisfactory configurations and provide a decent model performance for several trials in tuning. We can accelerate the tuning by employing *consolidated learning*. Random search and Bayesian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

## 6.1 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 portfolio 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 Fig. 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 *consolidated 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.





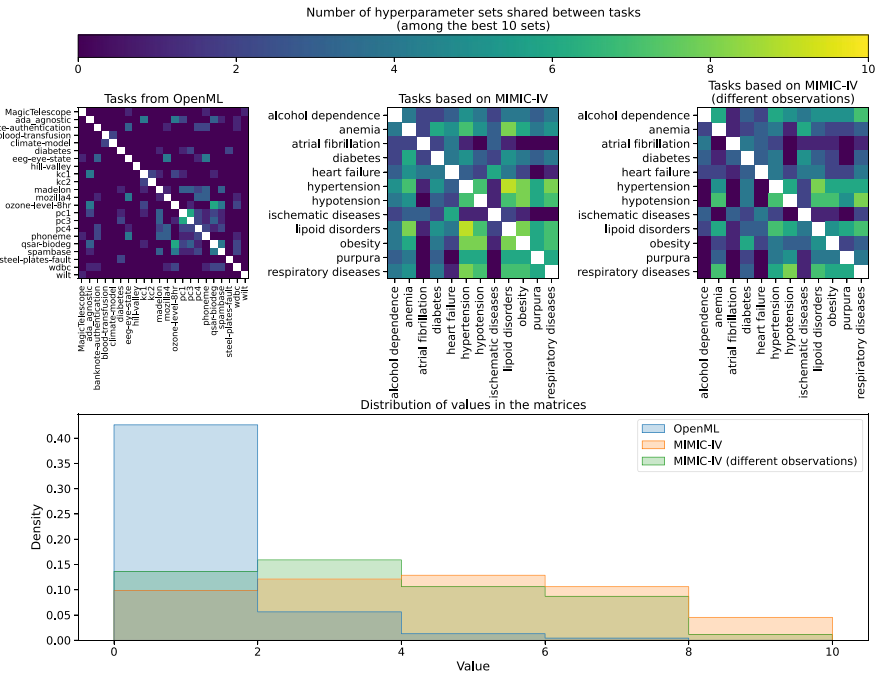
**Fig. 5** Transferability of hyperparameters changes with the number of available datasets in meta-train. For every task we modify the size of meta-train in Scenarios S2 and S4 and check the distribution of the ADTM values after 10 optimization iterations. Grey headings indicate target from metaMIMIC 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

## 7 Robustness of transferability

In Sect. 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.

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 (Fig. 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 ( $0.165 \pm 0.469$  for S4,  $0.885 \pm 0.072$  for S1, and  $0.849 \pm 0.078$  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 right-most 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.



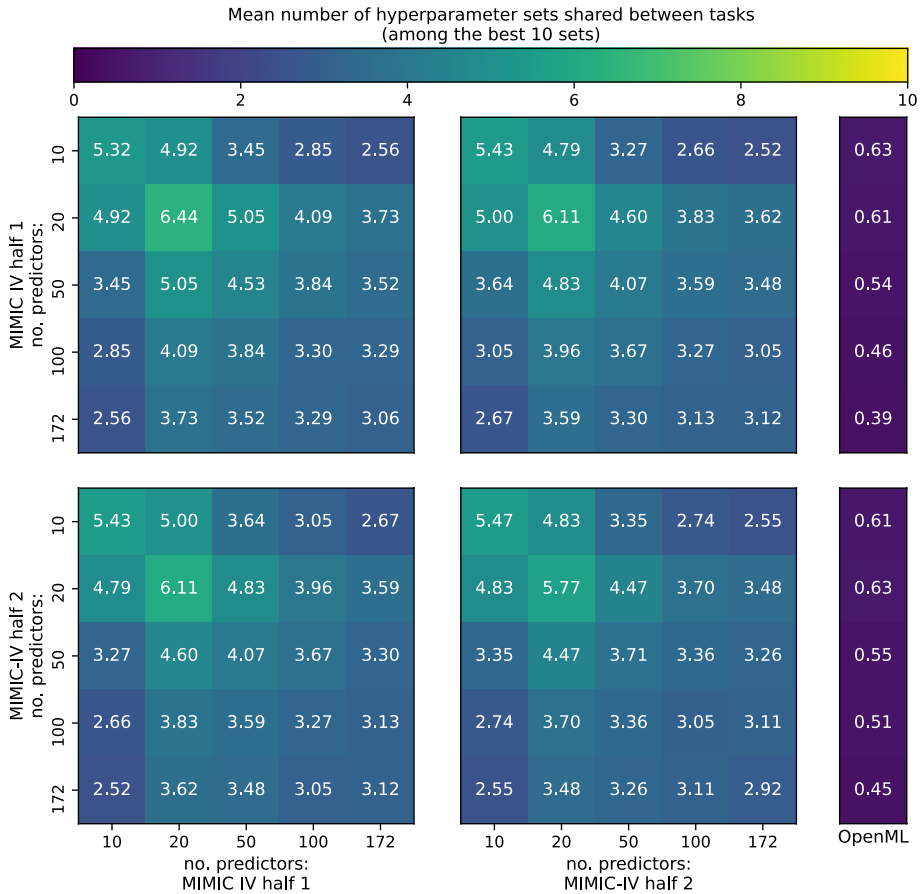
**Fig. 6** Numbers of the best 10 hyperparameter sets (regarding the mean 4-CV ROC AUC measure) shared between tasks from S4, S1, and S2. An individual cell of the matrix corresponds to the number of hyperparameter sets shared between a given pair of tasks. Histograms summarize the distribution of the values of each matrix. White color on the diagonal means that the value is not considered

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 (Fig. 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 Sect. 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.

### 8 Conclusions

The results presented in this work 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



**Fig. 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

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 (Jomaa et al., 2021). 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 (Wang et al., 2002).

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>.

**Acknowledgements** The work on this paper is financially supported by the NCN Opus grant NCN Sonata Bis-9 grant 2019/34/E/ST6/00052. We would like to thank Mikołaj Spytek, Anna Kozak, Hubert Baniecki for their useful comments.

**Author contributions** All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Katarzyna Woźnica, Mateusz Grzyb and Zuzanna Trafas. The original draft of the manuscript was written by Katarzyna Woźnica and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript. Przemysław Biecek was supervisor of the core team.

**Funding** The work on this paper is financially supported by the NCN Sonata Bis-9 grant 2019/34/E/ST6/00052.

**Data availability** The MIMIC-IV database is available on <https://physionet.org/content/mimiciv/0.4/> The code needed to reproduce the metaMIMIC: <https://github.com/ModelOriented/metaMIMIC>

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

## Declarations

**Conflict of interest** The authors have no conflict of interest to declare that are relevant to the content of this article.

**Ethics approval** Not applicable.

**Consent to participate** Not applicable.

**Consent for publication** Not applicable.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Alaa, A., & Schaar, M. (2018). AutoPrognosis: Automated clinical prognostic modeling via bayesian optimization with structured kernel learning. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 139–148).
- Alibrahim, H., & Ludwig, S. A. (2021). Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)* (pp. 1551–1559). <https://doi.org/10.1109/CEC45853.2021.9504761>
- Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24.
- Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10), 281–305.
- Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2015). Hyperopt: A python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1), 13–19. <https://doi.org/10.1088/1749-4699/8/1/014008>
- Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., & Vanschoren, J. (2017). OpenML benchmarking suites and the OpenML100. *stat* 1050, 11.
- Bouthillier, X., & Varoquaux, G. (2020). Survey of machine-learning experimental methods at NeurIPS2019 and ICLR2020. PhD thesis, Inria Saclay Ile de France.
- Brazdil, P. B., & Soares, C. (2000). A comparison of ranking methods for classification algorithm selection. In *Proceedings of the 11th European conference on machine learning (ECML)* (pp. 63–75). Springer. [https://doi.org/10.1007/3-540-45164-1\\_8](https://doi.org/10.1007/3-540-45164-1_8)
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- Brierley, J., O'Sullivan, B., Asamura, H., Byrd, D., Huang, S. H., Lee, A., Piñeros, M., Mason, M., Moraes, F. Y., Rösler, W., et al. (2019). Global consultation on cancer staging: Promoting consistent understanding and use. *Nature Reviews Clinical Oncology*, 16(12), 763–771.


- Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., & Zdeborová, L. (2019). Machine learning and the physical sciences. *Reviews of Modern Physics*, *91*(4), 045002.
- Casey, J. A., Schwartz, B. S., Stewart, W. F., & Adler, N. E. (2016). Using electronic health records for population health research: A review of methods and applications. *Annual Review of Public Health*, *37*, 61–81.
- Celik, B., & Vanschoren, J. (2021). Adaptation strategies for automated machine learning on evolving data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(9), 3067–3078.
- Chen, T., Guestrin, C. (2016) XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794).
- Davis, C., & Giraud-Carrier, C. (2018). Annotative experts for hyperparameter selection. In *AutoML workshop at ICML*.
- Edwards, H., & Storkey, A. (2017). Towards a neural statistician. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.
- Escalante, H. J., Montes, M., & Sucar, E. (2010). Ensemble particle swarm model selection. In *The 2010 international joint conference on neural networks (IJCNN)* (pp. 1–8).
- Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th international conference on machine learning (ICML)* (pp. 1437–1446).
- Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., Hutter, F. (2019). In: F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), *Auto-sklearn: Efficient and robust automated machine learning* (pp. 113–134). Springer. [https://doi.org/10.1007/978-3-030-05318-5\\_6](https://doi.org/10.1007/978-3-030-05318-5_6)
- Feurer, M., Springenberg, J., & Hutter, F. (2015). Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the 29th AAAI conference on artificial intelligence* (pp. 1128–1135).
- Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2022). Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*, *23*(261), 1–61.
- Fisher, A., Rudin, C., & Dominici, F. (2019). 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.
- Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., et al. (2012). ChEMBL: A large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, *40*(D1), 1100–1107.
- Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., Statnikov, A., Tu, W.-W., Viegas, E. (2019). In F. Hutter, L. Kotthoff, & Vanschoren, J. (Eds.), *Analysis of the AutoML challenge series 2015–2018* (pp. 177–219). Springer. [https://doi.org/10.1007/978-3-030-05318-5\\_10](https://doi.org/10.1007/978-3-030-05318-5_10)
- Hewitt, L. B., Nye, M. I., Gane, A., Jaakkola, T. S., & Tenenbaum, J. B. (2018). The variational homocoder: Learning to learn high capacity generative models from few examples. Preprint [arxiv:1807.08919](https://arxiv.org/abs/1807.08919)
- Hibbard, J. U., Wilkins, I., Sun, L., Gregory, K., Haberman, S., Hoffman, M., Kominiarek, M. A., Reddy, U., Bailit, J., Branch, D. W., et al. (2010). Respiratory morbidity in late preterm births. *JAMA: The Journal of the American Medical Association*, *304*(4), 419.
- Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). *Sequential model-based optimization for general algorithm configuration* Lecture Notes in Computer Science (pp. 507–523). Springer. [https://doi.org/10.1007/978-3-642-25566-3\\_40](https://doi.org/10.1007/978-3-642-25566-3_40)
- Jamieson, K., & Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th international conference on artificial intelligence and statistics (AISTATS)* (pp. 240–248).
- Johnson, A., Bulgarelli, L., Pollard, T., Horng, S., Celi, L., & Mark, R. (2020). MIMIC-IV (version 1.0). <https://doi.org/10.13026/a3wn-hq05>
- Johnson, A. E. W., Pollard, T. J., & Mark, R. G. (2017). Reproducibility in critical care: A mortality prediction case study. In *Proceedings of the 2nd machine learning for health care conference (MLHC)* (pp. 361–376).
- Jomaa, H. S., Schmidt-Thieme, L., & Grabocka, J. (2021). Dataset2vec: Learning dataset meta-features. *Data Mining and Knowledge Discovery*, *35*(3), 964–985. <https://doi.org/10.1007/s10618-021-00737-9>
- Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, *13*(4), 455–492. <https://doi.org/10.1023/A:1008306431147>
- Karelson, M., Lobanov, V. S., & Katritzky, A. R. (1996). Quantum-chemical descriptors in QSAR/QSPR studies. *Chemical Reviews*, *96*(3), 1027–1044.
- Koyner, J. L., Carey, K. A., Edelson, D. P., & Churpek, M. M. (2018). The development of a machine learning inpatient acute kidney injury prediction model. *Critical Care Medicine*, *46*(7), 1070–1077. <https://doi.org/10.1097/CCM.0000000000003123>

- Kretowicz, W., & Biecek, P. (2020). MementoML: Performance of selected machine learning algorithm configurations on OpenML100 datasets. Preprint [arXiv:2008.13162](https://arxiv.org/abs/2008.13162)
- Kumar, S., Oh, I., Schindler, S., Lai, A. M., Payne, P. R., & Gupta, A. (2021). Machine learning for modeling the progression of Alzheimer disease dementia using clinical data: A systematic literature review. *JAMIA Open*, 4(3), 052. <https://doi.org/10.1093/jamiaopen/ooab052>
- Lavesson, N., & Davidsson, P. (2006). Quantifying the impact of learning algorithm parameter tuning. In *Proceedings of the 21st AAAI conference on artificial intelligence* (Vol. 6, pp. 395–400).
- Li, K., & Malik, J. (2017). Learning to optimize. In *International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=ry4Vrt5gl>
- Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.
- Liu, T., Zhao, Q., & Du, B. (2021). 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. <https://doi.org/10.1186/s12890-021-01526-2>
- Mantovani, R. G., Rossi, A. L. D., Alcobaca, E., Gertrudes, J. C., Junior, S. B., & de Carvalho, A. C. P. D. L. F. (2020). Rethinking default values: A low cost and efficient strategy to define hyperparameters. arXiv preprint [arXiv:2008.00025](https://arxiv.org/abs/2008.00025)
- Meng, C., Trinh, L., Xu, N., Enouen, J., & Liu, Y. (2022). Interpretability and fairness evaluation of deep learning models on MIMIC-IV dataset. *Scientific Reports*, 12(1), 7166. <https://doi.org/10.1038/s41598-022-11012-2>
- Moorman, A. C., Gordon, S. C., Rupp, L. B., Spradling, P. R., Teshale, E. H., Lu, M., Nerenz, D. R., Nakasato, C. C., Boscarino, J. A., Henkle, E. M., et al. (2013). Baseline characteristics and mortality among people in care for chronic viral hepatitis: The chronic hepatitis cohort study. *Clinical Infectious Diseases*, 56(1), 40–50.
- Nemati, S., Holder, A., Razmi, F., Stanley, M. D., Clifford, G. D., & Buchman, T. G. (2018). An interpretable machine learning model for accurate prediction of sepsis in the ICU. *Critical Care Medicine*, 46(4), 547–553. <https://doi.org/10.1097/CCM.0000000000002936>
- Oh, C., Gavves, E., & Welling, M. (2018). BOCK: Bayesian optimization with cylindrical kernels. In *Proceedings of the 35th international conference on machine learning (ICML)* (pp. 3868–3877).
- Olier, I., Sadawi, N., Bickerton, G. R., Vanschoren, J., Grosan, C., Soldatova, L., & King, R. D. (2018). Meta-QSAR: A large-scale application of meta-learning to drug design and discovery. *Machine Learning*, 107(1), 285–311.
- Olson, R. S., & Moore, J. H. (2019). In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *TPOT: A tree-based pipeline optimization tool for automating machine learning* (pp. 151–160). Springer. [https://doi.org/10.1007/978-3-030-05318-5\\_8](https://doi.org/10.1007/978-3-030-05318-5_8)
- Perrone, V., Shen, H., Seeger, M. W., Archambeau, C., & Jenatton, R. (2019). Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. In *Advances in neural information processing systems*.
- Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. G. (2000). Meta-learning by landmarking various learning algorithms. In *Proceedings of the 17th international conference on machine learning (ICML)* (pp. 743–750).
- Pfisterer, F., van Rijn, J. N., Probst, P., Müller, A. C., & Bischl, B. (2021). Learning multiple defaults for machine learning algorithms. In *Proceedings of the genetic and evolutionary computation conference companion (GECCO)* (pp. 241–242). <https://doi.org/10.1145/3449726.3459523>
- Probst, P., Boulesteix, A.-L., & Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20(53), 1–32.
- Purushotham, S., Meng, C., Che, Z., & Liu, Y. (2018). Benchmarking deep learning models on large healthcare datasets. *Journal of Biomedical Informatics*, 83, 112–134.
- Purushotham, S., Meng, C., Che, Z., & Liu, Y. (2018). Benchmarking deep learning models on large healthcare datasets. *Journal of Biomedical Informatics*, 83, 112–134.
- Reif, M., Shafait, F., Goldstein, M., Breuel, T., & Dengel, A. (2014). Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, 17(1), 83–96. <https://doi.org/10.1007/s10044-012-0280-z>
- Rivoli, A., Garcia, L. P., Soares, C., Vanschoren, J., & de Carvalho, A. C. (2022). Meta-features for meta-learning. *Knowledge-Based Systems*, 240, 108101. <https://doi.org/10.1016/j.knsys.2021.108101>
- Roth, C., Foraker, R. E., Payne, P. R., & Embi, P. J. (2014). 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.
- Sadeghi, R., Banerjee, T., & Romine, W. (2018). Early hospital mortality prediction using vital signals. *Smart Health*, 9, 265–274.

- Selby, J. V. (1997). Linking automated databases for research in managed care settings. *Annals of Internal Medicine*, 127(82), 719–724.
- Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81, 84–90.
- Simonov, M., Ugwuowo, U., Moreira, E., Yamamoto, Y., Biswas, A., Martin, M., Testani, J., & Wilson, F. P. (2019). A simple real-time model for predicting acute kidney injury in hospitalized patients in the us: A descriptive modeling study. *PLoS Medicine*, 16(7), 1002861.
- Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25.
- Souza, A., Nardi, L., Oliveira, L., Olukotun, K., Lindauer, M., & Hutter, F. (2021). Bayesian Optimization with a prior for the optimum (pp. 265–296). [https://doi.org/10.1007/978-3-030-86523-8\\_17](https://doi.org/10.1007/978-3-030-86523-8_17)
- Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 847–855). <https://doi.org/10.1145/2487575.2487629>
- Turgeman, L., May, J. H., & Scullli, R. (2017). 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.
- Vakhrushev, A., Ryzhkov, A., Savchenko, M., Simakov, D., Daminov, R., & Tuzhilin, A. (2021). LightAutoML: AutoML solution for a large financial services ecosystem. Preprint [arXiv:2109.01528](https://arxiv.org/abs/2109.01528)
- Vanschoren, J. (2019). In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Meta-learning* (pp. 35–61). Springer. [https://doi.org/10.1007/978-3-030-05318-5\\_6](https://doi.org/10.1007/978-3-030-05318-5_6)
- Vilalta, R., Giraud-Carrier, C., Brazdil, P., & Soares, C. (2004). Using meta-learning to support data mining. *International Journal of Computer Science & Applications*, 1, 31–45.
- Wang, A. Y., Sable, J. H., & Spackman, K. A. (2002). The SNOMED clinical terms development process: Refinement and analysis of content. In *Proceedings of the AMIA symposium* (p. 845). American Medical Informatics Association.
- Winkelmolen, F., Ivkin, N., Bozkurt, H. F., & Karnin, Z. (2020). Practical and sample efficient zero-shot hpo. arXiv preprint [arXiv:2007.13382](https://arxiv.org/abs/2007.13382)
- Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2015). Learning hyperparameter optimization initializations. In *Proceedings in the IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 1–10). <https://doi.org/10.1109/dsaa.2015.7344817>
- Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2016). Sequential model-free hyperparameter tuning. In *Proceedings in the IEEE International Conference on Data Mining (ICDM)* (pp. 1033–1038). <https://doi.org/10.1109/ICDM.2015.20>
- Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2018). Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1), 43–78. <https://doi.org/10.1007/s10994-017-5684-y>
- Woźnica, K., & Biecek, P. (2021). Towards explainable meta-learning. In: Kamp, M., et al. *Machine Learning and Principles and Practice of Knowledge Discovery in Databases. ECML PKDD 2021. Communications in Computer and Information Science*, vol 1524. Springer. [https://doi.org/10.1007/978-3-030-93736-2\\_38](https://doi.org/10.1007/978-3-030-93736-2_38)
- Zhang, Z., Ho, K. M., & Hong, Y. (2019). Machine learning for the prediction of volume responsiveness in patients with oliguric acute kidney injury in critical care. *Critical Care*, 23(1), 1–10. <https://doi.org/10.1186/s13054-019-2411-z>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Authors and Affiliations

Katarzyna Woźnica<sup>1</sup>  · Mateusz Grzyb<sup>1</sup> · Zuzanna Trafas<sup>2</sup> · Przemysław Biecek<sup>1,3</sup>

✉ Katarzyna Woźnica  
katarzyna.woznica.dokt@pw.edu.pl; woznickatarzyna22@gmail.com

Mateusz Grzyb  
mateusz.jakub.grzyb@gmail.com



Zuzanna Trafas  
zuz.trafas@gmail.com

Przemysław Biecek  
przemyslaw.biecek@pw.edu.pl

- <sup>1</sup> Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Masovian Voivodeship, Poland
- <sup>2</sup> Faculty of Computing and Telecommunications, Poznan University of Technology, Piotrowo 3A, 60-965 Poznan, Greater Poland Voivodeship, Poland
- <sup>3</sup> Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw, Banacha 2, 02-097 Warsaw, Masovian Voivodeship, Poland