PMLBmini: A Tabular Classification Benchmark Suite for Data-Scarce Applications

Ricardo Knauer[1] **Marvin Grimm**[1] **Erik Rodner**[1]

[1]KI-Werkstatt, University of Applied Sciences Berlin, Germany

Abstract In practice, we are often faced with small-sized tabular data. However, current tabular benchmarks are not geared towards data-scarce applications, making it very difficult to derive meaningful conclusions from empirical comparisons. We introduce PMLBmini, a tabular benchmark suite of 44 binary classification datasets with sample sizes ≤ 500. We use our suite to thoroughly evaluate current automated machine learning (AutoML) frameworks, off-the-shelf tabular deep neural networks, as well as classical linear models in the low-data regime. Our analysis reveals that state-of-the-art AutoML and deep learning approaches often fail to appreciably outperform even a simple logistic regression baseline, but we also identify scenarios where AutoML and deep learning methods are indeed reasonable to apply. Our benchmark suite, available on `https://github.com/RicardoKnauer/TabMini`, allows researchers and practitioners to analyze their own methods and challenge their data efficiency.

1 Introduction

The easy access to data has fueled machine learning research in recent years. Massive text corpora crawled from the web have given rise to large language models such as GPT-3 with emergent abilities such as in-context learning (Brown et al., 2020). Large-scale image data have served as the foundation for text-to-image systems like DALL-E 3 (Betker et al., 2023), large-scale video data for text-to-video generators like Sora (Brooks et al., 2024). In contrast to text, image, or video data, collecting large- or even medium-sized tabular data is often challenging in practice, despite tabular data being among the most ubiquitous dataset types in real-world applications (Borisov et al., 2022; McElfresh et al., 2024; Shwartz-Ziv and Armon, 2022). GitTables, for example, a curated corpus of 1 million tables from GitHub, only contains 142 instances on average (Hulsebos et al., 2023). In clinical diagnostic or prognostic settings, the number of instances is frequently limited due to the rareness of medical conditions or patient losses at follow-up assessments, respectively (Moons et al., 2015, 2019; Steyerberg, 2019).

The difficulty to acquire large- or even medium-scale tabular datasets in many domains presents researchers and practitioners with a unique set of challenges. On the one hand, overfitting is a major concern when applying complex algorithms on small-sized datasets. On the other hand, cross-validation folds may become too small to adequately represent both the original sample and the population of interest. This makes it very difficult to find good hyperparameter settings so that data-driven hyperparameter optimization may fail to increase, or may even decrease, the predictive performance for individual datasets in the low-data regime (Riley et al., 2021; Šinkovec et al., 2021; Van Calster et al., 2020). To facilitate empirical comparisons across studies in this setting, it is therefore imperative to systematically evaluate machine learning pipelines not on a hand-picked, narrow selection of datasets, but on a standardized, diverse dataset collection - a benchmark suite (Bischl et al., 2021; Fischer et al., 2023; Gijsbers et al., 2019, 2022; McElfresh et al., 2024; Olson et al., 2017; Romano et al., 2022).

In this paper, we contribute to the tabular benchmarking literature in the following ways:

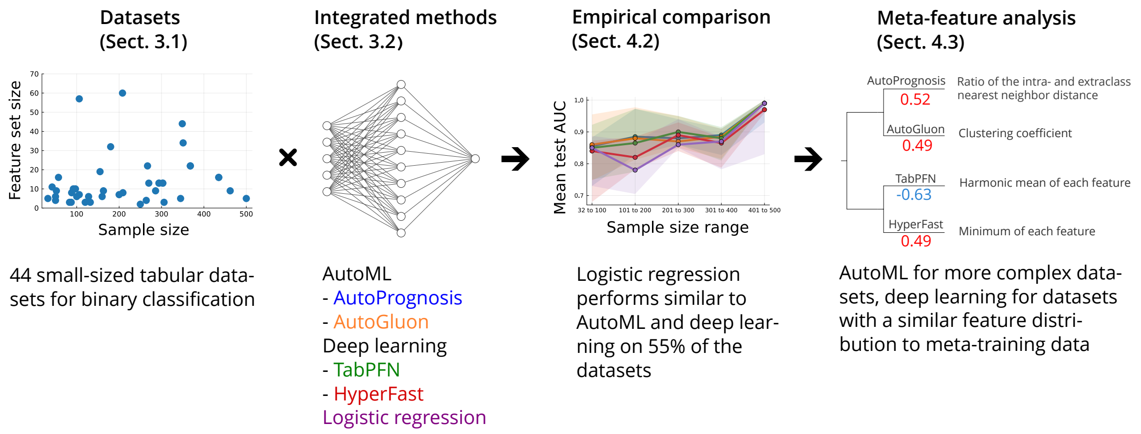

Datasets (Sect. 3.1)
Feature set size
Sample size
44 small-sized tabular data-sets for binary classification
Integrated methods (Sect. 3.2)
AutoML
- AutoPrognosis
- AutoGluon
Deep learning
- TabPFN
- HyperFast
Logistic regression
Empirical comparison (Sect. 4.2)
Mean test AUC
Sample size range
32 to 100
101 to 200
201 to 300
301 to 400
401 to 500
Logistic regression performs similar to AutoML and deep lear-ning on 55% of the datasets
Meta-feature analysis (Sect. 4.3)
AutoPrognosis 0.52
Ratio of the intra- and extraclass nearest neighbor distance
AutoGluon 0.49
Clustering coefficient
TabPFN -0.63
Harmonic mean of each feature
HyperFast 0.49
Minimum of each feature
AutoML for more complex data-sets, deep learning for datasets with a similar feature distri-bution to meta-training data

Figure 1: Overview of our work on PMLBmini, the first tabular classification benchmark suite specifically for data-scarce applications.

1. We conduct a **narrative review on tabular benchmark suites for data-scarce applications** with sample sizes ≤ 500, and find that (very-)small-sized datasets are generally underrepresented in current tabular benchmarks (Sect. 2).

2. We introduce **PMLBmini, the first tabular benchmark suite specifically for the low-data regime with 44 binary classification datasets (Sect. 3.1), and use our suite to compare state-of-the-art machine learning methods, i.e., automated machine learning (AutoML) frameworks and off-the-shelf deep neural networks, against logistic regression (Sect. 3.2)**. Overall, we show that logistic regression performs similar to AutoML and deep learning approaches in terms of discrimination on 55% of the datasets, and present the best L2-regularization hyperparameter obtained on each dataset that can be used for meta-learning in data-scarce applications (Sect. 4.2). We also conduct an extensive meta-feature analysis to assess under which conditions, i.e., dataset properties, AutoML and deep learning methods outperform a logistic regression baseline (Sect. 4.3). Please refer to Fig. 1 for an overview of our work.

3. We **release PMLBmini for researchers and practitioners to benchmark their own tabular classifier, and to analyze if and when it succeeds or fails**. This way, we aim to equip the community with an easily accessible, practical set of tools for empirical evaluations in the low-data regime (Sect. 3.3).

2 Related Work and Desiderata

Machine learning repositories for tabular data abound, prominent examples being Kaggle, UCI, or OpenML (Vanschoren et al., 2014), which form the basis for a wide range of carefully curated tabular benchmark suites (Bischl et al., 2021; Fischer et al., 2023; Gijsbers et al., 2019, 2022; McElfresh et al., 2024; Olson et al., 2017; Romano et al., 2022). In spite of the practical relevance and unique challenges in the low-data regime (Sect. 1), small-sized datasets with sample sizes ≤ 500 have received very little attention in benchmarking studies so far, though (Fischer et al., 2023; Gijsbers et al., 2019). The suites that do include ≥ 1 small-sized tabular benchmark dataset are the Penn Machine Learning Benchmarks (PMLB) (Olson et al., 2017; Romano et al., 2022), the AutoML Benchmark (AMLB) (Gijsbers et al., 2022), the OpenML-CC18 (Bischl et al., 2021), and TabZilla (McElfresh et al., 2024). Their respective dataset sources, task types, sample size range, number of included datasets, and number of included datasets with sample sizes ≤ 500 are shown in Table 1. AMLB, OpenML-CC18, and TabZilla include an insufficient number of small-sized datasets to allow

Table 1: Benchmark suites that include small-sized tabular datasets

Benchmark	**Dataset sources**	**Task types**	**Sample size range**	**# Datasets**	**# Small-sized datasets**
PMLB (Olson et al., 2017; Romano et al., 2022)	Kaggle, UCI, OpenML, and others	Classification, regression	32 to over 1 million	419	158
AMLB (Gijsbers et al., 2022)	OpenML	Classification, regression	100 to 10 million	104	2
OpenML-CC18 (Bischl et al., 2021)	OpenML	Classification	500 to 96320	72	1
TabZilla (McElfresh et al., 2024)	OpenML	Classification	148 to over 1 million	36	4
PMLBmini (ours)	OpenML	Classification	32 to 500	44	44

for meaningful comparisons in the low-data regime. PMLB, on the other hand, provides no interface for users to easily evaluate their own machine learning pipeline against a range of simple and state-of-the-art baselines on a preselected collection of small-sized datasets. We therefore formulate our desiderata for a tabular classification benchmark suite for data-scarce applications as follows:

- **Dataset size**: The collection should include only small-sized datasets with sample sizes ≤ 500, unlike any other tabular benchmark suite (Bischl et al., 2021; Fischer et al., 2023; Gijsbers et al., 2019, 2022; McElfresh et al., 2024; Olson et al., 2017; Romano et al., 2022). A cut-off at a sample size of 500 allows us to assess machine learning methods where the OpenML-CC18 leaves off (Bischl et al., 2021) and extend prior evaluations on this benchmark to the low-data regime (Bonet et al., 2024; Hollmann et al., 2023; Müller et al., 2023).

- **Dataset complexity**: To allow for assessing if and when state-of-the-art machine learning methods perform better than simple baselines, the suite should include both more and less complex datasets, and not exclude easy problems a priori (Bischl et al., 2021; Fischer et al., 2023; Gijsbers et al., 2019, 2022; McElfresh et al., 2024). In Sect. 4.2, we show that the inclusion of less difficult datasets does not prevent us from finding statistically significant performance differences between approaches when using our entire dataset collection.

- **Collection size**: The suite should be sufficiently large to allow for thorough empirical comparisons. Small-sized datasets are typically underrepresented in current tabular benchmarks (Bischl et al., 2021; Gijsbers et al., 2022; McElfresh et al., 2024) or not included at all (Fischer et al., 2023; Gijsbers et al., 2019).

As an additional preference, the suite should provide a user-friendly Python interface for researchers and practitioners to benchmark their own tabular classifier against baseline methods on the preselected dataset collection. Users should also be able to perform a comprehensive meta-feature analysis to find out under which conditions, i.e., dataset properties, their approach is better suited, which is not supported by most frameworks out of the box (Bischl et al., 2021; Fischer et al., 2023; Gijsbers et al., 2019; Olson et al., 2017; Romano et al., 2022). In the next section, we describe how we constructed our tabular classification benchmark suite for data-scarce applications based on our desiderata and preference.

3 Benchmark Design

In the following, we report on the design of our tabular classification benchmark, PMLBmini, starting with the included datasets. We then outline the baseline methods that we integrated into our suite, and finally provide details on how our benchmarking tool can be used for both in-depth empirical comparisons and meta-feature analyses in the low-data regime.

3.1 Datasets

We selected all binary classification datasets with sample sizes ≤ 500 from the curated benchmark suite with the largest number of small-sized datasets, PMLB (Sect. 2), resulting in 44 datasets (Table 3 in Appendix C). There was no sensitive personally identifiable information or offensive content in the selected datasets, there were no missing values, and categorical features were already numerically encoded in PMLB. This provided a simple, extensible basis for our tabular classification benchmark. We shuffled all datasets and encoded all labels to $\{0, 1\}$. The 2 binary classification datasets from the TabZilla benchmark (*colic* and *heart-h*) are already included in PMLB and therefore in our suite, the 2 binary classification datasets from AMLB and OpenML-CC18 (*arcene* and *dresses-sales*, respectively) are not. Both *arcene* and *dresses-sales* would be clear outliers in our collection, though; the former because its feature set size is roughly 6000% larger than the maximal feature set size in our suite, the latter because > 80% of its instances contain missing values. Even without the 2 excluded datasets from AMLB and OpenML-CC18, our final tabular classification benchmark suite, PMLBmini, includes more than 6 times more small-sized datasets than AMLB, OpenML-CC18, and TabZilla combined (Sect. 2). To demonstrate that our dataset selection is not just large in quantity, but also represents a diverse set of data science problems like PMLB (Olson et al., 2017; Romano et al., 2022), we summarize key dataset characteristics in Table 2 and plot the feature set size against the sample size for each included dataset in Fig. 4b in Appendix A, showing that our benchmark suite indeed covers a wide range of problem instances with a focus on smaller feature set and sample sizes.

3.2 Available Methods

Next to our preselected collection of small-sized tabular datasets, our suite also provides researchers and practitioners with a number of baseline methods. The baselines were chosen to represent a range of different machine learning approaches, but our suite can also be easily extended to include additional pipelines (Sect. 3.3). We integrated a simple L2-regularized logistic regression classifier (Knauer and Rodner, 2023), state-of-the-art automated machine learning (AutoML) frameworks (Alaa and van der Schaar, 2018; Imrie et al., 2023; Erickson et al., 2020; Salinas and Erickson, 2023), and recent off-the-shelf deep neural networks (Hollmann et al., 2023; Bonet et al., 2024). Some of these methods have already been shown to perform well when data is scarce (Christodoulou et al., 2019; Knauer and Rodner, 2023; McElfresh et al., 2024). An extensive comparison for sample sizes ≤ 500 is currently missing, though (Bonet et al., 2024; Hollmann et al., 2023; Müller et al., 2023; Puri et al., 2023). We offer the first comprehensive evaluation for these methods in the low-data regime in Sect. 4.2 and assess when state-of-the-art AutoML and deep learning approaches are better suited than a simple logistic regression baseline in this setting in Sect. 4.3.

Logistic regression We integrated an L2-regularized logistic regression classifier as a simple, transparent, intrinsically interpretable baseline. We use a continuous conic formulation that is designed to be run-to-run deterministic (MOSEK ApS, 2023) and can be easily extended to include cardinality or budget constraints for best subset selection (Deza and Atamtürk, 2022; Knauer and Rodner, 2023). We re-encode all labels to $\{-1, 1\}$ and "min-max" scale all features for logistic regression. The L2-regularization hyperparameter λ is tuned via a (nested) stratified, 3-fold cross-validation using $[0.5, 0.1, 0.02, 0.004]$ as the hyperparameter grid and the deviance as the validation score.

AutoML We focused on 2 recently updated AutoML frameworks for implementation into our suite, AutoPrognosis (Alaa and van der Schaar, 2018; Imrie et al., 2023) and AutoGluon (Erickson et al., 2020; Salinas and Erickson, 2023). AutoPrognosis considers logistic regression and decision tree ensembles to automatically build end-to-end machine learning pipelines, including preprocessing, model, and hyperparameter selection. Meta-learned hyperparameters from external datasets are used to initialize the default pipeline optimization procedure, and a pipeline ensemble is constructed following the search process. AutoGluon, on the other hand, considers k-nearest neighbors, decision tree ensembles, and neural networks for its algorithm search. It meta-learns a model hyperparameter portfolio from external datasets and different random seeds (zero-shot hyperparameter optimization); the training time budget is then spent on ensembling rather than further hyperparameter optimization. Unfortunately, we had to exclude another state-of-the-art framework, Auto-sklearn (Feurer et al., 2015a, 2022), in its current version because the runtime budget was not respected with our setup (Sect. 4.1) [1] [2], but we hope to extend our results in the future.

Deep learning We also integrated 2 recent pretrained deep learning models, sometimes referred to as tabular foundation models (Müller et al., 2023), TabPFN (Hollmann et al., 2023) and HyperFast (Bonet et al., 2024). TabPFN is a meta-trained transformer ensemble that performs in-context learning on tabular datasets with up to 100 features, without needing any hyperparameter tuning. On our only benchmark dataset with > 100 features, *clean1*, we use subsampling to select 100 features at random for TabPFN (Feuer et al., 2023). In contrast to TabPFN, HyperFast uses external datasets to meta-train a hypernetwork, generates smaller, task-specific main networks with the pretrained hypernetwork and the actual training dataset, and optionally fine-tunes and ensembles the main networks.

With the selected datasets and integrated baselines, we can perform thorough benchmark tests in the low-data regime and analyze when certain approaches succeed or fail, as described in the next section.

3.3 Python Interface

PMLBmini is hosted as a Python package on GitHub [3]. It can either be imported into an existing project and run in an existing environment, or used standalone in a Docker container. We provide additional information, including how to evaluate a custom tabular classifier on our dataset collection, both on GitHub and in Appendix B.

Extensibility Each baseline classifier (Sect. 3.2) has been re-implemented as a scikit-learn `BaseEstimator` with a `ClassifierMixin`. Therefore, any class that adheres to scikit-learn's standardized duck-typing approach for creating estimators can be used with our benchmarking tool, allowing researchers and practitioners to add new classifiers with minimal overhead. To provide users with a template on how to extend our suite, we implemented logistic regression with a `Pipeline` and `GridSearchCV` from scikit-learn.

The `tabmini` module The benchmarking interface is exposed through the `tabmini` module, which provides a set of convenience functions for automating the benchmarking process:

- **Benchmark dataset loading** to load our dataset collection (Sect. 3.1), with the option to set `reduced=True` for only loading datasets that have not been used to develop TabPFN's prior (Sect. 4.2). The data is returned through a generator. Although this process could have been integrated within the `compare` function, the latter is able to accept any sort of iterable that maps a

[1]`https://github.com/automl/auto-sklearn/issues/1683`
[2]`https://github.com/automl/auto-sklearn/issues/1695`
[3]`https://github.com/RicardoKnauer/TabMini`

string (dataset name) to a tuple of `pandas dataframes` - design matrix X and labels y. Therefore, the functions have been decoupled.

- **Dummy dataset loading** to validate estimator functionality.
- **Empirical comparison** of a given estimator with the baseline classifiers (Sect. 4.2). By default, we evaluate estimators with the area under the receiver operating characteristic curve (AUC), using a stratified, 3-fold cross validation procedure to obtain mean test performances (Sect. 4.1). The results of the `compare` function are returned as a tuple of `pandas dataframes` - the first dataframe holds the training scores per dataset and estimator, while the second holds test scores.
- **Meta-feature analysis** to extract meta-features from our benchmark datasets and compute associations of these meta-features with performance differences between a given estimator and a simple logistic regression baseline [4], allowing us to analyze under which conditions the estimator succeeds or fails in the low-data regime (Sect. 4.3). We use PyMFE 0.4.3 to extract meta-features for all available meta-feature groups (Alcobaça et al., 2020), including but not limited to simple meta-features (such as the sample size), statistical meta-features (such as the feature means), clustering meta-features (such as the mean silhouette value), information-theoretic meta-features (such as the feature entropies), model-based meta-features (such as the feature importances in a decision tree), and complexity and landmarking meta-features (such as the predictive performance of a k-nearest neighbors classifier). It is possible for some meta-features, such as the feature importances, to contain multiple values, hence we use all available PyMFE summary functions to aggregate them, including but not limited to the mean and frequency in a particular histogram bin. This yields 3932 meta-features in total for in-depth analyses.

4 Experiments

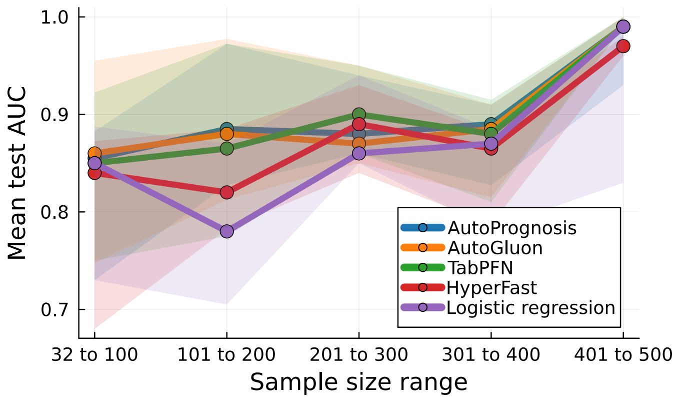
In this section, we describe the experimental setup of our tabular classification benchmark for data-scarce applications (Sect. 3.1) on AutoML and deep learning against logistic regression (Sect. 3.2) using our suite (Sect. 3.3). We then present the experimental results, the best L2-regularization hyperparameter for each benchmark dataset, and the conditions under which AutoML frameworks and pretrained deep neural networks outperform logistic regression in the low-data regime. Note that we also provide additional experimental results for gradient-boosted decision trees using their package defaults in Table 4 in Appendix C. Overall, we find that logistic regression shows a similar discriminative performance to AutoML and deep learning approaches on 55% of the datasets. AutoML methods are better suited for more complex datasets, when more complex classifiers are needed; off-the-shelf deep neural networks are better suited for certain feature distributions, potentially those that resemble their meta-training data.

4.1 Experimental Setup

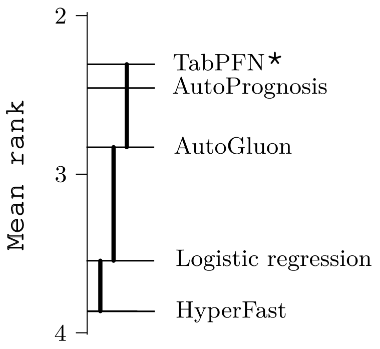
Method details We used our logistic regression implementation with the default optimality gaps via MOSEK 10 (MOSEK ApS, 2023) and JuMP 1.4.0 (Dunning et al., 2017) from Julia 1.8.3. AutoPrognosis 0.1.21 was run from Python 3.10.13 with the default settings, AutoGluon 1.0.0 with the "best quality" preset, and TabPFN 0.1.9 with 32 ensemble members (Hollmann et al., 2023). HyperFast 0.1.3 was used without fine-tuning of the main networks (due to excessive runtimes exceeding the time budget on our hardware configuration, see below), but with 32 ensemble members like TabPFN (Bonet et al., 2024; Hollmann et al., 2023).

Evaluation metrics We measured the discriminative performance in terms of the AUC. The training AUC was recorded to assess overfitting and evaluated with a 1h runtime limit for each

[4] `https://github.com/RicardoKnauer/TabMini/blob/master/tabmini/analysis/meta_feature.py`



(a) Mean test AUC medians and interquartile ranges across sample size ranges in steps of 100.



(b) Critical difference diagram, with a bold vertical bar connecting methods that are not statistically different. ★Note that TabPFN results are biased (Sect. 4.2).

Figure 2: Discriminative performance for AutoML, deep learning, and logistic regression on our benchmark suite PMLBmini.

method per benchmark dataset, the mean test AUC with a 3h runtime limit via a stratified, 3-fold cross-validation procedure (i.e., 1h per fold). The time budget was chosen to reflect current practice and to ensure comparability with prior work (Bonet et al., 2024; Erickson et al., 2020; Hollmann et al., 2023; Müller et al., 2023; Salinas and Erickson, 2023). For logistic regression, we also tracked the best L2-regularization hyperparameter value λ^* obtained on the whole data, and used a nested instead of a non-nested cross-validation procedure (Knauer and Rodner, 2023). To detect pairwise mean test AUC differences between our methods, we used a critical difference diagram based on the Holm-adjusted Wilcoxon signed-rank test with a 0.05 significance level (Demšar, 2006; Benavoli et al., 2016). Each experiment was run on 8 vCPU cores with 32GiB memory on an internal cluster.

4.2 Experimental Results

AutoPrognosis, AutoGluon, TabPFN, HyperFast, and logistic regression show a relatively similar discriminative performance at different sample sizes (Fig. 2a). The largest difference occurs at sample sizes from 101 to 200, with HyperFast and logistic regression only reaching a mean test AUC of 0.82 and 0.78, respectively. There is a trend for all methods to perform better when the sample size increases (Fig. 2a). Each approach wins on at least one benchmark dataset and looses on at least one other dataset (Table 3 in Appendix C), echoing the results of McElfresh et al. (2024). Logistic regression performs on par with or better than the best AutoML or deep learning approach on 16% of the datasets, and lies within 1%, 2%, and 3% of the best approach on 34%, 48%, and 55% of the datasets, respectively - possibly because it is less likely to overfit, especially with smaller sample sizes. Interestingly, even when AutoML methods consider logistic regression for algorithm selection (i.e., AutoPrognosis), they may be outperformed by logistic regression alone (e.g., on the backache dataset), again possibly due to overfitting. When statistically comparing pairwise mean test AUC differences (Fig. 2b), we observe that AutoPrognosis and TabPFN achieve a better rank than logistic regression ($p < 0.05$), whereas AutoGluon and HyperFast are not different from a simple logistic regression baseline ($p > 0.05$). TabPFN's prior was developed on 45% of our benchmark datasets, though (Hollmann et al., 2023), its performance estimates are therefore likely to be overoptimistic. However, manually excluding all overlapping datasets reduced the benchmark dataset collection in our analysis too much to find statistically significant performance differences between any of

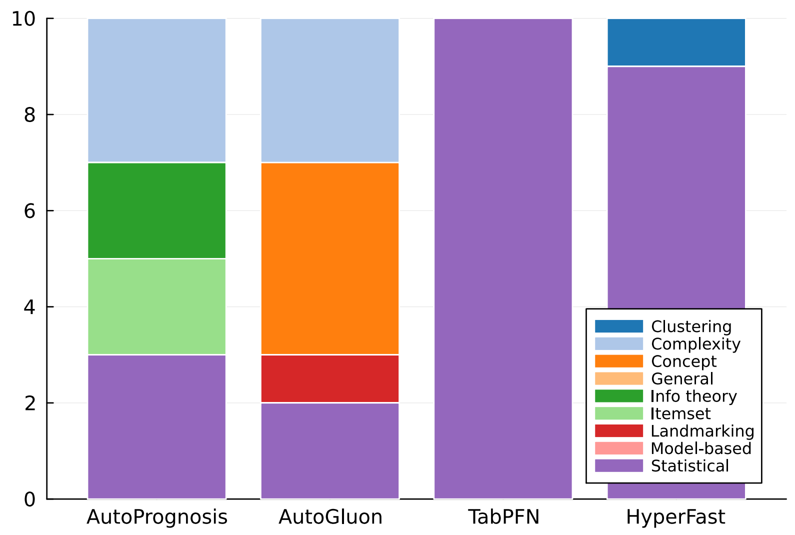


Figure 3: What dataset meta-features influence model performance? The plot shows the meta-feature groups (Alcobaça et al., 2020) that are represented in the top-10 meta-features per approach. To that end, we computed all PyMFE meta-features per dataset, the mean test AUC differences between each AutoML / deep learning method and logistic regression per dataset, the absolute Spearman rank correlation coefficient between each PyMFE meta-feature and the performance difference across datasets (Sect. 3.3); and finally selected the top-10 meta-features with the largest absolute correlations.

the evaluated methods ($p > 0.05$). Finally, we report the best L2-regularization hyperparameter for each dataset that can be used for meta-learning (Table 3 in Appendix C), for instance by leveraging the most similar PMLBmini dataset(s) for zero- or few-shot hyperparameter optimization (Feurer et al., 2015b; Reif et al., 2012).

4.3 Meta-Feature Analysis

In the following, we use our meta-feature analysis tool to extract meta-features from our benchmark datasets and compute relationships of these meta-features with performance differences between each AutoML / deep learning method and logistic regression (Sect. 3.3). This way, we can determine which dataset properties make more complex machine learning methods well- or less well-suited than a simple logistic regression baseline in the low-data regime. Fig. 5 in Appendix C shows the top-3 correlations for each approach. For AutoPrognosis, the most discriminating meta-feature is the ratio of the intra- and extraclass nearest neighbor distance; for AutoGluon, it is the clustering coeffcient. Both measures capture the dataset complexity, are larger for harder classification problems (Lorena et al., 2019), and show a positive relationship with the performance differences. For AutoML, most meta-features with the largest absolute correlations in fact measure the dataset complexity (Fig. 3). For TabPFN and HyperFast, the most discriminating meta-features are the harmonic mean and minimum of each feature, i.e., summary scores from the feature distribution. For deep learning, meta-features with the largest absolute correlations are almost exclusively statistical meta-features that describe the feature distribution (Fig. 3). Therefore, AutoML methods appear to be better suited for more complex datasets that need more complex classifiers, whereas pretrained deep neural networks may be better suited for datasets with feature distributions that resemble their meta-training data. These insights may prove beneficial when developing new or updating existing machine learning systems, for example by increasing the meta-training set diversity for off-the-shelf deep learning models in the low-data regime.

5 Broader Impact and Limitations

Our benchmarking tool, PMLBmini, provides a standardized, diverse collection of 44 small-sized tabular datasets (Sect. 3.1) in combination with a range of different machine learning baselines (Sect. 3.2), accessible via a user-friendly Python interface (Sect. 3.3). This allows researchers and practitioners working with tabular data to rigorously challenge their own classifier for data-scarce applications without much effort, and the community to more easily track progress in the field. Interestingly, our initial set of machine learning approaches reveals that simple baselines like logistic regression should not be prematurely discarded since they frequently perform similar to state-of-the-art AutoML and deep learning methods when data is limited (Sect. 4.2). In fact, trying a simple logistic regression baseline in the low-data regime first and only switching to more complex approaches when needed could save resources (McElfresh et al., 2024) and improve the transparency, trust, and applicability of machine learning systems in practice, for example in the clinical domain (Falla et al., 2021; Knauer and Rodner, 2023; Steyerberg, 2019). We also provide users with a practical meta-feature analysis tool to investigate under which conditions, i.e., dataset properties, certain methods are better suited than others. AutoML frameworks appear to work better when more complex classifiers are needed, pretrained deep learning models when their meta-training data are more similar to the test data (Sect. 4.3). We hope that this will support researchers and practitioners to develop new or improve upon existing machine learning systems, for instance by meta-training deep neural networks on a wider range of small-sized tabular data.

Nevertheless, the development of a benchmark suite also carries risks. As our selected datasets are publicly available, we cannot guarantee that they have not already been used for training or tuning the system that is intended to be benchmarked. As many state-of-the-art AutoML and deep learning methods leverage meta-learning (Sect. 3.2), it is quite possible that meta-training and benchmark datasets overlap to a large extent for some approaches - in this case, users could either accept the inherent bias in the benchmark results or have the option to manually exclude the affected datasets (Sect. 3.3). Overoptimism and "arbitrary" dataset exclusions can potentially undermine the suite's promise to increase the comparability between methods and studies, though. Moreover, we also want to emphasize that our benchmark suite only contains binary classification problems without missing values and with categorical features being numerically encoded (Sect. 3.1), i.e., categorical features contain no more textual information that could be used by AutoML methods (Erickson et al., 2020) or multimodal foundation models (Achiam et al., 2023). For now, PMLBmini therefore only represents a small slice of the data-scarce problems commonly encountered in practice and disadvantages approaches that naturally handle missing values and categorical features such as gradient-boosted decision trees. Additionally, our meta-feature analysis tool currently only relies on a bivariate association measure that does not take confounding factors into account and is not designed to detect meta-feature interactions or non-monotonic relationships (Gijsbers et al., 2022; McElfresh et al., 2024). Further limitations could be addressed by integrating assessments for training or inference speed and encountered errors (Gijsbers et al., 2022; McElfresh et al., 2024), scaling to slightly larger sample sizes for finding out at which point more complex methods start to consistently outperform simple baselines, and subsampling larger tabular datasets to increase the number of benchmark datasets in our suite. We strongly encourage future research and contributions in these directions.

6 Conclusion

Although researchers and practitioners are frequently confronted with data-scarce applications, (very-)small sized tabular data are generally underrepresented in current machine learning benchmarks. In this work, we introduced PMLBmini, a tabular benchmark suite of 44 binary classification datasets with sample sizes ≤ 500. We showed how our benchmarking tool can be used to evaluate current AutoML and deep learning methods, and to analyze if and when they are better suited than a simple logistic regression baseline. In summary, we found that state-of-the-art machine learning approaches fail to appreciably outperform logistic regression on 55% of our benchmark datasets. AutoML frameworks appear to work better for more complex datasets, pretrained deep neural networks for datasets with feature distributions that are more similar to their meta-training data. We therefore recommend to increase the dataset diversity when meta-training off-the-shelf deep neural networks. Since hyperparameter optimization in the low-data regime is inherently difficult, we also provide the community with L2-regularization hyperparamters that can be directly used for meta-learning when data is limited. Finally, we encourage researchers and practitioners to challenge the data efficiency of their own tabular classifier using our suite, and to assess under which conditions it succeeds or fails.

Acknowledgements. This research was funded by the Bundesministerium für Bildung und Forschung (16DHBKI071, 01IS23041C).

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alaa, A. and van der Schaar, M. (2018). Autoprognosis: Automated clinical prognostic modeling via bayesian optimization with structured kernel learning. In *International Conference on Machine Learning (ICML)*, pages 139–148. PMLR.

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

Benavoli, A., Corani, G., and Mangili, F. (2016). Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161.

Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., et al. (2023). Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8.

Bischl, B., Casalicchio, G., Feurer, M., Gijsbers, P., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. (2021). OpenML benchmarking suites. In *Proceedings of the NeurIPS 2021 Datasets and Benchmarks Track.*

Bonet, D., Montserrat, D. M., Giró-i Nieto, X., and Ioannidis, A. G. (2024). Hyperfast: Instant classification for tabular data. *arXiv preprint arXiv:2402.14335.*

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems.*

Brooks, T., Peebles, B., Homes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C. W. Y., Wang, R., and Ramesh, A. (2024). Video generation models as world simulators.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Christodoulou, E., Ma, J., Collins, G. S., Steyerberg, E. W., Verbakel, J. Y., and Van Calster, B. (2019). A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. *Journal of clinical epidemiology*, 110:12–22.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7(1):1–30.

Deza, A. and Atamtürk, A. (2022). Safe screening for logistic regression with l0-l2 regularization. *arXiv*, 2202.

Dunning, I., Huchette, J., and Lubin, M. (2017). Jump: A modeling language for mathematical optimization. *SIAM review*, 59(2):295–320.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. (2020). Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*.

Falla, D., Devecchi, V., Jiménez-Grande, D., Rügamer, D., and Liew, B. X. (2021). Machine learning approaches applied in spinal pain research. *Journal of Electromyography and Kinesiology*, 61:102599.

Feuer, B., Hegde, C., and Cohen, N. (2023). Scaling tabpfn: Sketching and feature selection for tabular prior-data fitted networks. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

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

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015a). Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28.

Feurer, M., Springenberg, J., and Hutter, F. (2015b). Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Fischer, S. F., Feurer, M., and Bischl, B. (2023). OpenML-CTR23–a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*.

Gijsbers, P., Bueno, M. L., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., and Vanschoren, J. (2022). AMLB: an AutoML benchmark. *arXiv preprint arXiv:2207.12560*.

Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., and Vanschoren, J. (2019). An open source AutoML benchmark. *arXiv preprint arXiv:1907.00909*.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2023). Tabpfn: A transformer that solves small tabular classification problems in a second. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hulsebos, M., Demiralp, Ç., and Groth, P. (2023). Gittables: A large-scale corpus of relational tables. *Proceedings of the ACM on Management of Data*, 1(1):1–17.

Imrie, F., Cebere, B., McKinney, E. F., and van der Schaar, M. (2023). Autoprognosis 2.0: Democratizing diagnostic and prognostic modeling in healthcare with automated machine learning. *PLOS Digital Health*, 2(6):e0000276.

Knauer, R. and Rodner, E. (2023). Cost-sensitive best subset selection for logistic regression: A mixed-integer conic optimization perspective. In *German Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 114–129. Springer.

Lorena, A. C., Garcia, L. P., Lehmann, J., Souto, M. C., and Ho, T. K. (2019). How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52(5):1–34.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. (2024). When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36.

Moons, K. G., Altman, D. G., Reitsma, J. B., Ioannidis, J. P., Macaskill, P., Steyerberg, E. W., Vickers, A. J., Ransohoff, D. F., and Collins, G. S. (2015). Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis (TRIPOD): explanation and elaboration. *Annals of internal medicine*, 162(1):W1–W73.

Moons, K. G., Wolff, R. F., Riley, R. D., Whiting, P. F., Westwood, M., Collins, G. S., Reitsma, J. B., Kleijnen, J., and Mallett, S. (2019). PROBAST: a tool to assess risk of bias and applicability of prediction model studies: explanation and elaboration. *Annals of internal medicine*, 170(1):W1–W33.

MOSEK ApS (2023). MOSEK optimizer API for Python. Manual.

Müller, A., Curino, C., and Ramakrishnan, R. (2023). Mothernet: A foundational hypernetwork for tabular classification. *arXiv preprint arXiv:2312.08598*.

Olson, R. S., La Cava, W., Orzechowski, P., Urbanowicz, R. J., and Moore, J. H. (2017). PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10:1–13.

Puri, V., Kataria, A., and Puri, B. (2023). Semi-automated diabetes prediction using autogluon and tabpfn models. In *International Conference on Artificial Intelligence of Things*, pages 289–295. Springer.

Reif, M., Shafait, F., and Dengel, A. (2012). Meta-learning for evolutionary parameter optimization of classifiers. *Machine learning*, 87:357–380.

Riley, R. D., Snell, K. I., Martin, G. P., Whittle, R., Archer, L., Sperrin, M., and Collins, G. S. (2021). Penalization and shrinkage methods produced unreliable clinical prediction models especially when sample size was small. *Journal of Clinical Epidemiology*, 132:88–96.

Romano, J. D., Le, T. T., La Cava, W., Gregg, J. T., Goldberg, D. J., Chakraborty, P., Ray, N. L., Himmelstein, D., Fu, W., and Moore, J. H. (2022). PMLB v1. 0: an open-source dataset collection for benchmarking machine learning methods. *Bioinformatics*, 38(3):878–880.

Salinas, D. and Erickson, N. (2023). Tabrepo: A large scale repository of tabular model evaluations and its automl applications. *arXiv preprint arXiv:2311.02971*.

Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.

Šinkovec, H., Heinze, G., Blagus, R., and Geroldinger, A. (2021). To tune or not to tune, a case study of ridge logistic regression in small or sparse datasets. *BMC Medical Research Methodology*, 21:1–15.

Steyerberg, E. W. (2019). *Clinical prediction models: a practical approach to development, validation, and updating*. Springer.

Van Calster, B., van Smeden, M., De Cock, B., and Steyerberg, E. W. (2020). Regression shrinkage methods for clinical prediction models do not guarantee improved performance: simulation study. *Statistical methods in medical research*, 29(11):3166–3178.

Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. (2014). OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60.

Submission Checklist

1. For all authors…

    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We made sure that the claims in the abstract and introduction are coherent with the contributions.

    (b) Did you describe the limitations of your work? [Yes] See Sect. 5.

    (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sect. 5.

    (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes] See Sect. 5.

2. If you ran experiments…

    (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes] See Sect. 4.

    (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes] See Sect. 3.2 and 4.

    (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes] We accounted for randomness by calculating the mean test AUC from 3 folds for each method per benchmark dataset.

    (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes] We considered the uncertainty across benchmark datasets, both in terms of interquartile ranges across different sample size ranges and in terms of Holm-adjusted Wilcoxon signed-rank tests to detect pairwise mean test AUC differences between methods.

    (e) Did you report the statistical significance of your results? [Yes] See Sect. 4.

    (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [No] We propose our own benchmark, and conduct in-depth evaluations on it.

    (g) Did you compare performance over time and describe how you selected the maximum duration? [No] We did not compare performance over time, but chose a fixed runtime limit of 4h per benchmark dataset for each method. The details and reasoning are described in Sect. 4.1. We concede that recording anytime performance can be very insightful for AutoML frameworks, though, to assess how quickly they converge.

    (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sect. 4.1.

    (i) Did you run ablation studies to assess the impact of different components of your approach? [No] In our experiments, we were primarily interested in comparing a range of existing machine learning approaches on our benchmark datasets in a plug-and-play fashion, mostly using the default settings. However, PMLBmini allows users to easily implement custom machine learning pipelines (Sect. 3.3), and we encourage further research into which specific pipeline components yield appreciable performance benefits in the low-data regime.

3. With respect to the code used to obtain your results…

    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] See `https://github.com/RicardoKnauer/TabMini`.

(b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes] See `https://github.com/RicardoKnauer/TabMini`.

(c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes] See `https://github.com/RicardoKnauer/TabMini`.

(d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] See `https://github.com/RicardoKnauer/TabMini`.

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] See `https://github.com/RicardoKnauer/TabMini`.

4. If you used existing assets (e.g., code, data, models)...

   (a) Did you cite the creators of used assets? [Yes] See Sect. 3.1 and 3.2.

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes] The PMLB license does not require it.

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Sect. 3.1.

5. If you created/released new assets (e.g., code, data, models)...

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes] See `https://github.com/RicardoKnauer/TabMini`.

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes] See `https://github.com/RicardoKnauer/TabMini`.

6. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not use crowdsourcing or conducted research with human subjects.

7. If you included theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We did not include theoretical results.

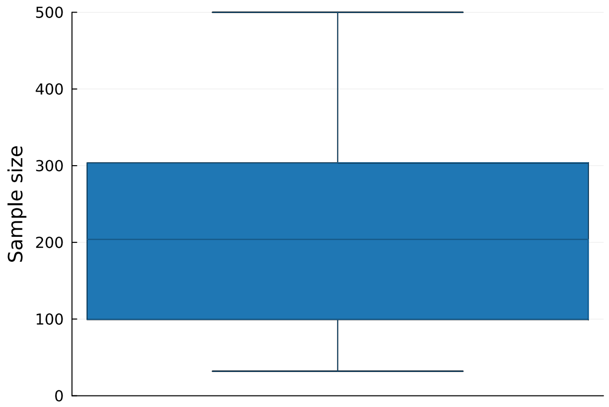
   (b) Did you include complete proofs of all theoretical results? [N/A] We did not include theoretical results.

A Additional Dataset Details

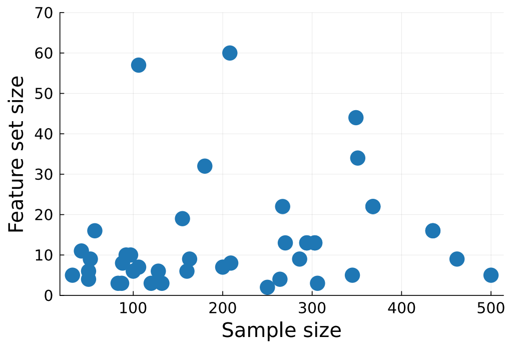
In the following, we provide additional details on our dataset collection in terms of key dataset characteristics in Table 2 and in terms of dataset dimensionality in Fig. 4.

Table 2: Key summary statistics for our benchmark suite in terms of the sample size, feature set size, relative minority class frequency, events per variable (number of instances in the minority class per feature), and number of binary features.

Summary statistic	Sample size	Feature set size	% Minority class	Events per variable	# Binary features
Mean	219	17	37	10	2
Std	133	26	11	12	5
Min	32	2	7	1	0
25%	99	6	29	3	0
50%	204	9	38	7	1
75%	304	16	46	11	2
Max	500	168	50	63	22



(a) Dataset dimensionality in terms of the number of instances across all datasets.



(b) Number of features and instances for each dataset. The *clean1* dataset with a feature set size of 168 is not shown to make the plot more readable.

Figure 4: Dataset dimensionality in our benchmark suite.

B PMLBmini Exemplary Usage

Below, we illustrate how researchers and practitioners can use our benchmarking tool for empirical comparisons and meta-feature analyses with their own tabular classifier in the low-data regime:

```
from yourpackage import YourEstimator
import tabmini

# Load the dataset
# Tabmini also provides a dummy dataset for testing purposes, you can load it with
    tabmini.load_dummy_dataset()
```

```
# If reduced is set to True, the dataset will exclude all the data that has been
    used to develop TabPFN's prior
dataset = tabmini.load_dataset(reduced=False)

# Prepare the estimator you want to benchmark against the other estimators
estimator = YourEstimator()

# Perform the comparison
train_results, test_results = tabmini.compare(
    "MyEstimator",
    estimator,
    dataset,
    working_directory=pathlib.Path.cwd() / "results",
    scoring_method="roc_auc",
    cv=3,
    time_limit=3600,
    device="cpu"
)

# Generate the meta-feature analysis
meta_features = tabmini.get_meta_feature_analysis(dataset, test_results,
    "MyEstimator", correlation_method="spearman")

# Save the results and meta-feature analysis to a CSV file
test_results.to_csv("results.csv")
meta_features.to_csv("meta_features.csv")
```

C Additional Results

In this section, we present additional experimental results in Table 3 and 4 and show the top-3 meta-features for each AutoML and deep learning method in Fig. 5.

Table 3: Training AUC (mean test AUC) and λ^* across all 44 datasets in PMLBmini with sample size M and feature set size N, ordered for sample size ranges in steps of 100.

PMLBmini dataset	**M**	**N**	**AutoML**		**Deep learning**		**Logistic regression**	λ^*
			Auto-Prognosis	**Auto-Gluon**	**TabPFN**	**HyperFast**		
parity5	32	5	0.50 (0.27)	0.04 (**0.98**)	1.00 (0.02)	1.00 (0.02)	0.50 (0.17)	0.5
analcatdata_fraud	42	11	0.93 (**0.86**)	0.99 (0.68)	1.00 (0.79)	0.99 (0.73)	0.89 (0.77)	0.5
analcatdata_aids	50	4	1.00 (**0.73**)	0.94 (0.67)	1.00 (0.63)	0.80 (0.53)	0.78 (0.61)	0.004
analcatdata_bankruptcy	50	6	1.00 (**0.98**)	1.00 (0.97)	1.00 (0.96)	0.99 (0.88)	0.99 (0.97)	0.004
analcatdata_japansolvent	52	9	1.00 (0.85)	0.99 (0.88)	1.00 (**0.91**)	0.97 (**0.91**)	0.94 (0.85)	0.1
labor	57	16	1.00 (0.88)	1.00 (0.95)	1.00 (**0.99**)	1.00 (0.98)	1.00 (0.97)	0.02
analcatdata_asbestos	83	3	0.87 (**0.87**)	0.89 (0.85)	0.93 (0.85)	0.87 (**0.87**)	0.87 (0.86)	0.5
lupus	87	3	0.92 (0.84)	0.86 (0.77)	0.86 (0.82)	0.83 (0.79)	0.85 (**0.85**)	0.1

postoperative_ patient_data	88	8	0.59 (**0.49**)	0.12 (0.46)	0.99 (0.44)	0.87 (0.34)	0.65 (0.38)	0.5
analcatdata_ cyyoung9302	92	10	1.00 (**0.89**)	0.99 (0.85)	0.99 (0.87)	0.96 (0.84)	0.94 (0.87)	0.1
analcatdata_ cyyoung8092	97	10	0.91 (0.73)	0.99 (**0.87**)	0.98 (0.85)	0.91 (0.84)	0.93 (0.79)	0.1
analcatdata_ creditscore	100	6	1.00 (**1.00**)	1.00 (0.99)	1.00 (**1.00**)	0.94 (0.87)	0.97 (0.94)	0.02
median M = 32, ..., 100			0.97 (**0.86**)	0.99 (**0.86**)	1.00 (0.85)	0.95 (0.84)	0.91 (0.85)	
appendicitis	106	7	0.88 (0.78)	0.91 (0.85)	0.97 (0.82)	0.86 (**0.87**)	0.86 (0.84)	0.5
molecular_bio- logy_promoters	106	57	1.00 (0.88)	1.00 (**0.91**)	1.00 (0.88)	1.00 (0.89)	1.00 (0.88)	0.5
analcatdata_ boxing1	120	3	0.97 (**0.89**)	0.96 (0.85)	0.99 (0.76)	0.72 (0.67)	0.68 (0.67)	0.5
mux6	128	6	0.50 (**1.00**)	1.00 (**1.00**)	1.00 (**1.00**)	1.00 (0.95)	0.78 (0.70)	0.5
analcatdata_ boxing2	132	3	0.92 (**0.82**)	0.91 (0.75)	0.85 (0.71)	0.75 (0.70)	0.70 (0.68)	0.5
hepatitis	155	19	0.87 (**0.85**)	0.99 (0.80)	0.99 (**0.85**)	0.92 (0.83)	0.93 (0.84)	0.5
corral	160	6	1.00 (**1.00**)	1.00 (**1.00**)	1.00 (**1.00**)	1.00 (**1.00**)	0.97 (0.96)	0.1
glass2	163	9	1.00 (0.89)	1.00 (**0.91**)	1.00 (0.89)	0.89 (0.79)	0.81 (0.72)	0.004
backache	180	32	0.90 (0.60)	1.00 (0.71)	1.00 (0.75)	0.93 (**0.78**)	0.90 (0.72)	0.5
prnn_crabs	200	7	1.00 (**1.00**)	1.00 (**1.00**)	1.00 (**1.00**)	0.83 (0.81)	1.00 (**1.00**)	0.004
median M = 101, ..., 200			0.95 (**0.89**)	1.00 (0.88)	1.00 (0.87)	0.91 (0.82)	0.88 (0.78)	
sonar	208	60	1.00 (0.88)	1.00 (**0.92**)	1.00 (**0.92**)	0.95 (0.89)	0.95 (0.85)	0.5
biomed	209	8	1.00 (**1.00**)	1.00 (0.96)	1.00 (0.95)	0.96 (0.93)	0.96 (0.94)	0.004
prnn_synth	250	2	0.98 (0.94)	0.96 (**0.95**)	0.97 (**0.95**)	0.93 (0.94)	0.94 (0.94)	0.02
analcatdata_ lawsuit	264	4	1.00 (0.99)	1.00 (0.99)	1.00 (**1.00**)	0.99 (0.98)	1.00 (**1.00**)	0.004
spect	267	22	0.87 (**0.84**)	0.94 (0.81)	0.95 (0.83)	0.90 (0.83)	0.90 (0.82)	0.5
heart_statlog	270	13	0.94 (**0.91**)	0.97 (0.87)	0.98 (0.90)	0.93 (0.89)	0.93 (0.89)	0.5
breast_cancer	286	9	0.76 (0.69)	0.96 (0.67)	0.90 (**0.73**)	0.80 (0.69)	0.73 (0.70)	0.5
heart_h	294	13	0.92 (0.87)	0.96 (0.87)	0.97 (**0.88**)	0.89 (0.85)	0.88 (0.86)	0.5
hungarian	294	13	0.99 (**0.86**)	1.00 (0.85)	0.96 (**0.86**)	0.92 (0.84)	0.89 (0.85)	0.5
median M = 201, ..., 300			0.98 (0.88)	0.97 (0.87)	0.97 (**0.90**)	0.93 (0.89)	0.93 (0.86)	
cleve	303	13	0.96 (**0.90**)	0.98 (**0.90**)	0.99 (0.89)	0.93 (0.88)	0.90 (0.88)	0.5
heart_c	303	13	0.94 (**0.91**)	0.95 (0.90)	0.98 (**0.91**)	0.94 (0.89)	0.92 (**0.91**)	0.5
haberman	306	3	0.86 (0.70)	0.76 (0.71)	0.82 (**0.72**)	0.65 (0.58)	0.70 (0.66)	0.5
bupa	345	5	0.70 (0.66)	0.77 (0.65)	0.72 (**0.68**)	0.72 (0.66)	0.68 (0.67)	0.1

spectf	349	44	1.00 (0.91)	1.00 (**0.94**)	1.00 (0.93)	0.91 (0.87)	0.94 (0.88)	0.1
ionosphere	351	34	1.00 (0.97)	1.00 (**0.98**)	1.00 (**0.98**)	0.96 (0.97)	0.97 (0.90)	0.5
colic	368	22	0.99 (**0.87**)	0.98 (**0.87**)	1.00 (**0.87**)	0.90 (0.86)	0.89 (0.86)	0.5
horse_colic	368	22	0.99 (**0.88**)	0.98 (0.85)	1.00 (0.84)	0.89 (0.83)	0.87 (0.82)	0.5
median M = 301, ..., 400			0.98 (**0.89**)	0.98 (**0.89**)	1.00 (0.88)	0.91 (0.87)	0.90 (0.87)	
house_votes_84	435	16	1.00 (**0.99**)	1.00 (**0.99**)	1.00 (**0.99**)	0.99 (0.98)	0.99 (**0.99**)	0.1
vote	435	16	1.00 (**1.00**)	1.00 (0.99)	1.00 (**1.00**)	0.99 (0.99)	1.00 (0.99)	0.5
saheart	462	9	0.82 (**0.77**)	0.69 (0.75)	0.83 (**0.77**)	0.81 (0.76)	0.79 (**0.77**)	0.5
clean1	476	168	1.00 (0.93)	1.00 (**1.00**)	1.00 (0.99)	0.98 (0.96)	1.00 (**1.00**)	0.004
irish	500	5	1.00 (**1.00**)	1.00 (**1.00**)	1.00 (**1.00**)	0.98 (0.97)	0.85 (0.83)	0.1
median M = 401, ..., 500			1.00 (**0.99**)	1.00 (**0.99**)	1.00 (**0.99**)	0.98 (0.97)	0.99 (**0.99**)	

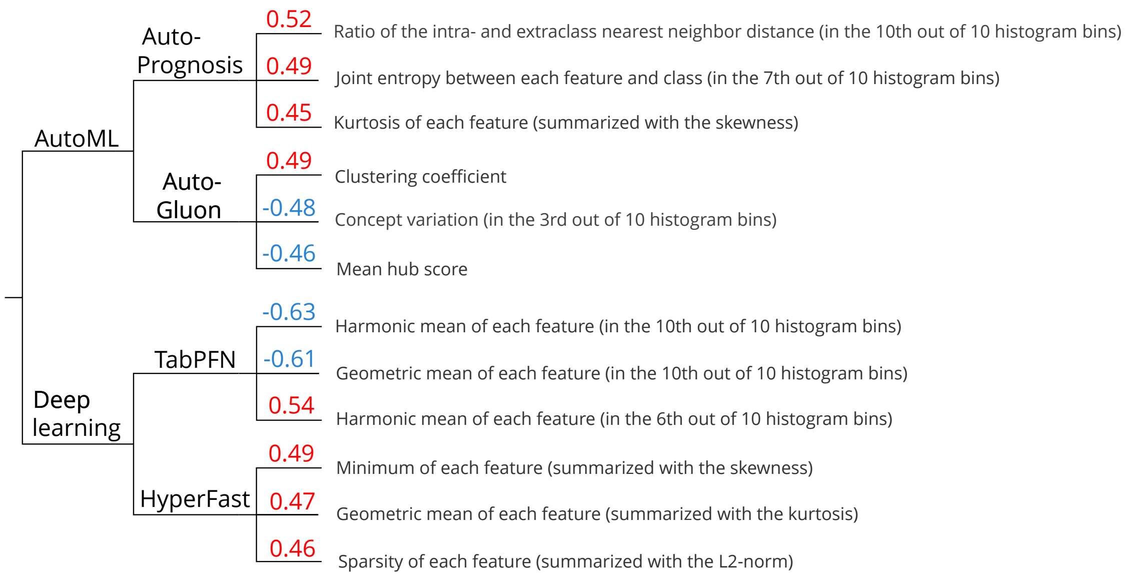

AutoML
Auto-Prognosis
0.52
Ratio of the intra- and extraclass nearest neighbor distance (in the 10th out of 10 histogram bins)
0.49
Joint entropy between each feature and class (in the 7th out of 10 histogram bins)
0.45
Kurtosis of each feature (summarized with the skewness)
Auto-Gluon
0.49
Clustering coefficient
-0.48
Concept variation (in the 3rd out of 10 histogram bins)
-0.46
Mean hub score
Deep learning
TabPFN
-0.63
Harmonic mean of each feature (in the 10th out of 10 histogram bins)
-0.61
Geometric mean of each feature (in the 10th out of 10 histogram bins)
0.54
Harmonic mean of each feature (in the 6th out of 10 histogram bins)
HyperFast
0.49
Minimum of each feature (summarized with the skewness)
0.47
Geometric mean of each feature (summarized with the kurtosis)
0.46
Sparsity of each feature (summarized with the L2-norm)

Figure 5: Top-3 meta-features per approach. We computed all PyMFE meta-features per dataset, the mean test AUC differences between each AutoML / deep learning method and logistic regression per dataset, the absolute Spearman rank correlation coefficient between each PyMFE meta-feature and the performance difference across datasets (Sect. 3.3); and finally selected the top-3 meta-features with the largest absolute correlations. Positive correlation coefficients are shown in red, negative correlation coefficients in blue.

Table 4: Mean test AUC for gradient-boosted decision trees across all 44 datasets in PMLBmini, ordered for sample size. LightGBM reaches a similar median performance to logistic regression; the median performance for XGBoost and CatBoost is worse than for logistic regression, AutoML, and deep neural networks, though.

PMLBmini dataset	**LightGBM**	**XGBoost**	**CatBoost**
parity5	**0.50**	0.19	0.21
analcatdata_fraud	0.50	**0.66**	0.53
analcatdata_aids	0.50	**0.73**	0.59
analcatdata_bankruptcy	0.50	**0.86**	0.84
analcatdata_japansolvent	0.50	0.73	**0.80**
labor	0.50	**0.74**	0.69
analcatdata_asbestos	**0.82**	0.77	0.80
lupus	0.73	0.74	**0.77**
postoperative_patient_data	0.36	0.47	**0.50**
analcatdata_cyyoung9302	**0.84**	0.71	0.77
analcatdata_cyyoung8092	**0.78**	0.74	0.68
analcatdata_creditscore	0.94	**0.99**	0.97
appendicitis	**0.78**	0.77	0.75
molecular_biology_promoters	**0.92**	0.80	0.70
analcatdata_boxing1	**0.69**	**0.69**	0.61
mux6	**0.96**	0.56	0.67
analcatdata_boxing2	**0.78**	**0.78**	0.76
hepatitis	**0.78**	0.69	0.62
corral	**1.00**	0.92	0.90
glass2	**0.92**	0.75	0.70
backache	**0.66**	0.50	0.53
prnn_crabs	**0.97**	0.81	0.78
sonar	**0.91**	0.71	0.71
biomed	**0.95**	0.85	0.73
prnn_synth	**0.94**	0.86	0.86
analcatdata_lawsuit	**0.99**	0.91	0.63
spect	**0.80**	0.61	0.50
heart_statlog	**0.86**	0.74	0.78
breast_cancer	**0.66**	0.59	0.55
heart_h	**0.86**	0.78	0.77
hungarian	**0.84**	0.81	0.77
cleve	**0.86**	0.75	0.75
heart_c	**0.88**	0.79	0.76
haberman	**0.70**	0.59	0.57
bupa	**0.63**	0.59	0.59
spectf	**0.91**	0.69	0.73
ionosphere	**0.97**	0.85	0.83
colic	**0.85**	0.83	0.81
horse_colic	**0.87**	0.83	0.80
house_votes_84	**0.99**	0.96	0.96
vote	**0.99**	0.96	0.95
saheart	**0.70**	0.65	0.67
clean1	**1.00**	**1.00**	**1.00**
irish	**1.00**	**1.00**	**1.00**