Appendix Contents.

A Additional Experiments

A.1 Alternative Leaderboard Versions

Aggregation Methods. Here, we provide more leaderboard variants, using different aggregation strategies. Specifically, we obtain errors err_i for each dataset i by averaging error metrics (1-AUROC for binary, logloss for multiclass, and RMSE for regression) over all outer folds. We then aggregate these errors as follows:

- **Elo**: As described in Section 2.3.
- **Normalized score**: Following Salinas and Erickson [37], we linearly rescale the error such that the best method has a normalized score of one, and the median method has a normalized score of 0. Scores below zero are clipped to zero. These scores are then averaged across datasets.
- **Average rank**: Ranks of methods are computed on each dataset (lower is better) and averaged.
- **Harmonic mean rank**: Taking the harmonic mean of ranks,
$$\frac{1}{\frac{1}{N}\sum_{i=1}^{N}(1/\mathrm{rank}_i)},$$
more strongly favors methods having very low ranks on some datasets. It therefore favors methods that are sometimes very good and sometimes very bad over methods that are always mediocre, as the former are more likely to be useful in conjunction with other methods.
- **Improvability**: We introduce improvability as a metric that measures how many percent lower the error of the best method is than the current method on a dataset. This is then averaged over datasets. Formally, for a single dataset,
$$\text{Improvability} := \frac{\mathrm{err}_i - \mathrm{best\_err}_i}{\mathrm{err}_i} \cdot 100\% \,.$$
Improvability is always between 0% and 100%.

Results. Figure A.1 presents a leaderboard including all models. We impute the results for models on datasets where they are not applicable with the results of RandomForest (default). We choose the default random forest since it is a fast baseline that is sufficiently but not unreasonably weak, to penalize models that are not applicable to all datasets. Table A.1 presents the same data in tabularized format, akin to the current version of the live leaderboard at tabarena.ai. Table A.1 further includes several additional metrics to asses peak average performance, some of which change the ranking (see the color coding) as they are less influenced by model-wise negative outlier results introduced by imputation.
We further investigate our results by presenting the leaderboard across task types. We show the results per task type by computing the results only with datasets from: binary classification in Figure A.2, multiclass classification in Figure A.3, and regression in Figure A.4.
Next, Figure A.5 and Figure A.6 present the results for the TabPFNv2-compatible and TabICL-compatible datasets, but also impute TabPFNv2/TabICL to enable a more direct comparison between these two foundation models. Finally, Figure A.7 presents the result only with datasets for which both TabPFNv2 and TabICL are compatible.

A.2 Analyzing Training Time Limit

In our experiments, we restrict the time to evaluate one configuration on one train split of a dataset to 1 hour. Thus, a model must finish training (across all 8 inner folds) within 1 hour, or its training will be gracefully stopped early. Figure A.8 presents the training runtime for all hyperparameter configurations for all models by visualizing what proportion of configurations (x-axis) took how many seconds for training (y-axis).
We observe that for all models except the GPU-optimized models and EBMs, all configurations trained in under 1 hour. We further investigate the time limit for the GPU-optimized models in Appendix A.3. For EBMs, we notice that the training was not stopped early at the 1 hour time limit, positively influencing its results. As this only concerns a small fraction of hyperparameter trials, we did not rerun the training for EBM.

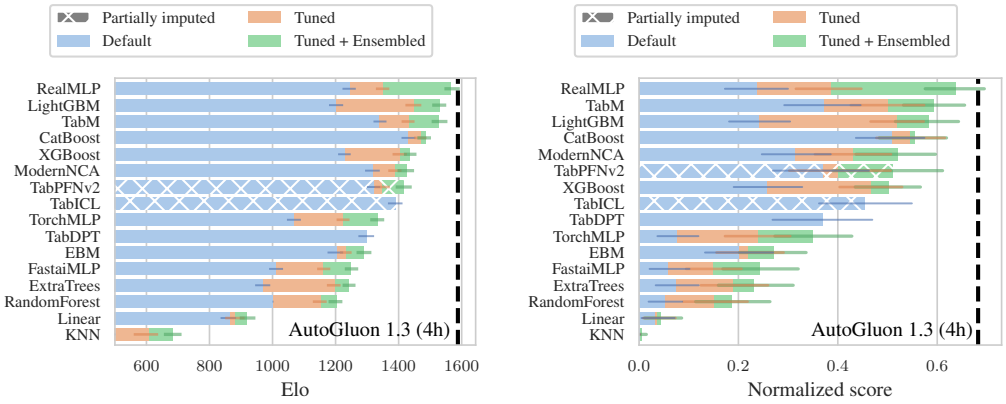

Figure A.1: **TabArena-v0.1 Leaderboard With Imputation for TabPFNv2 and TabICL, Elo (left) and Normalized Scores (right).** For TabPFNv2 and TabICL, on datasets where they are not applicable, we impute their results with RandomForest (default).

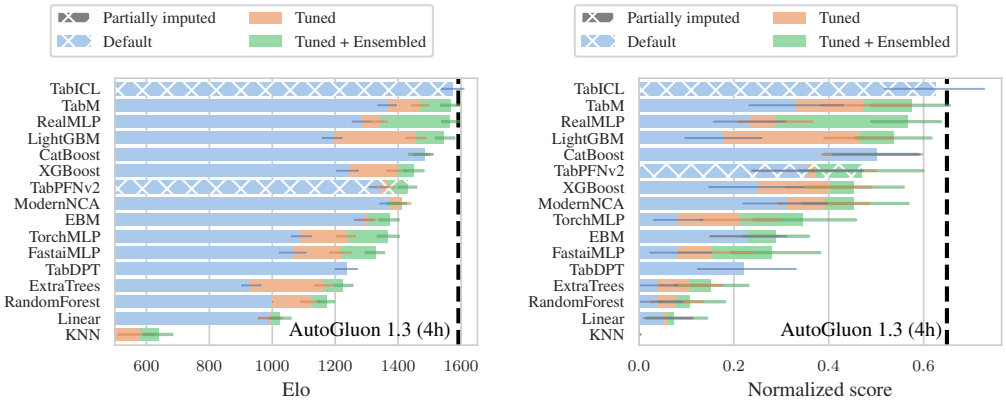

Figure A.2: **Benchmark results on binary classification with Elo (left) and normalized scores (right).** For TabPFNv2 and TabICL, on datasets where they are not applicable, we impute their results with RandomForest (default).

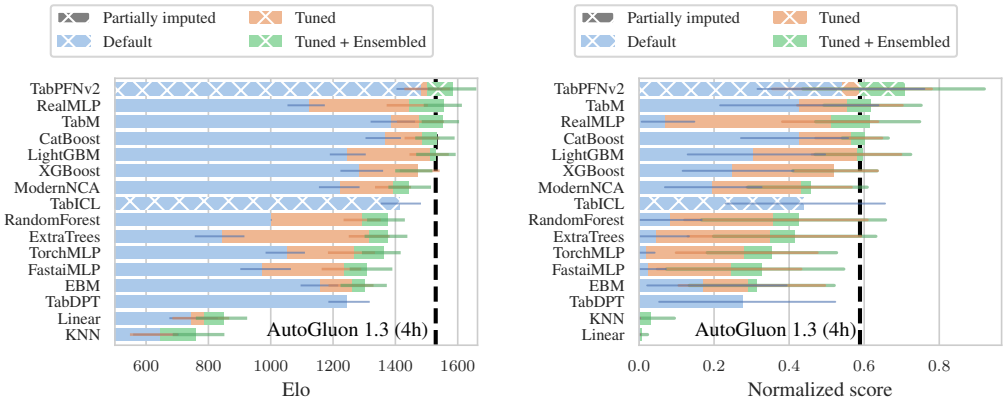

Figure A.3: **Benchmark results on multiclass classification with Elo (left) and normalized scores (right).** For TabPFNv2 and TabICL, on datasets where they are not applicable, we impute their results with RandomForest (default).

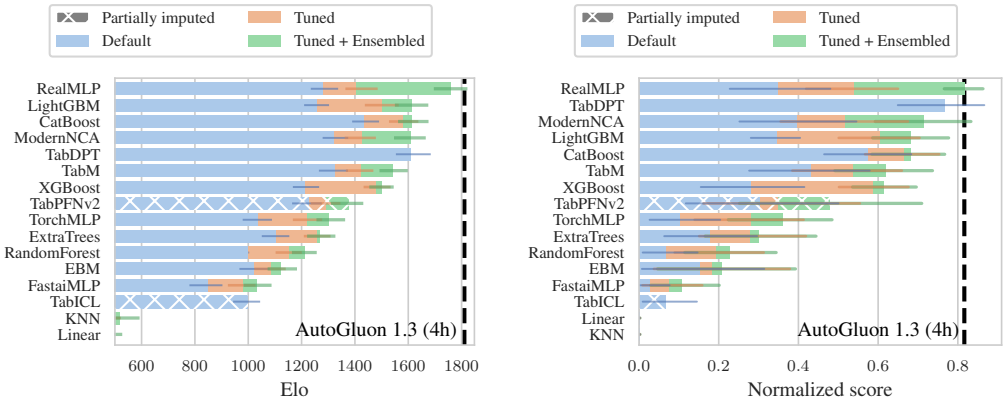

Figure A.4: **Benchmark results on regression with Elo (left) and normalized scores (right).** For TabPFNv2 and TabICL, on datasets where they are not applicable, we impute their results with RandomForest (default).

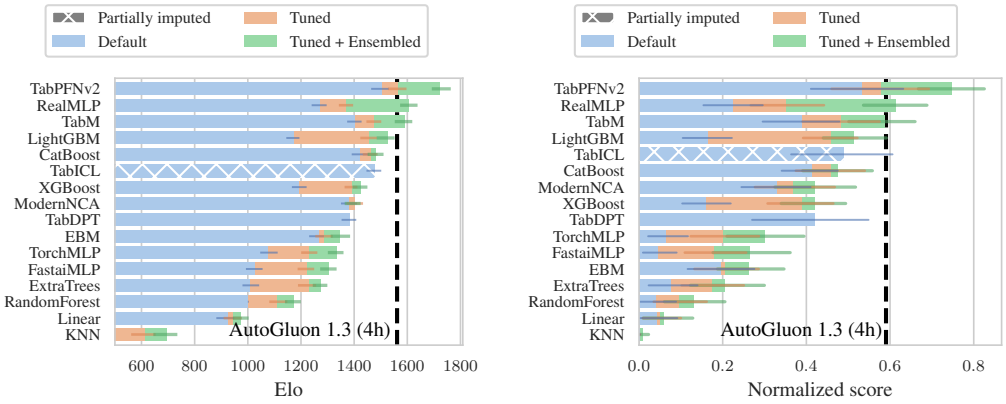

Figure A.5: **Benchmark results on TabPFNv2-compatible datasets with imputed results for TabICL, using Elo (left) and normalized scores (right).** On datasets where TabICL is not applicable, we impute its results with RandomForest (default).

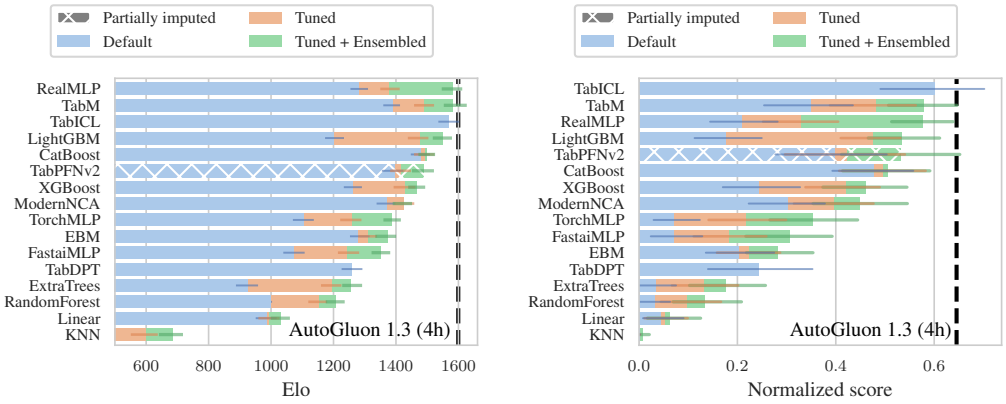

Figure A.6: **Benchmark results on TabICL-compatible datasets with imputed results for TabPFNv2, using Elo (left) and normalized scores (right).** On datasets where TabPFNv2 is not applicable, we impute its results with RandomForest (default).

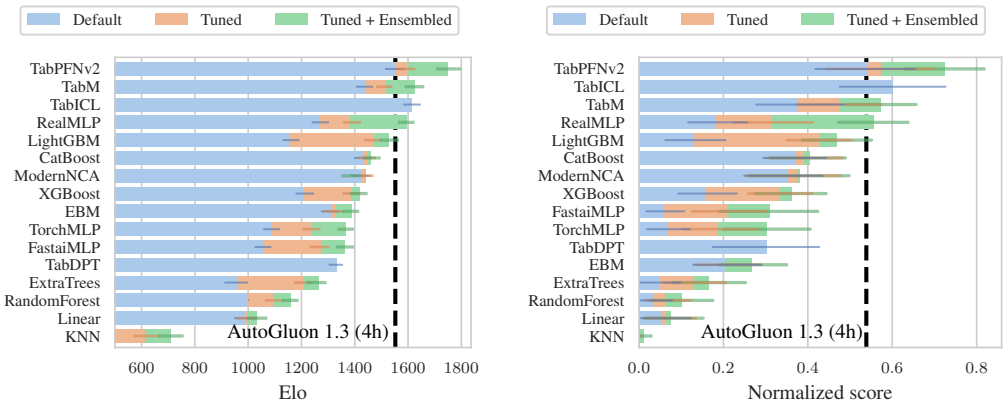

Figure A.7: **Benchmark results on TabPFNv2- and TabICL-compatible datasets using Elo (left) and normalized scores (right).**

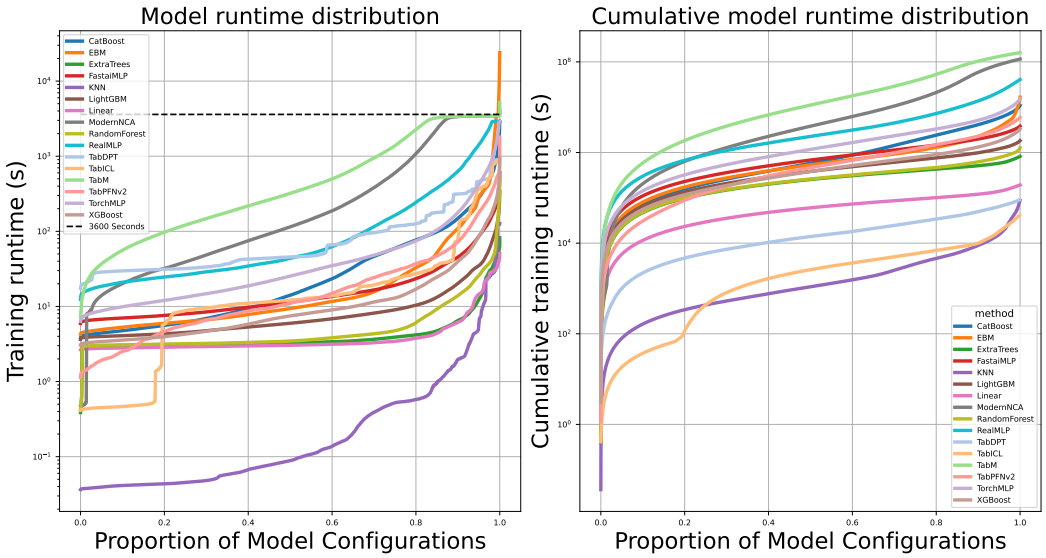

Figure A.8: **Training Runtime Analysis, Runtime Distribution (left) and Cumulative Total Runtime (right) Across Hyperparameter Configurations.** We show the training runtime in seconds for the hyperparameter configurations across models. Notably, TabM and ModernNCA on CPU are impacted by the 1-hour time limit in $\sim$16% of their configurations.

Table A.1: **TabArena-v0.1 Leaderboard.** We show default (D), tuned (T), and tuned + ensembled (T+E) performances. Results of TabPFNv2 and TabICL are imputed with RandomForest (default) for datasets on which they were not run. Times are median times per 1K samples across datasets, averaged over all outer folds per dataset. The best three values in columns are highlighted with gold, silver, and bronze colors. For Elo values, we also indicate their approximate 95% confidence intervals obtained through bootstrapping.

Model	Elo (↑)	Norm. score (↑)	Avg. rank (↓)	Harm. mean rank (↓)	#wins (↑)	Improva-bility (↓)	Train time per 1K [s]	Predict time per 1K [s]
RealMLP (T+E)	$1574_{-30,+22}$	0.638	8.4	4.5	2	6.2%	6566.62	10.26
LightGBM (T+E)	$1536_{-26,+29}$	0.583	9.7	5.3	2	8.1%	417.05	2.64
TabM (T+E)	$1534_{-29,+21}$	0.592	9.8	4.8	3	7.0%	38348.60	18.19
CatBoost (T+E)	$1488_{-22,+23}$	0.555	11.5	7.2	0	7.5%	1658.43	0.65
CatBoost (T)	$1475_{-27,+21}$	0.545	12.1	6.0	2	7.7%	1658.43	0.08
LightGBM (T)	$1453_{-24,+25}$	0.519	13.0	10.4	0	8.8%	417.05	0.33
XGBoost (T+E)	$1443_{-21,+24}$	0.502	13.4	7.8	1	8.9%	693.49	1.69
TabM (T)	$1434_{-32,+25}$	0.501	13.8	7.9	0	8.2%	38348.60	2.04
CatBoost (D)	$1432_{-24,+27}$	0.508	13.9	6.9	2	8.8%	6.83	0.08
ModernNCA (T+E)	$1428_{-28,+23}$	0.519	14.1	5.6	3	8.7%	20604.60	62.20
TabPFNv2 (T+E)	$1414_{-26,+25}$	0.509	14.8	3.1	11	8.1%	3031.50	21.44
XGBoost (T)	$1407_{-21,+24}$	0.467	15.0	11.9	0	9.2%	693.49	0.31
ModernNCA (T)	$1389_{-26,+30}$	0.429	15.9	10.0	0	9.3%	20604.60	3.08
TabICL (D)	$1388_{-23,+23}$	0.453	15.9	4.3	7	8.8%	6.63	1.48
RealMLP (T)	$1350_{-21,+26}$	0.385	17.6	14.7	0	9.7%	6566.62	0.49
TabPFNv2 (T)	$1345_{-23,+22}$	0.399	18.0	5.6	1	10.2%	3031.50	0.46
TabM (D)	$1339_{-20,+24}$	0.370	18.2	11.9	0	11.2%	65.60	1.01
TorchMLP (T+E)	$1333_{-19,+28}$	0.350	18.4	13.3	0	10.2%	2875.52	1.95
TabPFNv2 (D)	$1317_{-27,+28}$	0.370	19.3	5.5	4	11.2%	3.36	0.31
ModernNCA (D)	$1317_{-22,+23}$	0.314	19.3	11.1	1	12.8%	43.53	1.45
TabDPT (D)	$1297_{-23,+25}$	0.369	20.4	4.8	7	11.9%	22.53	8.55
EBM (T+E)	$1289_{-24,+26}$	0.271	20.7	11.7	1	14.3%	1331.68	0.20
FastaiMLP (T+E)	$1250_{-19,+24}$	0.243	22.6	11.7	1	13.6%	593.24	4.47
RealMLP (D)	$1246_{-22,+29}$	0.236	22.8	18.7	0	12.2%	21.86	0.84
ExtraTrees (T+E)	$1243_{-28,+25}$	0.230	22.9	14.9	0	13.9%	183.02	0.76
EBM (T)	$1234_{-23,+27}$	0.219	23.4	16.4	0	14.9%	1331.68	0.02
XGBoost (D)	$1227_{-25,+30}$	0.256	23.5	18.1	0	12.3%	1.94	0.12
TorchMLP (T)	$1221_{-29,+25}$	0.238	23.8	20.1	0	12.2%	2875.52	0.13
LightGBM (D)	$1206_{-28,+28}$	0.241	24.7	21.9	0	13.1%	1.96	0.14
RandomForest (T+E)	$1203_{-29,+22}$	0.187	24.8	12.8	1	14.7%	373.24	0.77
EBM (D)	$1202_{-30,+24}$	0.200	24.8	13.1	1	15.9%	4.67	0.04
ExtraTrees (T)	$1198_{-29,+22}$	0.188	25.1	16.5	0	15.0%	183.02	0.09
FastaiMLP (T)	$1158_{-20,+23}$	0.147	26.8	21.1	0	15.2%	593.24	0.31
RandomForest (T)	$1149_{-26,+26}$	0.150	27.2	16.5	0	15.7%	373.24	0.09
TorchMLP (D)	$1067_{-25,+27}$	0.076	30.6	27.8	0	17.1%	9.99	0.13
FastaiMLP (D)	$1008_{-24,+31}$	0.057	32.7	29.7	0	20.4%	2.86	0.37
RandomForest (D)	$1000_{-0,+0}$	0.052	33.0	31.2	0	20.9%	0.43	0.05
ExtraTrees (D)	$969_{-21,+36}$	0.073	34.0	30.3	0	22.7%	0.25	0.05
Linear (T+E)	$919_{-28,+29}$	0.044	35.6	25.4	0	30.6%	47.50	0.17
Linear (T)	$883_{-29,+22}$	0.036	36.5	31.9	0	31.3%	47.50	0.07
Linear (D)	$859_{-29,+33}$	0.031	37.1	29.8	0	32.7%	1.52	0.09
KNN (T+E)	$683_{-26,+24}$	0.005	40.5	40.2	0	45.2%	3.26	0.18
KNN (T)	$608_{-41,+33}$	0.000	41.4	41.3	0	46.8%	3.26	0.04
KNN (D)	$456_{-47,+46}$	0.000	42.8	42.6	0	54.1%	0.05	0.02

A.3 Tabular Deep Learning on GPU vs. CPU

In our main experiments, we ran TabM, ModernNCA, and RealMLP on CPU instead of GPU to make our experiments affordable. As observed in Appendix A.2, GPU-optimized models, especially TabM and ModernNCA, reach the per-configuration limit of 1 hour for $\sim$15% of their hyperparameter configurations. In these cases, their training is gracefully stopped, and a potentially non-converged checkpoint is used for inference.

To further investigate the impact of this early stopping on predictive performance, we ran the first 50 from the 200 configurations from TabM and ModernNCA on GPU. Figure A.9 shows that for TabM and ModernNCA, models trained on GPU outperform their CPU counterparts. Figure A.10 further

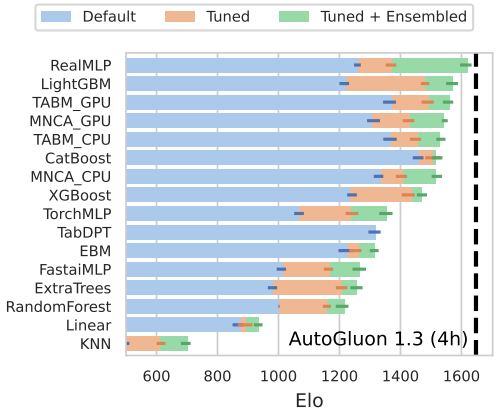

Figure A.9: **Predictive Performance of TabM and ModernNCA on CPU vs. GPU.** We show the predictive performance of TabM and ModernNCA when run on CPU (TabM_CPU, MNCA_CPU) and GPU (TabM_GPU, MNCA_GPU) across TabArena. For this study, the number of configurations for TabM and ModernNCA is limited to 50 for both CPU and GPU, while the other models use 200 configurations. Therefore, this figure should only be used to conclude that GPU improves results over CPU, rather than drawing conclusions on performance compared to other models.

demonstrates that the hyperparameter configurations of TabM and ModernNCA train much faster on GPU than on CPU. Moreover, we observe that only a tiny proportion ($\sim 0.1\%$) of configurations trained on GPU are affected by the time limit. We conclude from this ablation that training TabM, ModernNCA, and RealMLP on CPU with a time limit of 1 hour negatively influences their predictive performance. While the influence is marginal for RealMLP, it seems non-marginal for TabM and ModernNCA and could change the ranking on the full leaderboard.

As maintainers of `TabArena`, we aim to remedy this negative influence as a first proof-of-concept of the living benchmark. We are rerunning all 200 configurations for TabM, ModernNCA, and RealMLP on GPU and will update `TabArena` with the new results.

A.4 TabArena-Lite

Benchmarking can quickly become very expensive, especially with a sophisticated protocol to guarantee robust results. To reduce the cost of benchmarking, we introduce `TabArena-Lite`. `TabArena-Lite` is a continually maintained subset of `TabArena` that consists, in its first version, of all datasets with one outer fold. Figure A.11 shows results on `TabArena-Lite`, using 200 hyperparameter configurations per model, but only a single outer fold for all datasets. The results are similar to the results on `TabArena` in Figure 1, showing that `TabArena-Lite` is a good indicator of model performance.

To further reduce the cost of benchmarking, we also recommend running new models on `TabArena-Lite` with one default hyperparameter configuration and optionally with a lower number of random hyperparameter configurations (e.g., 25). As all other models in `TabArena` are tuned, a less heavily tuned model that performs comparably could already show that a new model is promising. We designate `TabArena-Lite` to be used in academic studies and find any novel model that outperforms other models on at least one dataset, even if it is not among the best on average, a valuable publication. Furthermore, we as maintainers use the performance on `TabArena-Lite` to prioritize the integration of new models into `TabArena`. We envision `TabArena-Lite` also as a living, continuously updated subset. Ideally, future work could determine a method that finds the optimal and most representative subset of partitions and datasets in `TabArena` to populate `TabArena-Lite`.

A.5 Investigating Statistical Significance

We investigate the statistical significance between models by using critical difference diagrams (CDDs) to represent the results of a Friedman test and then a Nemenyi post-hoc test ($\alpha = 0.05$) from

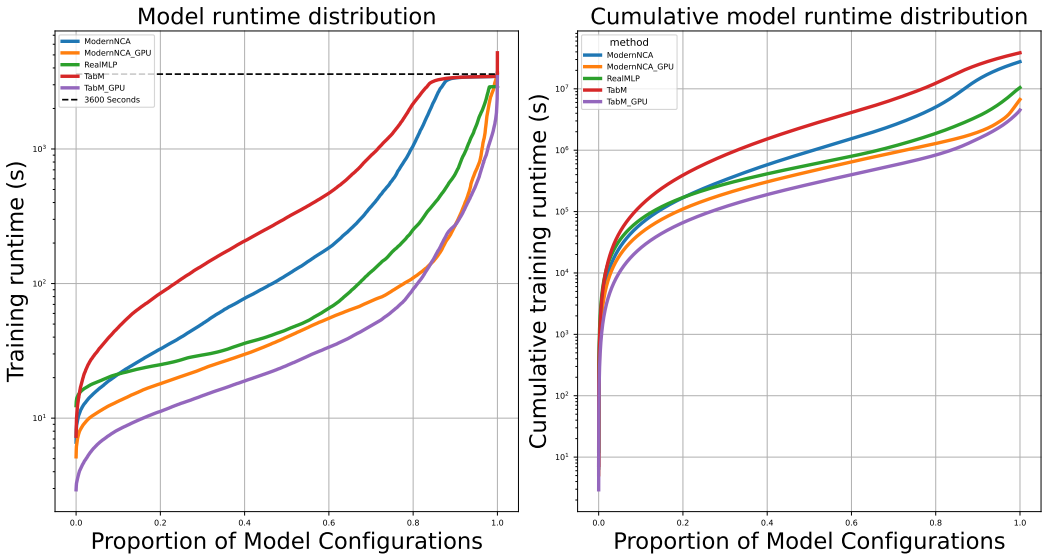

Figure A.10: **Runtime Analysis of TabM and ModernNCA on CPU vs. GPU.** We show the training runtime distribution of 50 hyperparameter configurations for TabM and ModernNCA trained on CPU and GPU across all TabArena datasets. For CPU, approximately 16% of runs are early stopped due to the 1-hour time limit. For GPU, less than 0.1% of runs are early stopped due to the time limit. We also include RealMLP trained on CPU for 50 configurations, for which less than 3% of the configurations were early stopped based on time.

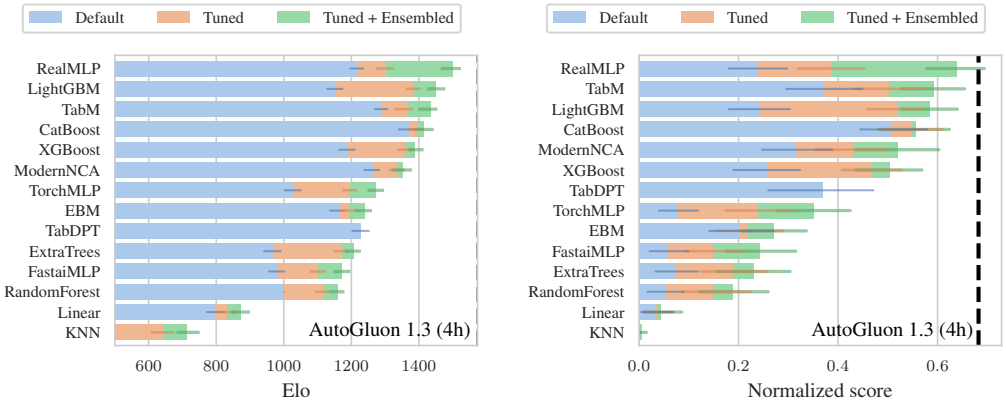

Figure A.11: **Benchmark results on `TabArena-Lite` using Elo (left) and normalized scores (right).** Our main leaderboard with `TabArena-Lite`, a subset of `TabArena` consisting of all datasets, but only with one outer fold.

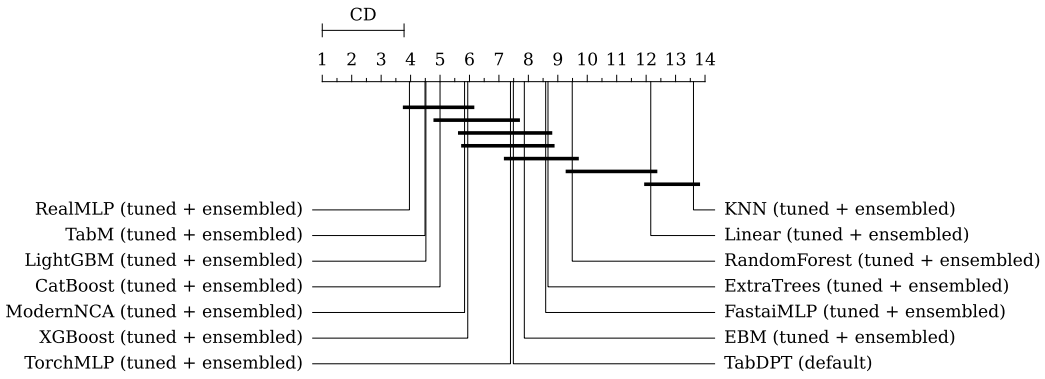

Figure A.12: **Critical Difference Diagram for tuned+ensembled methods on the full benchmark.** Lower ranks are better; horizontal bars connect methods that are not statistically significantly different.

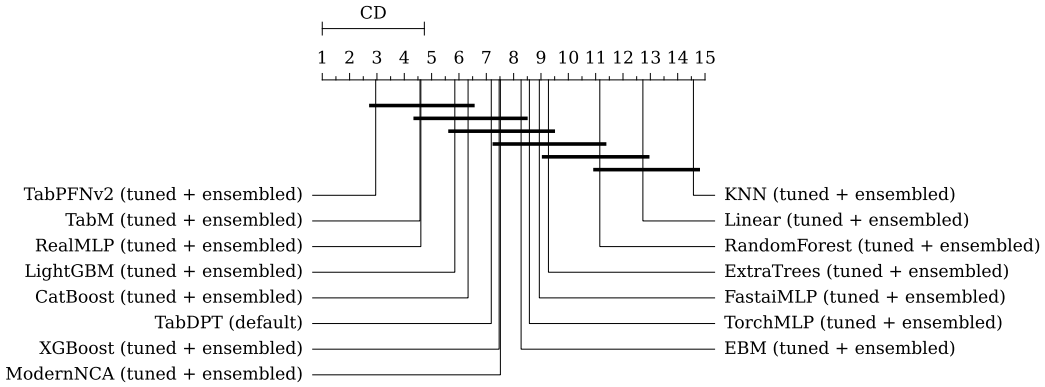

Figure A.13: **Critical Difference Diagram for tuned+ensembled methods on TabPFNv2-compatible datasets.** Lower ranks are better; horizontal bars connect methods that are not statistically significantly different.

AutoRank[2]. Figures A.12 to A.14 show the CDDs for the full benchmark, TabPFNv2-compatible datasets, and TabICL-compatible datasets with respect to the peak performance of the models, i.e., tuned + ensembled where available. We further investigate statistical significance per-dataset in Appendix E.

We observe that there always exists a group of not statistically significantly different top models containing at least one deep learning model and GBDT, and when available, TabPFNv2 and TabICL.

[2] `https://github.com/sherbold/autorank`

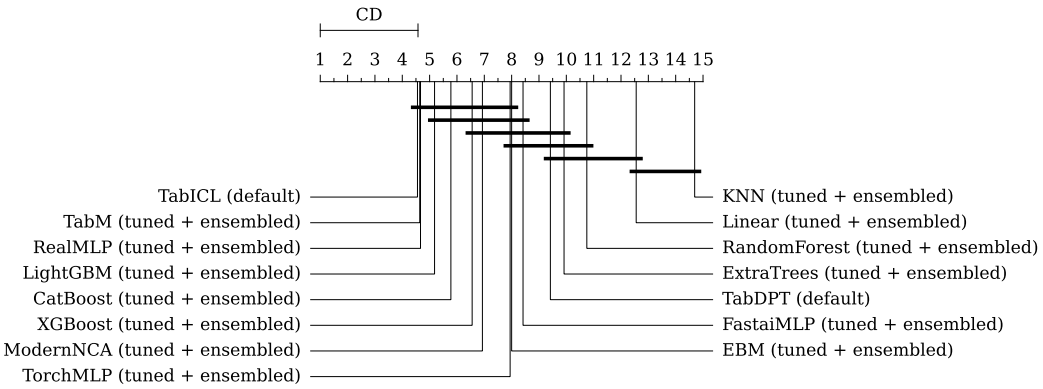

Figure A.14: **Critical Difference Diagram for tuned+ensembled methods on TabICL-compatible datasets.** Lower ranks are better; horizontal bars connect methods that are not statistically significantly different.

B Data curation

For initializing the `TabArena` benchmark, we surveyed all the datasets used in 13 previous benchmarking studies: 450 from PMLB(Mini) [41, 42, 43], 72 from OpenML-CC18 [29], 45 from Grinsztajn et al. [32], 11 from Shwartz-Ziv and Armon [31], 11 from Gorishniy et al. [30], 176 from TabZilla [33], 35 from OpenML-CTR23 [34], 104 from AMLB [35], 8 from TabRed [10], 10 from Tschalzev [8], 279 from TabRepo [37], 118 from PyTabKit [20], and 300 from TALENT [36, 45].
These studies were selected with the goal of covering a wide range of datasets used in tabular benchmarking so that we can clean up the field from problematic or unsuitable datasets. Therefore, each of the studies represents a frequently used benchmark or a general milestone study in the field of tabular machine learning. Combining the dataset collections results in 1053 uniquely named datasets.

For `TabArena`-v0.1, we aimed at using only those datasets representing realistic, predictive tabular data tasks practitioners would be interested to solve. Therefore, we define a set of selection criteria described in Appendix B.1. Two of the coauthors manually investigated each of the datasets and applied our selection criteria. We publicly share their notes and curated metadata: `tabarena.ai/dataset-curation`. Furthermore, we share insights from our curation process in Appendix B.3
Importantly, we did not exhaustively test each dataset for each of our curation criteria, but proceeded with the next dataset whenever a dataset clearly met at least one of our criteria for exclusion. Therefore, Figure 2 represents the first reason for exclusion that we noticed in a dataset.
Surprisingly, only 51 datasets satisfying all criteria remained. Appendix B.2 provides additional information on the selected datasets. We consider our data curation a clean-up for tabular data benchmarking that is necessary, but imperfect. Therefore, we aim to continuously improve the data selection and invite researchers to challenge our documented decisions. Appendix D.3 details protocols to contribute new datasets to `TabArena` by applying our criteria.

B.1 Dataset Selection Criteria

The datasets for Tabarena-v0.1 were curated by applying the following criteria:

Unique datasets: We want the `TabArena` benchmark to be representative of a wide range of tasks without overrepresenting particular tasks. Therefore, we conduct a four-stage deduplication procedure: (1) Automatically filter data sets by name if they match after transforming to the lower case and removing filling characters ' '| '_' | '-'. (2) Manually remove datasets where different names were used for the same data set in different studies. (3) Manually remove alternative versions of the same dataset, i.e., temporal data sampled at different rates, or dataset versions with alternative targets. (4) Remove different datasets representing the same task from the same source (i.e., a collection of ML for software tasks named kc1-3).

IID Tabular Data: We exclude datasets that are non-IID. More specifically, we exclude datasets whose tasks require a non-random split, such as a temporal or group-based split. We leave a non-IID realization of `TabArena` with temporal and time-series data for future work.

Tabular Domain Tasks: We exclude datasets from non-tabular modalities transformed into a tabular format. Thus, we exclude featureized image, text, audio, or time series forecasting data. Likewise, we exclude problems that would no longer be solved with tabular machine learning, such as tabular data of control problems solved nowadays by reinforcement learning. While some tasks from other modalities may still be solved using feature extraction and tabular learning methods, it is impossible to assess that without domain experts. Instead of making uninformed decisions, we exclude all datasets from other domains for `TabArena`-v0.1. In future versions, we consider adding datasets from other domains if there is evidence that tabular learning methods are still a reasonable solution for the task. Therefore, we actively invite researchers from other domains to share datasets for which they apply tabular learning methods.

Real Random Distribution: We exclude purely artificial data, or any subset thereof, generated by a deterministic function, by sampling from a seeded random process, or by simulating a random distribution. We note that such datasets are still interesting toy functions that help analyze the theoretical capabilities of models qualitatively. Yet, they do not represent distributions from real-world predictive machine learning tasks. While some simulated datasets (i.e., higgs, or MiniBooNE) were conceptualized as machine learning tasks, we decided to exclude them for `TabArena`-v0.1 for consistency.

Predictive Machine Learning Task: We exclude tabular data that does not originally stem from a predictive machine learning task for classification or regression. Thus, we exclude tabular data intended for *scientific discovery* tasks such as anomaly detection, subgroup discovery, data visualization, or causal inference. In particular, this includes survey data never intended for use in a predictive machine learning task. While data from scientific discovery applications can be used for predictive machine learning tasks, we only include it if the original data source intended its use for predictive machine learning, or if a follow-up work re-used the data in a real-world application.

Moreover, we exclude non-predictive tables, where the target label is not predictable based on statistical information from other columns, such as those commonly found in collections like WikiTables [50] or GitTables [51].

We exclude datasets that are trivial to solve and therefore do not represent challenging ML tasks allowing to investigate model differences. We define trivial datasets as datasets where one of the following criteria applies: (1) at least one of the models in our scope is consistently able to achieve perfect performance; (2) multiple models achieve exactly the same highest performance. Note that after applying our set of criteria, none of the considered datasets was found to be trivial.

Size Limit: We exclude datasets that are tiny or large because they tend to require fundamentally different methods. Tiny datasets require a methodological focus on avoiding overfitting, while methods for large datasets must be very efficient during training. We aim to include tiny and large datasets with dedicated evaluation protocols in future versions of `TabArena`. For `TabArena`-v0.1, we exclude datasets with fewer than 500 or more than $250,000$ samples, measured as the number of training samples after applying our train-test splits. Note that after applying our whole set of criteria, none of the datasets was excluded solely due to being too large, while many datasets were excluded due to a small sample size.

Data Quality: We exclude datasets that suffer from one of the following data quality issues: (1) heavily preprocessed datasets, such as those where the whole dataset was already used for preprocessing in a way that leaks the target (i.e., PCA); (2) datasets for which we could not find sufficient information to judge their source and preprocessing; (3) datasets with an irreversible target leak. In general, we try to find the original state of the dataset and include it, if applicable. We do not generally exclude preprocessed datasets, as datasets are rarely published without any preprocessing, e.g., due to anonymization. We leave a benchmark with model-specific, domain-specific pre-processing per dataset for future work.

No License Issues: We exclude any dataset whose license does not allow sharing or using it for an academic benchmark. By doing so, we guard the future of `TabArena` as a living benchmark, its maintainers, and, most importantly, its users from legal threat.

As a result, we exclude several promising datasets, e.g., due to the default license of Kaggle competitions. Thus, progress towards a less data-restrictive license on Kaggle could greatly benefit the academic community. Likewise, any progress towards sharing more public domain datasets for tabular predictive machine learning would be highly beneficial.

Open-access Structured Data API : We exclude datasets that cannot be automatically downloaded from a tabular data repository. Eligible data repositories must be open-access, i.e., users do not need an account to download data. Furthermore, the repositories require a structured data and task representation, including metadata information such as feature types, the target column, and outer splits. To the best of our knowledge, only OpenML fulfills these requirements so far. If applicable due to licensing, we manually upload datasets to OpenML to include them in `TabArena`. This criterion is necessary to enable automated benchmarking and a straightforward user experience.

Ethically Unambiguous Tasks: We exclude datasets with tasks that pose ethical concerns, such as the Boston Housing dataset[3]. While curating our datasets, we flagged such datasets and excluded them. We implore the community to investigate our curated datasets for ethical concerns further and immediately notify the maintainers of `TabArena` about potential problems.

B.2 Included Datasets Details

Table B.1 presents a detailed overview for all datasets included in `TabArena`-v0.1. We further share all tasks and datasets as an OpenML suite (ID 457, alias "tabarena-v0.1"). Namely, we included the following datasets: wine_quality [52], in_vehicle_coupon_recommendation [53], HR_Analytics_Job_Change_of_Data_Scientists [54], houses [55], hiva_agnostic [56], heloc [57], healthcare_insurance_expenses [58], hazelnut-spread-contaminant-detection [59], GiveMeSomeCredit [60], Food_Delivery_Time [61], Fitness_Club [62], E-CommereShippingData [63], diamonds [64], Diabetes130US [65], diabetes [66], customer_satisfaction_in_airline [67], credit_card_clients_default [68], credit-g [69], concrete_compressive_strength [70], coil2000_insurance_policies [71], churn [72], blood-transfusion-service-center [73], Bioresponse [74], Bank_Customer_Churn [75], bank-marketing [76, 77], APSFailure [78], Another-Dataset-on-used-Fiat-500 [79], anneal [80], Amazon_employee_access [81], airfoil_self_noise [82], Is-this-a-good-customer [83], jm1 [84], kddcup09_appetency [85], Marketing_Campaign [86], maternal_health_risk [87], miami_housing [88], MIC [89], NATICUSdroid [90], online_shoppers_intention [91], physiochemical_protein [92], polish_companies_bankruptcy [93], qsar-biodeg [94], QSAR-TID-11 [95], QSAR_fish_toxicity [96], SDSS17 [97], seismic-bumps [98], splice [99], students_dropout_and_academic_success [100], superconductivity [101], taiwanese_bankruptcy_prediction [102], website_phishing [103].

B.3 Noteworthy Observations from Curation

We observed several trends while curating the datasets for `TabArena`-v0.1. To improve the discussion related to datasets in our community, we share some noteworthy trends below.

- For various datasets, it was not possible to automate the selection process, because the metadata that would be required is not available. Therefore, given the current state of data repositories, we consider that automated curation procedures produce more biased results than careful manual curation. Finding out which splits are appropriate for a task, or whether the targets were created using deterministic functions, requires substantial effort and oftentimes, reading and understanding the papers where datasets were introduced. To still make the inclusion of datasets as objective as possible, we introduce a checklist for new datasets in Appendix D.3.
- Most of the datasets excluded due to license issues were Kaggle datasets with restrictive licenses, which otherwise would have been well-suited for inclusion. In the future, we hope that more high-quality datasets with less restrictive licenses will become available, also on Kaggle.

[3] `https://fairlearn.org/main/user_guide/datasets/boston_housing_data.html`

Table B.1: **Datasets included in TabArena-v0.1.** 'Dataset (Task) ID' represents the OpenML dataset and task IDs, 'name' the dataset name, and Ref. the reference corresponding to the dataset. 'N' represents the no. of samples, 'd' the no. of features, 'C' the no. of classes (- for regression tasks), and '% cat' represents the percentage of features that are categorical. 'Subset' indicates whether the dataset has been used for the sub-benchmarks focusing on TabPFNv2 (left) and TabICL (right).

Dataset (Task) ID	Name	Ref.	N	d	C	% cat	Subset
46913 (363621)	blood-transfusion-service-center	[73]	748	5	2	20.0	✓| ✓
46921 (363629)	diabetes	[66]	768	9	2	11.11	✓| ✓
46906 (363614)	anneal	[80]	898	39	5	84.62	✓| ✓
46954 (363698)	QSAR_fish_toxicity	[96]	907	7	-	0.0	✓| ✗
46918 (363626)	credit-g	[69]	1000	21	2	66.67	✓| ✓
46941 (363685)	maternal_health_risk	[87]	1014	7	3	14.29	✓| ✓
46917 (363625)	concrete_compressive_strength	[70]	1030	9	-	0.0	✓| ✗
46952 (363696)	qsar-biodeg	[94]	1054	42	2	14.29	✓| ✓
46931 (363675)	healthcare_insurance_expenses	[58]	1338	7	-	42.86	✓| ✗
46963 (363707)	website_phishing	[103]	1353	10	3	100.0	✓| ✓
46927 (363671)	Fitness_Club	[62]	1500	7	2	57.14	✓| ✓
46904 (363612)	airfoil_self_noise	[82]	1503	6	-	16.67	✓| ✗
46907 (363615)	Another-Dataset-on-used-Fiat-500	[79]	1538	8	-	12.5	✓| ✗
46980 (363711)	MIC	[89]	1699	112	8	84.82	✓| ✓
46938 (363682)	Is-this-a-good-customer	[83]	1723	14	2	64.29	✓| ✓
46940 (363684)	Marketing_Campaign	[86]	2240	26	2	34.62	✓| ✓
46930 (363674)	hazelnut-spread-contaminant-detection	[59]	2400	31	2	3.23	✓| ✓
46956 (363700)	seismic-bumps	[98]	2584	16	2	25.0	✓| ✓
46958 (363702)	splice	[99]	3190	61	3	100.0	✓| ✓
46912 (363620)	Bioresponse	[74]	3751	1777	2	0.06	✗| ✗
46933 (363677)	hiva_agnostic	[56]	3845	1618	3	100.0	✗| ✗
46960 (363704)	students_dropout_and_academic_success	[100]	4424	37	3	48.65	✓| ✓
46915 (363623)	churn	[72]	5000	20	2	25.0	✓| ✓
46953 (363697)	QSAR-TID-11	[95]	5742	1025	-	0.0	✗| ✗
46950 (363694)	polish_companies_bankruptcy	[93]	5910	65	2	1.54	✓| ✓
46964 (363708)	wine_quality	[52]	6497	13	-	7.69	✓| ✗
46962 (363706)	taiwanese_bankruptcy_prediction	[102]	6819	95	2	1.05	✓| ✓
46969 (363689)	NATICUSdroid	[90]	7491	87	2	100.0	✓| ✓
46916 (363624)	coil2000_insurance_policies	[71]	9822	86	2	4.65	✓| ✓
46911 (363619)	Bank_Customer_Churn	[75]	10000	11	2	45.45	✓| ✓
46932 (363676)	heloc	[57]	10459	24	2	4.17	✓| ✓
46979 (363712)	jm1	[84]	10885	22	2	4.55	✓| ✓
46924 (363632)	E-CommereShippingData	[63]	10999	11	2	45.45	✓| ✓
46947 (363691)	online_shoppers_intention	[91]	12330	18	2	44.44	✓| ✓
46937 (363681)	in_vehicle_coupon_recommendation	[53]	12684	25	2	88.0	✓| ✓
46942 (363686)	miami_housing	[88]	13776	16	-	6.25	✓| ✗
46935 (363679)	HR_Analytics_Job_Change_ of_Data_Scientists	[54]	19158	13	2	76.92	✗| ✓
46934 (363678)	houses	[55]	20640	9	-	0.0	✗| ✗
46961 (363705)	superconductivity	[101]	21263	82	-	0.0	✗| ✗
46919 (363627)	credit_card_clients_default	[68]	30000	24	2	16.67	✗| ✓
46905 (363613)	Amazon_employee_access	[81]	32769	10	2	100.0	✗| ✓
46910 (363618)	bank-marketing	[76, 77]	45211	14	2	57.14	✗| ✓
46928 (363672)	Food_Delivery_Time	[61]	45451	10	-	30.0	✗| ✗
46949 (363693)	physiochemical_protein	[92]	45730	10	-	0.0	✗| ✗
46939 (363683)	kddcup09_appetency	[85]	50000	213	2	18.31	✗| ✓
46923 (363631)	diamonds	[64]	53940	10	-	30.0	✗| ✗
46922 (363630)	Diabetes130US	[65]	71518	48	2	83.33	✗| ✓
46908 (363616)	APSFailure	[78]	76000	171	2	0.58	✗| ✓
46955 (363699)	SDSS17	[97]	78053	12	3	25.0	✗| ✓
46920 (363628)	customer_satisfaction_in_airline	[67]	129880	22	2	77.27	✗| ✓
46929 (363673)	GiveMeSomeCredit	[60]	150000	11	2	9.09	✗| ✓

- The large amount of datasets from other modalities seems to be an artifact from times before the development of high-performing modality-specific approaches. At least 16 datasets were images for handwritten digit or letter recognition. As those tasks are clearly outdated, we excluded them. To be consistent, we also excluded datasets consisting of features from image data for which we were not able to assess whether the tasks are outdated. Features extracted from image data are not an exclusion criterion for datasets in `TabArena`, as long

as they represent a meaningful task and tabular models are a reasonable approach to solve those tasks.

- The huge amount of tiny datasets is likely an artifact of a time when data collection was done at a much smaller scale than nowadays. Only four of the 142 tiny datasets for which we found a publication date were published later than 2010. Moreover, many of the tiny datasets seem to have originated in educational content, such as books or toy examples in tutorials.
- Several datasets used in previous benchmarking studies were originally introduced as part of a series of AutoML Challenges. Datasets in these challenges were often released (and shared) with obscured, non-meaningful names. Most of the datasets are ablated versions of other datasets, and therefore have led to unintended duplicates in existing benchmarks. Furthermore, many of those datasets were from other domains, like images or text.
- Of the 254 datasets with alternative versions listed in Figure 2, most are from the PMLB benchmark [42] and represent differently parameterized versions of artificially created datasets: 118 are Feynman equations and 62 are Friedman data generation functions.
- Throughout the benchmarks, inconsistent versions of the same datasets were used: tasks were binarized, features were removed, and sometimes even targets were changed. This can be partially attributed to the misleading versioning system of OpenML. Subsequent versions of the same datasets correspond to a different upload with the same name, not necessarily an improved version of the same datasets. Therefore, some studies reused the alternative versions of the dataset uploaded under the same name for specific studies. In gathering the datasets, we disregarded which version of a dataset was used and solely focused on names. Therefore, some alternative versions were already filtered for the set of 1053 datasets with unique names. In our benchmark, we always searched for the raw version and used the dataset with minimal preprocessing.
- After applying all other criteria, only 51 datasets were found to satisfy the IID criterion, while 68 did not. This underscores the findings of Rubachev et al. [10] that all previous benchmarks used random splits inappropriately. `TabArena` aims to end this malpractice.
- A large number of datasets are tabular but were not intended to be used for predictive tasks. Most of these datasets were filtered due to being 'scientific discovery' tasks, some due to quality issues. In general, some of these datasets might still be useful for benchmarking if they represent realistic distributions and target functions. However, most of the datasets filtered due to this criterion appeared to be relatively simple tasks. That is, some were already found to be trivially solvable in other studies, and some contained only a few features. In the future, we are open to considering including datasets not initially intended for predictive tasks, if no other issues are found, and if one can argue for potential predictive machine learning applications.

C Model Curation

C.1 Implementation Framework Details

We implement models (and their unit tests) based on the `AbstractModel` framework[4] from AutoGluon [19]. In particular, we implement model-specific preprocessing, training, and inference within the `AbstractModel` framework for all models. The framework allows us to use all functionalities from AutoGluon, TabRepo, and in extension scikit-learn [24] to run models in a standardized way. Moreover, the pipeline logic encompassing models within `TabArena` is implemented in a tested, sophisticated framework that is regularly used in real-world applications.

To integrate models in `AbstractModel` framework, we require two properties of a model implementation: (**I**) Iteratively trained models (e.g., GBDTs or MLPs) must support early stopping based on a time limit. Moreover, they must support the use of externally provided validation data. (**II**) We require a default model-specific preprocessing pipeline that handles, if needed, data anomalies such as NaN values, categorical features, or feature scaling. The model-agnostic preprocessing of the

[4] `https://auto.gluon.ai/stable/tutorials/tabular/advanced/tabular-custom-model.html`

AbstractModel framework detects categorical features, transforms text or image features, and cleans common data problems.
The original implementations of some models do not fulfill these requirements; thus, we added support ourselves or together with the model authors. Our requirements aim to get the most out of models in a proper benchmark and in real-world pipelines. Early stopping based on a time limit avoids model failures due to time constraints in benchmarks and is quintessential for integration into time-constrained, real-world pipelines. Likewise, only models that support externally provided validation data can be properly used in pipelines with pre-defined validation splits. Finally, different models require different preprocessing, and relying only on one shared model-agnostic preprocessing pipeline is inappropriate. We detail the model-agnostic and model-specific below.

Foundation Model Implementation Details. For all foundation models, we refit on training and validation data instead of using cross-validation ensembles, following recommendations from the authors of TabPFN and TabICL. We hypothesize that the foundation models do not gain much from cross-validation ensembles because, unlike other models, they do not utilize the validation data per fold for early stopping during training. Thus, their in-context learning might benefit more from using the training and validation data as context for inference on test data.
The foundation models TabPFNv2 and TabICL have been released with restrictions in terms of dataset size. In particular, TabPFNv2 is restricted to datasets with up to $10,000$ training samples, 500 features, and 10 classes for classification tasks. TabICL is constrained to classification tasks with up to $100,000$ training samples and 500 features. TabDPT has no size restrictions because it natively relies on context retrieval, dimensionality reduction, and class codes during inference [23]. For context retrieval, we use the default context size of 1024 described in the paper. Thereby, we override the implementation's default of 128, which we found to perform poorly in preliminary experiments. We use the newest available checkpoints for all foundation models. For TabDPT, we use `tabdpt1_1.pth`. For TabICL, we use `tabicl-classifier-v1.1-0506.ckpt`. For TabPFN, we use the defaults for classification `tabpfn-v2-classifier.ckpt` and regression `tabpfn-v2-regressor.ckpt`, as well as all other checkpoints during HPO: `tabpfn-v2-classifier-gn2p4bpt.ckpt`, `tabpfn-v2-classifier-llderlii.ckpt`, `tabpfn-v2-classifier-od3j1g5m.ckpt`, `tabpfn-v2-classifier-vutqq28w.ckpt`, `tabpfn-v2-classifier-znskzxi4.ckpt`, `tabpfn-v2-regressor-09gpqh39.ckpt`, `tabpfn-v2-regressor-2noar4o2.ckpt`, `tabpfn-v2-regressor-5wof9ojf.ckpt`, `tabpfn-v2-regressor-wyl4o83o.ckpt`.

Model-agnostic Preprocessing. Our model-agnostic preprocessing relies on AutoGluon's `AutoMLPipelineFeatureGenerator`[5]. The model-agnostic preprocessing can handle boolean, numerical, categorical, datetime, and text columns. Importantly, the implementation of a model can control how the model-agnostic preprocessing treats the input data. As a result, a model could obtain raw text and datetime columns as input, such that its model-specific preprocessing can handle them. For `TabArena`, we let the model-agnostic preprocessing handle text and datetime columns. Text columns are transformed to n-hot encoded n-grams. Datetime columns are converted into a Pandas datetime and into multiple columns representing the year, month, day, and day of the week. Numerical columns are left untouched. Categorical columns are replaced with categorical codes to save memory space. The columns are, nevertheless, further treated as categorical. Finally, constant or duplicated columns are dropped. Importantly, we always keep missing values and delegate handling them to the model-specific preprocessing.

Model-specific Preprocessing. We perform minimal invasive model-specific preprocessing and otherwise rely on the preprocessing already implemented within the model's code. Specifically, we use the following model-specific preprocessing before passing the data to the model's code:

- **CatBoost**, **LightGBM**, **XGBoost**, **EBM**, **TabICL**, **TabPFNv2**, **FastaiMLP**, and **TorchMLP** do not use any custom model-specific preprocessing and rely entirely on the model's code.
- **RandomForest** and **ExtraTrees** use ordinal encoding for categorical variables. Missing values are imputed to 0.
- **TabDPT** uses ordinal encoding for categorical variables.

[5] `https://auto.gluon.ai/stable/tutorials/tabular/tabular-feature-engineering.html`

- **RealMLP** handles missing numericals by mean imputation with a missingness indicator.
- **TabM** and **ModernNCA** use the numerical quantile-based preprocessing from TabM and then use mean imputation with an indicator for numerical features.
- **Linear** uses one-hot-encoding, mean or median imputation (hyperparameter), standard scaling, and quartile transformation (hyperparameter).
- **KNN** drops all categorical features and fills missing numerical values with 0. Moreover, it uses leave-one-out cross-validation instead of 8-fold cross-validation. The leave-one-out cross-validation is natively implemented into the KNN model logic and allows for obtaining the validation predictions per sample very efficiently.

C.2 Hyperparameter search spaces

In the following, we list some details and hyperparameter search spaces for different models:

- The search spaces for **CatBoost** (Table C.1), **LightGBM** (Table C.2), **XGBoost** (Table C.3), **RandomForest** (Table C.4), and **ExtraTrees** (Table C.5) were determined based on experiments. We assessed a large set of hyperparameters inspired by the respective documentations as well as several papers [e.g., 20, 37, 104] and experimentally determined good ranges for them for tuning. We verified that the new search spaces outperform the original search spaces on TabRepo [37].

  For gradient-boosted trees, we use the implementation from AutoGluon with its early stopping logic and `n_estimators=10_000`. Compared to TabRepo, which used 300 overall estimators for Random Forest and ExtraTrees and used out-of-bag predictions for validation, we fit these models using 8-fold CV with 50 estimators per model, resulting in 400 estimators overall.
- For **EBM**(Table C.6), we use a search space provided by the authors.
- For **RealMLP** (Table C.7), we also use a search space provided by the authors. For the default parameters, we turn off label smoothing since we are not using accuracy as our evaluation metric, as recommended by Holzmüller et al. [20].
- For **TabM** (Table C.8) and **ModernNCA** (Table C.9), we use search spaces coordinated with the authors. For the batch size, we choose a training-set-size dependent logic following the TabM paper [9]:

$$\text{batch\_size} = \begin{cases} 32 & , N < 2800 \\ 64 & , N \in [2800, 4500) \\ 128 & , N \in [4500, 6400) \\ 256 & , N \in [6400, 32000) \\ 512 & , N \in [32000, 108000) \\ 1024 & , N \geq 108000\ . \end{cases}$$

- **FastaiMLP** (Table C.10) and **TorchMLP** (Table C.11) were taken from AutoGluon, in dialogue with the maintainers/authors.
- **Linear** (Table C.12) and **KNN** (Table C.13) were taken from TabRepo.
- For **TabPFNv2**, we use the search space from the original paper and the official repository, in coordination with the authors.
- For **TabICL** and **TabDPT**, we only use their default configurations. For TabICL and TabDPT, we use the newest checkpoint (see Appendix C.1), unlike the original paper.

Table C.1: Hyperparameter search space for CatBoost.

Hyperparameter	Space
learning_rate	LogUniform([0.005, 0.1])
bootstrap_type	Bernoulli
subsample	Uniform([0.7, 1.0])
grow_policy	Choice(["SymmetricTree", "Depthwise"])
depth	UniformInt([4, 8])
colsample_bylevel	Uniform([0.85, 1.0])
l2_leaf_reg	LogUniform([1e-4, 5])
leaf_estimation_iterations	LogUniformInt([1, 20])
one_hot_max_size	LogUniformInt([8, 100])
model_size_reg	LogUniform([0.1, 1.5])
max_ctr_complexity	UniformInt([2, 5])
boosting_type	Plain
max_bin	254

Table C.2: Hyperparameter search space for LightGBM.

Hyperparameter	Space
learning_rate	LogUniform([0.005, 0.1])
feature_fraction	Uniform([0.4, 1.0])
bagging_fraction	Uniform([0.7, 1.0])
bagging_freq	1
num_leaves	LogUniformInt([2, 200])
min_data_in_leaf	LogUniformInt([1, 64])
extra_trees	Choice([False, True])
min_data_per_group	LogUniformInt([2, 100])
cat_l2	LogUniform([0.005, 2])
cat_smooth	LogUniform([0.001, 100])
max_cat_to_onehot	LogUniformInt([8, 100])
lambda_l1	Uniform([1e-4, 1.0])
lambda_l2	Uniform([1e-4, 2.0])

Table C.3: Hyperparameter search space for XGBoost.

Hyperparameter	Space
learning_rate	LogUniform([0.005, 0.1])
max_depth	LogUniformInt([4, 10])
min_child_weight	LogUniform([0.001, 5.0])
subsample	Uniform([0.6, 1.0])
colsample_bylevel	Uniform([0.6, 1.0])
colsample_bynode	Uniform([0.6, 1.0])
reg_alpha	Uniform([1e-4, 5.0])
reg_lambda	Uniform([1e-4, 5.0])
grow_policy	Choice(["depthwise", "lossguide"])
max_cat_to_onehot	LogUniformInt([8, 100])
max_leaves	LogUniformInt([8, 1024])

Table C.4: Hyperparameter search space for Random Forest.

Hyperparameter	Space
max_features	Uniform([0.4, 1.0])
max_samples	Uniform([0.5, 1.0])
min_samples_split	LogUniformInt([2, 4])
bootstrap	Choice([False, True])
n_estimators	50
min_impurity_decrease	LogUniform([1e-5, 1e-3])

Table C.5: Hyperparameter search space for ExtraTrees.

Hyperparameter	Space
max_features	Choice(["sqrt", 0.5, 0.75, 1.0])
min_samples_split	LogUniformInt([2, 32])
bootstrap	False
n_estimators	50
min_impurity_decrease	Choice([0.0, 1e-5, 3e-5, 1e-4, 3e-4, 1e-3], p=[0.5, 0.1, 0.1, 0.1, 0.1, 0.1])

Table C.6: Hyperparameter search space for EBM.

Hyperparameter	Space
max_leaves	UniformInt([2, 3])
smoothing_rounds	UniformInt([0, 1000])
learning_rate	LogUniform([0.0025, 0.2])
interactions	Uniform([0.95, 0.999])
interaction_smoothing_rounds	UniformInt([0, 200])
min_hessian	LogUniform([1e-10, 1e-2])
min_samples_leaf	UniformInt([2, 20])
validation_size	Uniform([0.05, 0.25])
early_stopping_tolerance	LogUniform([1e-10, 1e-5])
gain_scale	LogUniform([0.5, 5.0])

Table C.7: Hyperparameter search space for RealMLP. With probability 0.5, either the "Default" or the "Large" option is chosen for each configuration.

Hyperparameter	Space	
n_hidden_layers	UniformInt([2, 4])	
hidden_sizes	rectangular	
hidden_width	Choice([256, 384, 512])	
p_drop	Uniform([0.0, 0.5])	
act	mish	
plr_sigma	LogUniform([1e-2, 50])	
sq_mom	$1 -$ LogUniform([0.005, 0.05])	
plr_lr_factor	LogUniform([0.05, 0.3])	
scale_lr_factor	LogUniform([2.0, 10.0])	
first_layer_lr_factor	LogUniform([0.3, 1.5])	
ls_eps_sched	coslog4	
ls_eps	LogUniform([0.005, 0.1])	
lr	LogUniform([0.02, 0.3])	
wd	LogUniform([0.001, 0.05])	
use_ls	Choice([False, True])	
early_stopping_additive_patience	40	
early_stopping_multiplicative_patience	3	
	Default (prob=0.5)	Large (prob=0.5)
plr_hidden_1	16	Choice([8, 16, 32, 64])
plr_hidden_2	4	Choice([8, 16, 32, 64])
n_epochs	256	Choice([256, 512])
use_early_stopping	False	True

Table C.8: Hyperparameter search space for TabM.

Hyperparameter	Space
batch_size	auto
patience	16
amp	False
arch_type	tabm-mini
tabm_k	32
gradient_clipping_norm	1.0
share_training_batches	False
lr	LogUniform([1e-4, 3e-3])
weight_decay	Choice([0.0, LogUniform([1e-4, 1e-1])])
n_blocks	UniformInt([2, 5])
d_block	Choice([128, 144, 160, . . . , 1008, 1024])
dropout	Choice([0.0, Uniform([0.0, 0.5])])
num_emb_type	pwl
d_embedding	Choice([8, 12, 16, 20, 24, 28, 32])
num_emb_n_bins	UniformInt([2, 128])

Table C.9: Hyperparameter search space for ModernNCA.

Hyperparameter	Space
dropout	Uniform([0.0, 0.5])
d_block	UniformInt([64, 1024])
n_blocks	Choice([0, UniformInt([0, 2])])
dim	UniformInt([64, 1024])
num_emb_n_frequencies	UniformInt([16, 96])
num_emb_frequency_scale	LogUniform([0.005, 10.0])
num_emb_d_embedding	UniformInt([16, 64])
sample_rate	Uniform([0.05, 0.6])
lr	LogUniform([1e-5, 1e-1])
weight_decay	Choice([0.0, LogUniform([1e-6, 1e-3])])
temperature	1.0
num_emb_type	plr
num_emb_lite	True
batch_size	auto

Table C.10: Hyperparameter search space for FastaiMLP.

Hyperparameter	Space
layers	Choice([[200], [400], [200, 100], [400, 200], [800, 400], [200, 100, 50], [400, 200, 100]])
emb_drop	Uniform([0.0, 0.7])
ps	Uniform([0.0, 0.7])
bs	Choice([128, 256, 512, 1024, 2048])
lr	LogUniform([5e-4, 1e-1])
epochs	UniformInt([20, 50])

Table C.11: Hyperparameter search space for TorchMLP.

Hyperparameter	Space
learning_rate	LogUniform([1e-4, 3e-2])
weight_decay	LogUniform([1e-12, 0.1])
dropout_prob	Uniform([0.0, 0.4])
use_batchnorm	Choice([False, True])
num_layers	UniformInt([1, 5])
hidden_size	UniformInt([8, 256])
activation	Choice(["relu", "elu"])

Table C.12: Hyperparameter search space for LinearModel.

Hyperparameter	Space
C	Uniform([0.1, 1000])
proc.skew_threshold	Choice([0.9, 0.99, 0.999, None])
proc.impute_strategy	Choice(["median", "mean"])
penalty	Choice(["L2", "L1"])

Table C.13: Hyperparameter search space for KNN.

Hyperparameter	Space
n_neighbors	Choice([3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 20, 30, 40, 50])
weights	Choice(["uniform", "distance"])
p	Choice([2, 1])

C.3 TabArena Ensemble

The `TabArena` ensemble highlighted in Figure 6 was created by ensembling a portfolio, a set of hyperparameter configurations across models. Given a portfolio, we evaluate each of its models in sequence until a time limit is reached or all models have been evaluated. Then, we post-hoc ensemble [1] all evaluated hyperparameter configurations. For the sake of Figure 6, we simulated the `TabArena` ensemble using the result artifacts.
We created a portfolio following the learning procedure introduced by Salinas and Erickson [37] using leave-one-dataset-out cross-validation with a portfolio of size 200 and 40 ensemble selection steps. We leave further discussion and investigation of portfolio learning with the results of `TabArena` to future work.

D Using and Contributing to the Living Benchmark

D.1 Benchmarking with TabArena

To benchmark a model, a user must (1) implement their model in the `AbstractModel` framework; (2) create a search space; (3) run the implementation on `TabArena` or `TabArena-Lite`; (4) and analyze the results. We provide code and more detailed documentation for these three steps in our code repositories with examples: tabarena.ai/code-examples. Below, we provide a snapshot[6] of code snippets for each step: model implementation (Listing 1), search space (Listing 2), benchmarking (Listing 3), and analysis of the results (Listing 4).

Listing 1: Implementing a custom RandomForest model for TabArena.

```python
import numpy as np
from autogluon.core.models import AbstractModel
from autogluon.features import LabelEncoderFeatureGenerator

class CustomRandomForestModel(AbstractModel):
    ag_key = "CRF"
    ag_name = "CustomRF"

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self._feature_generator = None

    def _preprocess(self, X: pd.DataFrame, is_train=False, **kwargs)
        -> np.ndarray:
        """Model-specific preprocessing of the input data."""
        X = super()._preprocess(X, **kwargs)
        if is_train:
            self._feature_generator = LabelEncoderFeatureGenerator(
                verbosity=0)
            self._feature_generator.fit(X=X)
        if self._feature_generator.features_in:
            X = X.copy()
            X[self._feature_generator.features_in] = self.
                _feature_generator.transform(
                X=X
            )
        return X.fillna(0).to_numpy(dtype=np.float32)

    def _fit(self, X, y):
        from sklearn.ensemble import RandomForestRegressor,
            RandomForestClassifier
        if self.problem_type in ["regression"]:
            model_cls = RandomForestRegressor
        else:
            model_cls = RandomForestClassifier

```

[6]Parts of this snapshot may become outdated due to the benchmarking system being updated.

```
        X = self.preprocess(X, is_train=True)
        self.model = model_cls(**self._get_model_params())
        self.model.fit(X, y)
```

Listing 2: Creating a search space for the custom RandomForest model.

```
def get_configs_for_custom_rf(num_random_configs):
    from autogluon.common.space import Int
    from tabrepo.utils.config_utils import ConfigGenerator

    gen_custom_rf = ConfigGenerator(
        model_cls=CustomRandomForestModel,
        manual_configs=[{}],
        search_space== {
            "n_estimators": Int(4, 50),
        },
    )
    return gen_custom_rf.generate_all_bag_experiments(
        num_random_configs=num_random_configs
    )
```

Listing 3: Benchmarking the custom RandomForest model.

```
import openml
from tabrepo.benchmark.experiment import run_experiments

task_ids = openml.study.get_suite("tabarena-v0.1").tasks
task_metadata = {
    task_id: openml.tasks.get_task(task_id).get_dataset().name
    for task_id in task_ids
}
methods = get_configs_for_custom_rf(num_random_configs=1)

run_experiments(
    expname="/path/to/output/dir",
    tids=task_ids,
    task_metadata=task_metadata,
    methods=methods,
    # TabArena-Lite - only run on the first split of each dataset.
    repeat_fold_pairs=[(0, 0)],
    folds=None,
    repeats=None,
)
```

Listing 4: Comparing the custom RandomForest model to the leaderboard.

```
import pandas as pd
from tabrepo import EvaluationRepository
from tabrepo.nips2025_utils.load_final_paper_results import
    load_paper_results
from tabrepo.paper.paper_runner_tabarena import PaperRunTabArena

from . import post_process_local_results
from . import rename_default

# Prepare local results (e.g., simulate HPO and ensembling)
repo_dir = post_process_local_results(output_dir="/path/to/output/dir"
    )
repo = EvaluationRepository.from_dir(repo_dir)
repo.set_config_fallback(repo.configs()[0])
plotter = PaperRunTabArena(repo=repo, output_dir="example_artifacts",
    backend="native")
df_results = plotter.run_no_sim()
is_default = df_results["framework"].str.contains("_c1_") & (
    df_results["method_type"] == "config")
```

```
df_results.loc[is_default, "framework"] = df_results.loc[is_default]["
    config_type"].apply(rename_default)
datasets = list(df_results["dataset"].unique())
folds = list(df_results["fold"].unique())
config_types = list(df_results["config_type"].unique())

# Load and prepare pre-computed results
pre_df_results, _, _, _ = load_paper_results()
pre_df_results = pre_df_results[pre_df_results["fold"].isin(folds) &
    pre_df_results["dataset"].isin(datasets)]
df_results = PaperRunTabArena.compute_normalized_error_dynamic(
    df_results=pd.concat([df_results, pre_df_results], ignore_index=
    True))

# Create and save the leaderboard figure and table to the ./
    example_artifacts/ directory.
plotter.eval(df_results=df_results, framework_types_extra=config_types
    )
```

D.2 Contributing Models

To include a new model in `TabArena`, we ask users to open an issue on the `TabArena` benchmarking code repository (tabarena.ai/code) to start the process of adding a model. We envision this process not as a static request but as an ongoing interaction between the contributors and maintainers. During this process, the goal is to populate the issue over time with the information necessary to integrate a model. We require the following information to include a new model:

1. **Public Model Implementation.** The model must be implemented in the `AbstractModel` framework (see Appendix C.1), the code for this implementation must be publicly shared, and it must pass the default unit test for `TabArena` models. The implementation should represent a standalone model and not, for example, an ensembling pipeline of several existing models or sub-calls to other machine learning systems. We leave benchmarking for such pipelines, or in general, machine learning systems, to future iterations of `TabArena`. Finally, note that the model can also first be implemented in a scikit-learn API-like interface and then wrapped with the `AbstractModel` framework. This would be the recommended workflow in many cases.

2. **Preprocessing and Hyperparameters.** The implementation should specify model-specific preprocessing (see Appendix C.1). Moreover, the contributor must recommend default hyperparameters and a search space for hyperparameter optimization.

3. **Model Verification.** The maintainers of `TabArena` must have reviewed the source code of the model. In an ideal process, this review could also help the user to improve their model and implementation. In addition, the model should demonstrate promising results on `TabArena-Lite`. Moreover, if the contributor is not among the original authors of the model, the contributor (potentially in coordination with the maintainers of `TabArena`) shall reach out to the original authors to verify the implementation and its optimal intended usage. This may involve including the original author in GitHub issues, reviewing the pull request, or validating the results.

4. **Maintenance Commitment.** While the `TabArena` team generally maintains model implementations, we might need help from the original contributors to resolve future version conflicts or outdated functionality. Therefore, contributors must share their preferred way of being contacted. Note that the `TabArena` team may deprecate models that are no longer maintainable, consistently outperformed by newer models, or have bugs that cannot be reasonably resolved.

Once the issue is deemed finalized, two maintainers of `TabArena` need to review and approve the issue to complete the model integration.

D.3 Contributing Data: New Datasets and Curation Feedback

We envision `TabArena` to be a platform for discussing benchmarking practices. Therefore, we invite users, researchers, and practitioners to challenge our curation decisions or provide curation feedback using GitHub issues in the TabArena curation repository: tabarena.ai/data-tabular-ml-iid-study. Moreover, we also invite the community to add new datasets. For a new dataset to be added to `TabArena`, we outline the current template below:

1. Reference to pull request with a .yaml file including a dataset description following the template in the repository.
2. Reference to a .py file containing a preprocessing pipeline to transform data from the raw data source into a format suitable for benchmarking.
3. A checklist answering the following questions
   (a) Is the data available through an API for automatic downloading, or does the license allow for reuploading the data?
   (b) What is the sample size?
   (c) Was the data extracted from another modality (i.e., text, image, time-series)
      If yes: Are tabular learning methods a reasonable solution compared to domain-specific methods? (If possible, provide a reference)
   (d) Is there a deterministic function for optimally mapping the features to the target?
   (e) Was the data generated artificially or from a parameterized simulation?
   (f) Can you provide a one-sentence user story detailing the benefits of better predictive performance in this task?
   (g) Were the samples collected over time?
      If yes: Is the task about predicting future data, and, if yes, are there distribution shifts for samples collected later?
   (h) Were the samples collected in different groups (i.e., transactions from different customers, patients from multiple hospitals, repeated experimental results from different batches)?
      If yes: Is the task about predicting samples from unseen groups, and if yes, are distribution shifts of samples from different groups expected?
   (i) Are there known preprocessing techniques already applied to the 'rawest' available data version?
   (j) What preprocessing steps are recommended to conceptualize the task in the preprocessing Python file?
   (k) Do you have any other recommendations for how to use the dataset for benchmarking?

The maintainers will verify the provided information and engage in review-like discussions if necessary. After verifying that the task is reasonable, the dataset will be included in the next benchmark version.
The above protocol outlines the user-driven process for adding new datasets. However, we welcome any suggestions for datasets that could be included in future versions of `TabArena` and where we, as maintainers, drive the process to add the dataset. For that, we welcome GitHub issues with the 'Dataset Suggestion' template, which includes: (1) a link to the raw data, and (2) the dataset license. The `TabArena` maintainers will review the suggested dataset by applying the outlined protocol and, if the criteria are met, include it in the next version of `TabArena`.

The checklist results from our learnings during data curation and covers the essential aspects where we had to look closely at the data in our curation process. However, we want to emphasize that we do not generally exclude datasets using this checklist. On the contrary, for future versions of `TabArena`, we aim to explicitly extend the benchmark with tasks that are not covered sufficiently so far, either due to a lack of high-quality data or due to a lack of domain knowledge to judge the task quality on our end. Therefore, **we encourage users to propose datasets from other domains, non-IID data, and for any supervised learning task consisting of tabular features where strong performance is a desired property**.

D.3.1 Checklist Examples

In the following, we provide examples of the application of our checklist to one included and one excluded dataset.

Example for the APSFailure dataset, which represents one of the borderline cases that were included in `TabArena-v0.1`:

a) Is the data available through an API for automatic downloading, or does the license allow for reuploading the data? **Yes**

b) What is the sample size? **76,000**

c) Was the data extracted from another modality (i.e., text, image, time-series)? **Unclear, as the data was anonymized. Some features represent histograms, so some of the features possibly were extracted from time-series.**

   If yes: Are tabular learning methods a reasonable solution compared to domain-specific methods? (If possible, provide reference) **The data is from a 2016 challenge and was provided by a well-known company. Given that the dataset is comparably recent and the source is legitimate, we conclude that it still represents a meaningful tabular data task.**

d) Is there a deterministic function for optimally mapping the features to the target? **No**

e) Was the data generated artificially or from a parameterized simulation? **No**

f) Can you provide a one-sentence user story detailing the benefits of a better predictive performance in this task? **By automatically detecting component failures in trucks, the company can save costly manual effort and prevent accidents from releasing trucks with faulty components.**

g) Were the samples collected over time? **Probably yes.**

   If yes: Is the task about predicting future data, and, if yes, are there distribution shifts for samples collected later? **In a real application, future data would be predicted, however, the provided test dataset revealed that no distribution shifts between train and test data can be expected as the features are time-invariant.**

h) Were the samples collected in different groups (i.e. transactions from different customers, patients from multiple hospitals, repeated experimental results from different batches)? **No**

   If yes: Is the task about predicting samples from unseen groups, and if yes, are distribution shifts of samples from different groups expected? **N/A**

i) Are there known preprocessing techniques that have already been applied to the 'rawest' available data version? **The feature names were anonymized. Some features were preprocessed.**

j) What preprocessing steps are recommended to conceptualize the task in the preprocessing Python file? **Combine the original training and test files. Convert "na" strings to real NaN/missing values for numeric features.**

k) Do you have any other recommendations for how to use the dataset for benchmarking? **The data originally comes with a cost-matrix, which could be considered in future benchmark versions.**

Example for the socmob dataset, which was excluded for `TabArena-v0.1` as it represents a scientific discovery task where higher predictive performance is not relevant:

a) Is the data available through an API for automatic downloading, or does the license allow for reuploading the data? **Yes**

b) What is the sample size? **1156**

c) Was the data extracted from another modality (i.e., text, image, time-series) **No**

   If yes: Are tabular learning methods a reasonable solution compared to domain-specific methods? (If possible, provide reference) **N/A**

d) Is there a deterministic function for optimally mapping the features to the target? **No**

e) Was the data generated artificially or from a parameterized simulation? **No**

f) Can you provide a one-sentence user story detailing the benefits of a better predictive performance in this task? **No. The data was collected to empirically test the hypothesis that associations between socioeconomic and occupational attributes of fathers and sons among sons from intact families are stronger than associations between attributes of fathers and sons among sons from any kind of disrupted or reconstituted families. The dataset has one target and five predictive features, including the investigated family structure. Although supervised (linear) models are applied to the data, the goal is not to maximize performance, but to empirically quantify the relationship of one feature to the target while controlling for confounding factors (other features).**

g) Were the samples collected over time? **No, the study was cross-sectional and collected data in 1973.**

   If yes: Is the task about predicting future data, and, if yes, are there distribution shifts for samples collected later? **N/A**

h) Were the samples collected in different groups (i.e. transactions from different customers, patients from multiple hospitals, repeated experimental results from different batches)? **No**

   If yes: Is the task about predicting samples from unseen groups, and if yes, are distribution shifts of samples from different groups expected? **N/A**

i) Are there known preprocessing techniques that have already been applied to the 'rawest' available data version? **No noteworthy steps.**

j) What preprocessing steps are recommended to conceptualize the task in the preprocessing Python file? **None.**

k) Do you have any other recommendations for how to use the dataset for benchmarking? **Do not use the data for benchmarking the capabilities of predictive modeling approaches, but maybe for a scientific discovery benchmark in the future.**

D.4 Contributing Results: Leaderboard Submissions

We seek to define a process for `TabArena` to submit to the leaderboard that satisfies the following principles: (**1**) Equality: Submitting to the leaderboard is accessible in the same way to everyone. (**2**) Transparency: All attempts to submit to the leaderboard are transparent to the public. (**3**) Reproducibility: Submitted results are reproducible. (**4**) Fairness: Cheated results, i.e., by utilizing the test data in an inappropriate way or simply by submitting manually altered results, are rejected. (**5**) : Feasibility: The submission process, in particular the validation, must be manageable for the maintainers in a reasonable amount of time.

Using these guiding principles, we define our submission process:

1. To submit results to the leaderboard, users can write a pull request to tabarena.ai/community-results that contains:
   (a) An update to the results dataset collection with new data for their model.
   (b) Reproducible and documented code to obtain the results. We require users to start the process to add their new model to `TabArena` (as described in Appendix D.2) and to train and evaluate their approach using the provided `TabArena` benchmarking code.
   (c) A description or link to a description, e.g., a paper, for the new model.
   (d) The following statement: "I confirm that these results were produced using the attached modeling pipeline and to the best of my knowledge, I have used the test data appropriately and have not manipulated the results."
   (e) Indicate whether verification of the submitted results by the maintainers of `TabArena` is requested.
2. The maintainers will verify that all the required information is present and will proceed depending on whether verification was requested:

(a) Non-verified submission (fast): The request will be merged without recomputing the results. Non-verified submissions will not appear on the landing page and will be presented as a separate leaderboard on tabarena.ai[7].

(b) Verified submission: The maintainers will manually review the code and reproduce the results for a random sample of outer folds from different datasets. If the results can be reproduced successfully and no further issues are found, the request will be merged and the results will appear in the main `TabArena` leaderboard.

We aim to continuously improve our submission process and welcome any feedback or suggestions for future versions of `TabArena`.

D.5 Running TabArena Models in Practice

Models integrated into `TabArena` can be easily used to solve predictive machine learning tasks on new datasets, independent of the `TabArena` benchmark. Listing 5 shows how to run RealMLP on a toy dataset from scikit-learn. For more details on this code, please see our code repositories with examples: tabarena.ai/code-examples.

Listing 5: Running RealMLP from TabArena on a new dataset.

```python
from autogluon.core.data import LabelCleaner
from autogluon.features.generators import
    AutoMLPipelineFeatureGenerator
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split

# Import a TabArena model
from tabrepo.benchmark.models.ag.realmlp.realmlp_model import
    RealMLPModel

# Get Data
X, y = load_breast_cancer(return_X_y=True, as_frame=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.5, random_state=42)

# Model-agnostic Preprocessing
feature_generator, label_cleaner = AutoMLPipelineFeatureGenerator(),
    LabelCleaner.construct(problem_type="binary", y=y)
X_train, y_train = feature_generator.fit_transform(X_train),
    label_cleaner.transform(y_train)
X_test, y_test = feature_generator.transform(X_test), label_cleaner.
    transform(y_test)

# Train TabArena Model
clf = RealMLPModel()
clf.fit(X=X_train, y=y_train)

# Predict and score
prediction_probabilities = clf.predict_proba(X=X_test)
print("ROC AUC:", roc_auc_score(y_test, prediction_probabilities))
```

E Performance Results Per Dataset

Appendix E presents the performance per dataset for all methods in `TabArena`-v0.1.

[7] Note that for `TabArena`-v0.1 no non-validated leaderboard exists on the website. This will change with the first submission from the community using this protocol.

Table E.1: **Performance Per Dataset.** We show the average predictive performance per dataset with the standard deviation over folds. We show the performance for the default hyperparameter configuration (`Default`), for the model after tuning (`Tuned`), and for the ensemble after tuning (`Tuned + Ens.`). We highlight the best-performing methods with significance on three levels: (1) Green: The best performing method on average; (2) **Bold**: Methods that are not significantly worse than the best method on average, based on a Wilcoxon Signed-Rank test for paired samples with Holm-Bonferroni correction and $\alpha = 0.05$. (3) Underlined: Methods that are not significantly worse than the best method in the same pipeline regime (`Default`, `Tuned`, or `Tuned + Ens.`), based on a Wilcoxon Signed-Rank test for paired samples with Holm-Bonferroni correction and $\alpha = 0.05$. We exclude AutoGluon for significance tests in the `Tuned + Ens.` regime.

APSFailure (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.990 ± 0.002	0.990 ± 0.002	0.990 ± 0.002
ExtraTrees	0.990 ± 0.002	0.990 ± 0.003	0.991 ± 0.002
XGBoost	**0.992 ± 0.002**	**0.992 ± 0.002**	**0.992 ± 0.002**
LightGBM	0.992 ± 0.002	0.992 ± 0.002	0.992 ± 0.002
CatBoost	**0.992 ± 0.003**	0.992 ± 0.002	**0.992 ± 0.003**
EBM	0.991 ± 0.002	0.991 ± 0.002	0.991 ± 0.002
FastAIMLP	0.988 ± 0.003	0.989 ± 0.002	0.991 ± 0.002
TorchMLP	0.990 ± 0.002	0.991 ± 0.002	**0.992 ± 0.001**
RealMLP	0.991 ± 0.002	0.991 ± 0.002	0.992 ± 0.002
TabM	0.990 ± 0.002	0.991 ± 0.002	0.992 ± 0.002
MNCA	0.990 ± 0.002	0.990 ± 0.002	0.992 ± 0.003
TabPFNv2	-	-	-
TabDPT	0.990 ± 0.003	-	-
TabICL	**0.993 ± 0.002**	-	-
Linear	0.988 ± 0.002	0.987 ± 0.003	0.989 ± 0.001
KNN	0.910 ± 0.011	0.975 ± 0.004	0.979 ± 0.004
AutoGluon	-	-	**0.993 ± 0.002**

Amazon_employee_access (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.839 ± 0.005	0.841 ± 0.005	0.849 ± 0.005
ExtraTrees	0.833 ± 0.006	0.841 ± 0.007	0.845 ± 0.006
XGBoost	0.834 ± 0.007	0.859 ± 0.008	0.862 ± 0.008
LightGBM	0.843 ± 0.009	0.850 ± 0.007	0.858 ± 0.009
CatBoost	0.882 ± 0.008	**0.883 ± 0.008**	**0.883 ± 0.007**
EBM	0.839 ± 0.006	0.841 ± 0.007	0.842 ± 0.006
FastAIMLP	0.854 ± 0.007	0.853 ± 0.008	0.866 ± 0.007
TorchMLP	0.835 ± 0.007	0.838 ± 0.006	0.849 ± 0.007
RealMLP	0.844 ± 0.007	0.846 ± 0.008	0.864 ± 0.008
TabM	0.827 ± 0.008	0.844 ± 0.007	0.848 ± 0.007
MNCA	0.846 ± 0.008	0.861 ± 0.008	0.868 ± 0.007
TabPFNv2	-	-	-
TabDPT	0.841 ± 0.006	-	-
TabICL	0.854 ± 0.006	-	-
Linear	0.848 ± 0.009	0.848 ± 0.009	0.851 ± 0.008
KNN	0.839 ± 0.005	0.839 ± 0.005	0.839 ± 0.005
AutoGluon	-	-	**0.882 ± 0.005**

Another-Dataset-on-used-Fiat-500 (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	750.8 ± 28.4	736.5 ± 24.8	735.4 ± 25.3
ExtraTrees	744.2 ± 29.5	735.8 ± 26.5	735.1 ± 26.7
XGBoost	754.6 ± 23.0	741.4 ± 22.2	737.1 ± 22.6
LightGBM	746.0 ± 22.4	740.4 ± 24.6	729.4 ± 22.7
CatBoost	738.1 ± 20.8	736.3 ± 21.8	732.9 ± 21.9
EBM	749.9 ± 22.9	750.0 ± 24.1	745.7 ± 23.6
FastAIMLP	760.5 ± 17.6	761.2 ± 21.6	756.3 ± 21.4
TorchMLP	775.0 ± 26.3	769.8 ± 25.0	765.0 ± 24.7
RealMLP	757.4 ± 23.8	756.0 ± 22.4	**726.6 ± 24.0**
TabM	752.5 ± 23.8	755.3 ± 23.6	747.5 ± 23.1
MNCA	753.5 ± 25.0	752.4 ± 26.8	736.3 ± 27.2
TabPFNv2	727.7 ± 23.8	733.2 ± 27.2	**727.4 ± 26.0**
TabDPT	**724.0 ± 22.1**	-	-
TabICL	-	-	-
Linear	793.8 ± 25.4	764.7 ± 19.5	764.9 ± 19.6
KNN	882.0 ± 23.7	862.4 ± 30.5	852.8 ± 28.9
AutoGluon	-	-	729.6 ± 24.6

Bank_Customer_Churn (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.851 ± 0.008	0.857 ± 0.008	0.858 ± 0.008
ExtraTrees	0.851 ± 0.006	0.860 ± 0.007	0.861 ± 0.007
XGBoost	0.864 ± 0.009	0.866 ± 0.010	0.866 ± 0.010
LightGBM	0.864 ± 0.009	0.866 ± 0.009	0.867 ± 0.009
CatBoost	0.870 ± 0.009	0.870 ± 0.009	0.870 ± 0.009
EBM	0.862 ± 0.010	0.864 ± 0.010	0.864 ± 0.010
FastAIMLP	0.859 ± 0.009	0.863 ± 0.007	0.864 ± 0.008
TorchMLP	0.860 ± 0.008	0.866 ± 0.008	0.866 ± 0.008
RealMLP	0.866 ± 0.009	0.870 ± 0.009	0.871 ± 0.009
TabM	0.870 ± 0.010	0.871 ± 0.009	0.871 ± 0.010
MNCA	0.867 ± 0.009	0.868 ± 0.009	0.869 ± 0.008
TabPFNv2	0.872 ± 0.009	**0.874 ± 0.009**	**0.874 ± 0.008**
TabDPT	0.859 ± 0.008	-	-
TabICL	0.868 ± 0.009	-	-
Linear	0.772 ± 0.010	0.773 ± 0.010	0.773 ± 0.010
KNN	0.526 ± 0.005	0.558 ± 0.009	0.557 ± 0.008
AutoGluon	-	-	0.869 ± 0.009

Bioresponse (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.873 ± 0.007	0.873 ± 0.007	0.876 ± 0.006
ExtraTrees	0.867 ± 0.008	0.868 ± 0.009	0.871 ± 0.008
XGBoost	0.873 ± 0.008	0.875 ± 0.008	**0.876 ± 0.008**
LightGBM	0.872 ± 0.008	0.874 ± 0.007	0.875 ± 0.008
CatBoost	0.872 ± 0.009	0.875 ± 0.008	0.874 ± 0.008
EBM	0.852 ± 0.009	0.863 ± 0.008	0.866 ± 0.008
FastAIMLP	0.850 ± 0.011	0.857 ± 0.010	0.860 ± 0.010
TorchMLP	0.846 ± 0.008	0.856 ± 0.009	0.863 ± 0.008
RealMLP	0.858 ± 0.009	0.864 ± 0.008	**0.875 ± 0.008**
TabM	0.863 ± 0.005	0.870 ± 0.006	0.872 ± 0.006
MNCA	0.862 ± 0.009	0.867 ± 0.007	0.874 ± 0.008
TabPFNv2	-	-	-
TabDPT	0.862 ± 0.010	-	-
TabICL	-	-	-
Linear	0.789 ± 0.011	0.789 ± 0.011	0.793 ± 0.013
KNN	0.805 ± 0.009	0.840 ± 0.010	0.849 ± 0.009
AutoGluon	-	-	**0.878 ± 0.007**

Diabetes130US (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.631 ± 0.006	0.656 ± 0.008	0.657 ± 0.008
ExtraTrees	0.623 ± 0.009	0.651 ± 0.007	0.653 ± 0.008
XGBoost	0.662 ± 0.008	0.668 ± 0.008	0.670 ± 0.008
LightGBM	0.648 ± 0.008	0.668 ± 0.007	**0.672 ± 0.008**
CatBoost	0.671 ± 0.008	0.671 ± 0.008	0.672 ± 0.008
EBM	0.659 ± 0.008	0.662 ± 0.007	0.665 ± 0.008
FastAIMLP	0.647 ± 0.007	0.656 ± 0.008	0.662 ± 0.008
TorchMLP	0.655 ± 0.007	0.663 ± 0.009	0.667 ± 0.009
RealMLP	0.659 ± 0.005	0.662 ± 0.008	0.669 ± 0.008
TabM	0.660 ± 0.007	0.662 ± 0.008	0.663 ± 0.008
MNCA	0.657 ± 0.008	0.662 ± 0.009	0.666 ± 0.008
TabPFNv2	-	-	-
TabDPT	0.609 ± 0.008	-	-
TabICL	0.647 ± 0.008	-	-
Linear	0.648 ± 0.008	0.648 ± 0.008	0.648 ± 0.008
KNN	0.516 ± 0.004	0.539 ± 0.005	0.541 ± 0.005
AutoGluon	-	-	**0.673 ± 0.008**

E-CommereShippingData (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.739 ± 0.005	**0.740 ± 0.006**	**0.741 ± 0.005**
ExtraTrees	0.737 ± 0.006	0.741 ± 0.006	0.740 ± 0.005
XGBoost	0.740 ± 0.006	**0.742 ± 0.007**	**0.742 ± 0.006**
LightGBM	0.739 ± 0.006	**0.740 ± 0.005**	**0.741 ± 0.006**
CatBoost	**0.744 ± 0.006**	**0.742 ± 0.007**	0.741 ± 0.007
EBM	**0.744 ± 0.004**	**0.743 ± 0.005**	**0.743 ± 0.004**
FastAIMLP	0.737 ± 0.005	0.741 ± 0.007	**0.741 ± 0.007**
TorchMLP	0.737 ± 0.008	0.740 ± 0.007	**0.741 ± 0.007**
RealMLP	**0.741 ± 0.007**	**0.742 ± 0.007**	**0.742 ± 0.007**
TabM	**0.744 ± 0.007**	0.740 ± 0.006	**0.743 ± 0.007**
MNCA	**0.743 ± 0.008**	**0.743 ± 0.005**	**0.742 ± 0.005**
TabPFNv2	**0.744 ± 0.007**	**0.744 ± 0.007**	**0.744 ± 0.006**
TabDPT	0.735 ± 0.007	-	-
TabICL	**0.743 ± 0.006**	-	-
Linear	0.704 ± 0.006	0.721 ± 0.008	0.722 ± 0.008
KNN	0.731 ± 0.005	0.737 ± 0.007	0.738 ± 0.007
AutoGluon	-	-	**0.738 ± 0.005**

Fitness_Club (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.775 ± 0.018	0.801 ± 0.015	0.800 ± 0.015
ExtraTrees	0.769 ± 0.018	0.816 ± 0.013	0.815 ± 0.014
XGBoost	0.798 ± 0.015	0.808 ± 0.015	0.808 ± 0.015
LightGBM	0.795 ± 0.015	0.815 ± 0.015	0.814 ± 0.015
CatBoost	0.814 ± 0.014	0.812 ± 0.015	0.811 ± 0.015
EBM	0.813 ± 0.015	0.810 ± 0.015	0.812 ± 0.015
FastAIMLP	0.806 ± 0.013	0.814 ± 0.013	0.814 ± 0.014
TorchMLP	0.813 ± 0.016	0.813 ± 0.017	0.814 ± 0.016
RealMLP	0.812 ± 0.015	0.814 ± 0.015	0.816 ± 0.014
TabM	0.818 ± 0.014	0.817 ± 0.014	0.818 ± 0.014
MNCA	0.818 ± 0.014	0.814 ± 0.015	0.799 ± 0.016
TabPFNv2	**0.822 ± 0.012**	0.820 ± 0.013	0.817 ± 0.013
TabDPT	0.818 ± 0.014	-	-
TabICL	0.819 ± 0.013	-	-
Linear	0.819 ± 0.014	0.818 ± 0.015	0.818 ± 0.015
KNN	0.733 ± 0.019	0.802 ± 0.014	0.803 ± 0.014
AutoGluon	-	-	0.811 ± 0.012

Food_Delivery_Time (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	7.855 ± 0.041	7.587 ± 0.046	7.588 ± 0.046
ExtraTrees	8.179 ± 0.045	7.753 ± 0.053	7.749 ± 0.053
XGBoost	7.397 ± 0.055	7.397 ± 0.055	7.400 ± 0.055
LightGBM	7.616 ± 0.053	7.378 ± 0.054	7.374 ± 0.053
CatBoost	7.379 ± 0.051	**7.367 ± 0.051**	**7.368 ± 0.051**
EBM	7.433 ± 0.040	7.424 ± 0.051	7.412 ± 0.044
FastAIMLP	8.188 ± 0.053	8.085 ± 0.056	8.060 ± 0.052
TorchMLP	7.735 ± 0.059	7.579 ± 0.049	7.559 ± 0.052
RealMLP	7.926 ± 0.053	7.453 ± 0.051	7.414 ± 0.046
TabM	7.762 ± 0.049	7.651 ± 0.053	7.651 ± 0.052
MNCA	7.487 ± 0.046	7.421 ± 0.044	7.390 ± 0.046
TabPFNv2	-	-	-
TabDPT	7.551 ± 0.042	-	-
TabICL	-	-	-
Linear	8.563 ± 0.047	8.325 ± 0.065	8.271 ± 0.055
KNN	8.552 ± 0.047	8.256 ± 0.042	8.162 ± 0.049
AutoGluon	-	-	**7.362 ± 0.051**

GiveMeSomeCredit (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.846 ± 0.003	0.862 ± 0.003	0.863 ± 0.003
ExtraTrees	0.840 ± 0.003	0.857 ± 0.002	0.857 ± 0.003
XGBoost	0.865 ± 0.002	0.866 ± 0.002	0.866 ± 0.002
LightGBM	0.865 ± 0.002	0.866 ± 0.002	0.867 ± 0.002
CatBoost	0.866 ± 0.002	0.867 ± 0.002	0.867 ± 0.002
EBM	0.864 ± 0.002	0.865 ± 0.002	0.865 ± 0.002
FastAIMLP	0.829 ± 0.004	0.843 ± 0.005	0.847 ± 0.003
TorchMLP	0.863 ± 0.002	0.864 ± 0.002	0.865 ± 0.002
RealMLP	0.865 ± 0.002	0.866 ± 0.002	0.866 ± 0.002
TabM	0.866 ± 0.002	**0.867 ± 0.002**	**0.867 ± 0.002**
MNCA	0.865 ± 0.002	0.866 ± 0.002	0.866 ± 0.002
TabPFNv2	-	-	-
TabDPT	0.842 ± 0.003	-	-
TabICL	0.866 ± 0.002	-	-
Linear	0.841 ± 0.002	0.841 ± 0.002	0.841 ± 0.002
KNN	0.570 ± 0.003	0.672 ± 0.003	0.673 ± 0.003
AutoGluon	-	-	**0.867 ± 0.002**

HR_Analytics_Job_Change (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.789 ± 0.006	0.802 ± 0.006	0.802 ± 0.006
ExtraTrees	0.784 ± 0.007	0.800 ± 0.007	0.801 ± 0.007
XGBoost	**0.805 ± 0.006**	0.803 ± 0.005	**0.805 ± 0.006**
LightGBM	0.802 ± 0.007	0.803 ± 0.007	0.804 ± 0.006
CatBoost	**0.804 ± 0.006**	**0.804 ± 0.006**	0.804 ± 0.006
EBM	0.800 ± 0.006	0.800 ± 0.006	0.801 ± 0.006
FastAIMLP	0.801 ± 0.005	0.801 ± 0.007	0.803 ± 0.007
TorchMLP	0.801 ± 0.006	0.801 ± 0.006	0.803 ± 0.006
RealMLP	0.801 ± 0.007	0.801 ± 0.006	0.803 ± 0.006
TabM	0.801 ± 0.007	0.802 ± 0.006	0.803 ± 0.006
MNCA	0.801 ± 0.006	0.802 ± 0.007	0.803 ± 0.007
TabPFNv2	-	-	-
TabDPT	0.801 ± 0.006	-	-
TabICL	**0.805 ± 0.006**	-	-
Linear	0.796 ± 0.006	0.796 ± 0.006	0.796 ± 0.006
KNN	0.605 ± 0.009	0.663 ± 0.003	0.672 ± 0.003
AutoGluon	-	-	**0.805 ± 0.007**

Is-this-a-good-customer (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.721 ± 0.020	0.727 ± 0.018	0.729 ± 0.020
ExtraTrees	0.695 ± 0.021	0.713 ± 0.022	0.718 ± 0.024
XGBoost	0.723 ± 0.021	0.742 ± 0.023	0.744 ± 0.022
LightGBM	0.724 ± 0.020	0.741 ± 0.022	**0.746 ± 0.020**
CatBoost	**0.748 ± 0.020**	0.743 ± 0.019	0.744 ± 0.019
EBM	**0.751 ± 0.019**	0.745 ± 0.018	0.748 ± 0.018
FastAIMLP	0.711 ± 0.018	**0.742 ± 0.025**	0.745 ± 0.017
TorchMLP	0.728 ± 0.020	0.727 ± 0.023	0.733 ± 0.018
RealMLP	0.732 ± 0.023	0.731 ± 0.025	0.742 ± 0.020
TabM	0.743 ± 0.021	0.742 ± 0.021	0.743 ± 0.019
MNCA	0.739 ± 0.022	0.729 ± 0.025	0.705 ± 0.022
TabPFNv2	**0.746 ± 0.019**	0.735 ± 0.022	0.743 ± 0.018
TabDPT	0.742 ± 0.016	-	-
TabICL	0.744 ± 0.019	-	-
Linear	0.738 ± 0.021	0.737 ± 0.024	0.736 ± 0.023
KNN	0.498 ± 0.026	0.534 ± 0.034	0.534 ± 0.028
AutoGluon	-	-	0.745 ± 0.019

MIC (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.513 ± 0.031	0.485 ± 0.023	0.474 ± 0.019
ExtraTrees	0.521 ± 0.030	0.482 ± 0.020	0.470 ± 0.018
XGBoost	0.470 ± 0.020	0.440 ± 0.019	0.440 ± 0.019
LightGBM	0.503 ± 0.019	0.453 ± 0.020	0.453 ± 0.019
CatBoost	0.455 ± 0.020	0.453 ± 0.019	0.451 ± 0.018
EBM	0.475 ± 0.018	0.446 ± 0.016	0.445 ± 0.016
FastAIMLP	0.506 ± 0.024	0.462 ± 0.023	0.450 ± 0.020
TorchMLP	0.473 ± 0.019	0.465 ± 0.024	0.453 ± 0.017
RealMLP	0.492 ± 0.027	0.439 ± 0.021	0.434 ± 0.017
TabM	**0.432 ± 0.017**	0.432 ± 0.016	**0.430 ± 0.016**
MNCA	0.465 ± 0.019	0.456 ± 0.018	0.450 ± 0.019
TabPFNv2	0.468 ± 0.043	0.440 ± 0.022	**0.433 ± 0.022**
TabDPT	0.481 ± 0.021	-	-
TabICL	0.465 ± 0.022	-	-
Linear	0.589 ± 0.035	0.589 ± 0.035	0.589 ± 0.035
KNN	2.040 ± 0.075	1.105 ± 0.096	1.019 ± 0.096
AutoGluon	-	-	0.445 ± 0.018

Marketing_Campaign (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.883 ± 0.015	0.881 ± 0.016	0.882 ± 0.015
ExtraTrees	0.884 ± 0.015	0.886 ± 0.015	0.888 ± 0.015
XGBoost	0.897 ± 0.015	0.903 ± 0.016	0.904 ± 0.015
LightGBM	0.901 ± 0.014	0.911 ± 0.015	0.911 ± 0.014
CatBoost	0.907 ± 0.015	0.904 ± 0.014	0.903 ± 0.015
EBM	0.903 ± 0.015	0.905 ± 0.015	0.906 ± 0.016
FastAIMLP	0.890 ± 0.017	0.905 ± 0.015	0.909 ± 0.014
TorchMLP	0.898 ± 0.015	0.910 ± 0.014	0.915 ± 0.013
RealMLP	0.906 ± 0.015	0.907 ± 0.014	0.911 ± 0.014
TabM	0.901 ± 0.015	**0.917 ± 0.013**	0.916 ± 0.014
MNCA	0.909 ± 0.017	0.912 ± 0.015	0.908 ± 0.016
TabPFNv2	0.915 ± 0.015	0.915 ± 0.013	**0.919 ± 0.013**
TabDPT	0.896 ± 0.016	-	-
TabICL	0.911 ± 0.013	-	-
Linear	0.906 ± 0.013	0.906 ± 0.013	0.905 ± 0.013
KNN	0.591 ± 0.021	0.615 ± 0.022	0.628 ± 0.020
AutoGluon	-	-	0.915 ± 0.013

NATICUSdroid (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.977 ± 0.002	0.981 ± 0.003	0.981 ± 0.003
ExtraTrees	0.977 ± 0.002	0.982 ± 0.002	0.982 ± 0.002
XGBoost	0.985 ± 0.002	0.985 ± 0.002	0.985 ± 0.002
LightGBM	0.985 ± 0.002	0.986 ± 0.002	0.986 ± 0.002
CatBoost	0.986 ± 0.002	0.986 ± 0.002	0.986 ± 0.002
EBM	0.984 ± 0.002	0.985 ± 0.001	0.985 ± 0.001
FastAIMLP	0.985 ± 0.002	0.985 ± 0.001	0.986 ± 0.001
TorchMLP	0.985 ± 0.002	0.985 ± 0.001	0.986 ± 0.002
RealMLP	0.985 ± 0.002	0.986 ± 0.001	0.986 ± 0.001
TabM	0.986 ± 0.001	0.986 ± 0.002	0.986 ± 0.001
MNCA	0.983 ± 0.002	0.984 ± 0.001	0.985 ± 0.002
TabPFNv2	0.983 ± 0.002	0.984 ± 0.003	0.985 ± 0.002
TabDPT	0.985 ± 0.002	-	-
TabICL	**0.987 ± 0.001**	-	-
Linear	0.981 ± 0.002	0.981 ± 0.002	0.981 ± 0.002
KNN	0.977 ± 0.002	0.977 ± 0.002	0.977 ± 0.002
AutoGluon	-	-	0.987 ± 0.002

QSAR-TID-11 (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	0.806 ± 0.047	0.805 ± 0.048	0.796 ± 0.046
ExtraTrees	0.806 ± 0.046	0.802 ± 0.046	0.791 ± 0.046
XGBoost	0.786 ± 0.049	0.761 ± 0.047	0.760 ± 0.048
LightGBM	0.772 ± 0.050	0.758 ± 0.049	0.756 ± 0.048
CatBoost	0.774 ± 0.049	0.773 ± 0.048	0.771 ± 0.049
EBM	0.872 ± 0.039	0.859 ± 0.042	0.853 ± 0.042
FastAIMLP	0.776 ± 0.045	0.766 ± 0.049	0.761 ± 0.050
TorchMLP	0.774 ± 0.052	0.762 ± 0.049	0.748 ± 0.054
RealMLP	0.763 ± 0.047	0.764 ± 0.049	0.754 ± 0.050
TabM	0.761 ± 0.050	0.760 ± 0.048	0.760 ± 0.048
MNCA	0.770 ± 0.044	0.746 ± 0.045	**0.735 ± 0.044**
TabPFNv2	-	-	-
TabDPT	0.773 ± 0.046	-	-
TabICL	-	-	-
Linear	1.020 ± 0.032	0.940 ± 0.040	0.937 ± 0.039
KNN	0.806 ± 0.047	0.806 ± 0.047	0.806 ± 0.047
AutoGluon	-	-	0.747 ± 0.049

QSAR_fish_toxicity (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	0.907 ± 0.047	0.885 ± 0.050	0.884 ± 0.049
ExtraTrees	0.880 ± 0.052	0.873 ± 0.055	0.870 ± 0.052
XGBoost	0.905 ± 0.050	0.881 ± 0.043	0.879 ± 0.042
LightGBM	0.894 ± 0.043	0.889 ± 0.045	0.883 ± 0.044
CatBoost	0.877 ± 0.049	0.875 ± 0.045	0.874 ± 0.047
EBM	0.905 ± 0.050	0.904 ± 0.048	0.898 ± 0.047
FastAIMLP	0.908 ± 0.048	0.909 ± 0.046	0.897 ± 0.048
TorchMLP	0.906 ± 0.055	0.897 ± 0.055	0.890 ± 0.055
RealMLP	0.878 ± 0.054	0.884 ± 0.058	0.865 ± 0.052
TabM	0.910 ± 0.046	0.896 ± 0.050	0.886 ± 0.047
MNCA	0.885 ± 0.054	0.886 ± 0.054	0.873 ± 0.052
TabPFNv2	0.868 ± 0.047	0.873 ± 0.051	**0.860 ± 0.049**
TabDPT	**0.859 ± 0.049**	-	-
TabICL	-	-	-
Linear	0.950 ± 0.056	0.950 ± 0.056	0.950 ± 0.056
KNN	0.962 ± 0.064	0.935 ± 0.061	0.919 ± 0.065
AutoGluon	-	-	0.880 ± 0.054

SDSS17 (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.085 ± 0.002	0.073 ± 0.002	0.072 ± 0.002
ExtraTrees	0.131 ± 0.002	0.080 ± 0.002	0.078 ± 0.002
XGBoost	0.074 ± 0.002	0.074 ± 0.002	0.074 ± 0.002
LightGBM	0.087 ± 0.002	0.073 ± 0.003	0.073 ± 0.002
CatBoost	0.075 ± 0.002	0.074 ± 0.003	0.074 ± 0.003
EBM	0.087 ± 0.002	0.081 ± 0.002	0.081 ± 0.002
FastAIMLP	0.134 ± 0.004	0.111 ± 0.004	0.112 ± 0.003
TorchMLP	0.094 ± 0.003	0.081 ± 0.002	0.080 ± 0.002
RealMLP	0.103 ± 0.002	0.089 ± 0.002	0.087 ± 0.002
TabM	0.096 ± 0.002	0.084 ± 0.002	0.084 ± 0.002
MNCA	0.085 ± 0.002	0.079 ± 0.002	0.076 ± 0.002
TabPFNv2	-	-	-
TabDPT	0.088 ± 0.001	-	-
TabICL	0.076 ± 0.002	-	-
Linear	0.146 ± 0.003	0.147 ± 0.003	0.139 ± 0.003
KNN	1.155 ± 0.030	0.512 ± 0.006	0.440 ± 0.006
AutoGluon	-	-	**0.067 ± 0.002**

airfoil_self_noise (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	1.898 ± 0.095	1.891 ± 0.101	1.851 ± 0.096
ExtraTrees	1.822 ± 0.094	1.676 ± 0.106	1.683 ± 0.105
XGBoost	1.549 ± 0.104	1.439 ± 0.104	1.443 ± 0.104
LightGBM	1.554 ± 0.093	1.480 ± 0.108	1.451 ± 0.108
CatBoost	1.583 ± 0.096	1.327 ± 0.101	1.330 ± 0.105
EBM	2.010 ± 0.117	1.955 ± 0.104	1.935 ± 0.113
FastAIMLP	2.327 ± 0.141	1.624 ± 0.100	1.646 ± 0.105
TorchMLP	1.493 ± 0.113	1.372 ± 0.105	1.374 ± 0.098
RealMLP	1.179 ± 0.085	1.146 ± 0.078	1.109 ± 0.081
TabM	1.253 ± 0.098	1.150 ± 0.092	1.139 ± 0.086
MNCA	1.553 ± 0.093	1.513 ± 0.110	1.454 ± 0.095
TabPFNv2	1.119 ± 0.088	1.112 ± 0.102	**1.074 ± 0.094**
TabDPT	1.203 ± 0.084	-	-
TabICL	-	-	-
Linear	5.344 ± 0.199	4.750 ± 0.140	4.743 ± 0.144
KNN	5.586 ± 0.184	5.458 ± 0.195	5.287 ± 0.163
AutoGluon	-	-	1.269 ± 0.090

anneal (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.046 ± 0.010	0.028 ± 0.025	0.023 ± 0.013
ExtraTrees	0.064 ± 0.012	0.028 ± 0.022	0.026 ± 0.022
XGBoost	0.039 ± 0.025	0.031 ± 0.025	0.031 ± 0.025
LightGBM	0.055 ± 0.026	0.033 ± 0.019	0.034 ± 0.019
CatBoost	0.040 ± 0.022	0.022 ± 0.021	0.021 ± 0.021
EBM	0.043 ± 0.025	0.036 ± 0.033	0.034 ± 0.032
FastAIMLP	0.085 ± 0.029	0.056 ± 0.022	0.054 ± 0.021
TorchMLP	0.040 ± 0.034	0.052 ± 0.044	0.040 ± 0.038
RealMLP	0.039 ± 0.031	0.029 ± 0.032	0.024 ± 0.026
TabM	0.036 ± 0.028	0.029 ± 0.025	0.028 ± 0.024
MNCA	0.045 ± 0.038	0.035 ± 0.035	0.035 ± 0.033
TabPFNv2	**0.016 ± 0.014**	0.023 ± 0.019	0.019 ± 0.014
TabDPT	0.058 ± 0.022	-	-
TabICL	0.028 ± 0.014	-	-
Linear	0.090 ± 0.028	0.047 ± 0.035	0.039 ± 0.028
KNN	1.064 ± 0.202	0.918 ± 0.209	0.531 ± 0.112
AutoGluon	-	-	0.037 ± 0.059

bank-marketing (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.726 ± 0.006	0.761 ± 0.005	0.761 ± 0.005
ExtraTrees	0.721 ± 0.004	0.758 ± 0.005	0.758 ± 0.005
XGBoost	0.763 ± 0.005	**0.765 ± 0.006**	**0.765 ± 0.005**
LightGBM	0.763 ± 0.005	**0.765 ± 0.005**	**0.766 ± 0.005**
CatBoost	**0.766 ± 0.005**	**0.766 ± 0.005**	**0.765 ± 0.005**
EBM	0.762 ± 0.005	0.763 ± 0.005	0.763 ± 0.005
FastAIMLP	0.759 ± 0.005	0.760 ± 0.006	0.761 ± 0.005
TorchMLP	0.757 ± 0.006	0.758 ± 0.006	0.759 ± 0.006
RealMLP	0.761 ± 0.005	0.763 ± 0.005	0.765 ± 0.005
TabM	0.764 ± 0.006	0.764 ± 0.006	**0.765 ± 0.005**
MNCA	0.763 ± 0.005	0.763 ± 0.005	0.764 ± 0.005
TabPFNv2	-	-	-
TabDPT	0.761 ± 0.005	-	-
TabICL	0.764 ± 0.005	-	-
Linear	0.748 ± 0.004	0.748 ± 0.004	0.748 ± 0.004
KNN	0.610 ± 0.005	0.650 ± 0.006	0.653 ± 0.007
AutoGluon	-	-	**0.765 ± 0.006**

blood-transfusion-service-center (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.682 ± 0.026	0.714 ± 0.029	0.713 ± 0.029
ExtraTrees	0.689 ± 0.025	0.728 ± 0.033	0.727 ± 0.031
XGBoost	0.708 ± 0.030	0.733 ± 0.031	0.731 ± 0.031
LightGBM	0.726 ± 0.033	0.743 ± 0.033	0.743 ± 0.031
CatBoost	0.738 ± 0.029	0.737 ± 0.031	0.736 ± 0.031
EBM	0.742 ± 0.032	0.743 ± 0.033	0.743 ± 0.033
FastAIMLP	0.743 ± 0.030	**0.754 ± 0.030**	**0.756 ± 0.030**
TorchMLP	0.749 ± 0.032	0.747 ± 0.030	0.748 ± 0.030
RealMLP	0.750 ± 0.030	0.737 ± 0.029	0.742 ± 0.027
TabM	0.741 ± 0.030	0.737 ± 0.028	0.741 ± 0.030
MNCA	**0.748 ± 0.033**	0.732 ± 0.033	0.715 ± 0.029
TabPFNv2	**0.755 ± 0.029**	0.748 ± 0.034	0.746 ± 0.030
TabDPT	0.751 ± 0.030	-	-
TabICL	0.737 ± 0.031	-	-
Linear	0.731 ± 0.032	0.747 ± 0.030	**0.755 ± 0.026**
KNN	0.661 ± 0.032	0.658 ± 0.036	0.680 ± 0.033
AutoGluon	-	-	0.748 ± 0.032

churn (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.915 ± 0.009	0.913 ± 0.012	0.913 ± 0.011
ExtraTrees	0.917 ± 0.011	0.919 ± 0.011	0.921 ± 0.011
XGBoost	0.923 ± 0.009	0.921 ± 0.011	0.920 ± 0.011
LightGBM	0.916 ± 0.011	0.920 ± 0.011	0.920 ± 0.011
CatBoost	**0.924 ± 0.011**	0.920 ± 0.012	0.922 ± 0.011
EBM	0.922 ± 0.014	0.924 ± 0.011	0.924 ± 0.011
FastAIMLP	0.918 ± 0.011	0.921 ± 0.009	0.921 ± 0.010
TorchMLP	0.888 ± 0.009	0.918 ± 0.010	0.918 ± 0.011
RealMLP	0.920 ± 0.010	0.924 ± 0.012	0.927 ± 0.011
TabM	0.925 ± 0.011	0.923 ± 0.010	0.923 ± 0.010
MNCA	**0.930 ± 0.010**	**0.930 ± 0.014**	**0.930 ± 0.012**
TabPFNv2	**0.928 ± 0.011**	**0.925 ± 0.008**	**0.924 ± 0.010**
TabDPT	0.923 ± 0.009	-	-
TabICL	0.924 ± 0.011	-	-
Linear	0.777 ± 0.018	0.816 ± 0.013	0.816 ± 0.013
KNN	0.681 ± 0.021	0.740 ± 0.018	0.739 ± 0.019
AutoGluon	-	-	0.922 ± 0.011

coil2000_insurance_policies (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.697 ± 0.014	0.741 ± 0.017	0.742 ± 0.016
ExtraTrees	0.696 ± 0.016	0.744 ± 0.016	0.748 ± 0.017
XGBoost	0.757 ± 0.015	0.757 ± 0.016	0.758 ± 0.014
LightGBM	0.752 ± 0.014	0.759 ± 0.015	0.761 ± 0.015
CatBoost	0.757 ± 0.014	0.758 ± 0.013	0.759 ± 0.012
EBM	0.754 ± 0.014	0.757 ± 0.013	0.761 ± 0.013
FastAIMLP	0.719 ± 0.010	0.749 ± 0.013	0.747 ± 0.013
TorchMLP	0.740 ± 0.011	0.747 ± 0.015	0.752 ± 0.014
RealMLP	0.742 ± 0.012	0.755 ± 0.011	0.763 ± 0.013
TabM	0.761 ± 0.013	0.764 ± 0.013	0.766 ± 0.012
MNCA	0.764 ± 0.016	0.759 ± 0.014	0.765 ± 0.013
TabPFNv2	0.753 ± 0.015	**0.773 ± 0.015**	**0.773 ± 0.014**
TabDPT	0.725 ± 0.010	-	-
TabICL	0.756 ± 0.012	-	-
Linear	0.737 ± 0.012	0.735 ± 0.011	0.736 ± 0.012
KNN	0.605 ± 0.010	0.676 ± 0.018	0.690 ± 0.009
AutoGluon	-	-	0.759 ± 0.016

concrete_compressive_strength (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	5.261 ± 0.336	5.189 ± 0.366	5.106 ± 0.352
ExtraTrees	5.139 ± 0.341	5.073 ± 0.333	5.048 ± 0.348
XGBoost	4.755 ± 0.387	4.236 ± 0.373	4.222 ± 0.384
LightGBM	4.484 ± 0.388	4.235 ± 0.395	4.212 ± 0.396
CatBoost	4.214 ± 0.413	4.231 ± 0.415	4.209 ± 0.411
EBM	4.442 ± 0.295	4.429 ± 0.331	4.371 ± 0.308
FastAIMLP	6.369 ± 0.379	5.187 ± 0.355	5.272 ± 0.360
TorchMLP	4.817 ± 0.354	4.715 ± 0.350	4.654 ± 0.334
RealMLP	4.688 ± 0.364	4.344 ± 0.289	**4.133 ± 0.329**
TabM	4.268 ± 0.378	**4.289 ± 0.635**	**4.150 ± 0.420**
MNCA	4.932 ± 0.350	4.705 ± 0.423	4.484 ± 0.390
TabPFNv2	4.259 ± 0.379	4.171 ± 0.439	**4.118 ± 0.409**
TabDPT	4.267 ± 0.422	-	-
TabICL	-	-	-
Linear	8.228 ± 0.359	8.159 ± 0.361	8.150 ± 0.354
KNN	9.556 ± 0.486	8.544 ± 0.533	8.419 ± 0.493
AutoGluon	-	-	**4.165 ± 0.389**

credit-g (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.783 ± 0.017	0.781 ± 0.019	0.782 ± 0.019
ExtraTrees	0.779 ± 0.019	0.781 ± 0.018	0.782 ± 0.018
XGBoost	0.783 ± 0.021	0.792 ± 0.021	0.793 ± 0.021
LightGBM	0.771 ± 0.019	0.792 ± 0.020	**0.796 ± 0.020**
CatBoost	0.789 ± 0.017	**0.795 ± 0.020**	**0.795 ± 0.017**
EBM	0.790 ± 0.021	0.782 ± 0.025	0.787 ± 0.023
FastAIMLP	0.783 ± 0.029	0.784 ± 0.025	**0.793 ± 0.023**
TorchMLP	0.772 ± 0.017	0.782 ± 0.020	0.788 ± 0.020
RealMLP	0.785 ± 0.022	0.784 ± 0.023	0.791 ± 0.020
TabM	**0.795 ± 0.021**	0.789 ± 0.021	**0.795 ± 0.021**
MNCA	0.789 ± 0.020	0.785 ± 0.021	0.775 ± 0.020
TabPFNv2	0.776 ± 0.019	0.773 ± 0.020	0.792 ± 0.020
TabDPT	0.780 ± 0.019	-	-
TabICL	0.790 ± 0.017	-	-
Linear	0.781 ± 0.021	0.780 ± 0.022	0.780 ± 0.021
KNN	0.558 ± 0.024	0.569 ± 0.027	0.579 ± 0.029
AutoGluon	-	-	**0.794 ± 0.020**

credit_card_clients_default (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.765 ± 0.005	0.780 ± 0.004	0.780 ± 0.004
ExtraTrees	0.766 ± 0.004	0.781 ± 0.004	0.782 ± 0.004
XGBoost	0.783 ± 0.004	0.785 ± 0.004	0.785 ± 0.004
LightGBM	0.784 ± 0.004	0.785 ± 0.004	0.785 ± 0.004
CatBoost	0.784 ± 0.004	0.785 ± 0.004	0.785 ± 0.004
EBM	0.783 ± 0.004	0.783 ± 0.004	0.784 ± 0.004
FastAIMLP	0.781 ± 0.005	0.783 ± 0.005	0.783 ± 0.005
TorchMLP	0.779 ± 0.003	0.783 ± 0.003	0.785 ± 0.003
RealMLP	0.785 ± 0.004	0.785 ± 0.005	0.786 ± 0.004
TabM	0.784 ± 0.004	**0.788 ± 0.004**	**0.788 ± 0.004**
MNCA	0.786 ± 0.003	**0.787 ± 0.004**	**0.787 ± 0.004**
TabPFNv2	-	-	-
TabDPT	0.780 ± 0.004	-	-
TabICL	**0.788 ± 0.004**	-	-
Linear	0.745 ± 0.004	0.745 ± 0.004	0.745 ± 0.004
KNN	0.605 ± 0.004	0.659 ± 0.005	0.666 ± 0.004
AutoGluon	-	-	**0.787 ± 0.004**

customer_satisfaction_in_airline (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.993 ± 0.000	0.993 ± 0.000	0.993 ± 0.000
ExtraTrees	0.992 ± 0.000	0.994 ± 0.000	0.994 ± 0.000
XGBoost	0.994 ± 0.000	0.994 ± 0.000	0.994 ± 0.000
LightGBM	0.994 ± 0.000	0.995 ± 0.000	0.995 ± 0.000
CatBoost	0.995 ± 0.000	0.995 ± 0.000	0.995 ± 0.000
EBM	0.985 ± 0.000	0.986 ± 0.001	0.986 ± 0.001
FastAIMLP	0.995 ± 0.000	0.995 ± 0.000	0.995 ± 0.000
TorchMLP	0.993 ± 0.000	0.995 ± 0.000	0.995 ± 0.000
RealMLP	0.995 ± 0.000	0.995 ± 0.000	0.995 ± 0.000
TabM	0.995 ± 0.000	0.995 ± 0.000	0.995 ± 0.000
MNCA	0.991 ± 0.000	0.994 ± 0.000	0.995 ± 0.000
TabPFNv2	-	-	-
TabDPT	0.994 ± 0.000	-	-
TabICL	0.995 ± 0.000	-	-
Linear	0.964 ± 0.001	0.964 ± 0.001	0.965 ± 0.001
KNN	0.625 ± 0.002	0.676 ± 0.002	0.680 ± 0.002
AutoGluon	-	-	**0.996 ± 0.000**

diabetes (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.825 ± 0.023	0.830 ± 0.025	0.830 ± 0.024
ExtraTrees	0.826 ± 0.022	0.837 ± 0.021	0.837 ± 0.021
XGBoost	0.824 ± 0.024	0.830 ± 0.024	0.832 ± 0.024
LightGBM	0.829 ± 0.025	0.838 ± 0.026	0.837 ± 0.025
CatBoost	0.833 ± 0.025	0.834 ± 0.025	0.835 ± 0.024
EBM	0.840 ± 0.025	0.839 ± 0.023	0.840 ± 0.023
FastAIMLP	0.826 ± 0.024	0.832 ± 0.023	0.835 ± 0.023
TorchMLP	0.821 ± 0.024	0.825 ± 0.026	0.827 ± 0.025
RealMLP	0.833 ± 0.023	0.832 ± 0.022	0.836 ± 0.024
TabM	0.832 ± 0.023	0.832 ± 0.025	0.835 ± 0.024
MNCA	0.840 ± 0.024	0.836 ± 0.025	0.813 ± 0.022
TabPFNv2	**0.844 ± 0.023**	**0.842 ± 0.024**	0.839 ± 0.024
TabDPT	0.840 ± 0.023	-	-
TabICL	0.837 ± 0.023	-	-
Linear	0.832 ± 0.024	0.831 ± 0.023	0.831 ± 0.023
KNN	0.740 ± 0.030	0.809 ± 0.026	0.806 ± 0.027
AutoGluon	-	-	0.835 ± 0.023

diamonds (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	549.9 ± 8.3	549.9 ± 8.3	547.2 ± 9.3
ExtraTrees	536.6 ± 8.7	536.3 ± 9.1	534.6 ± 8.9
XGBoost	539.0 ± 10.0	530.1 ± 10.0	528.2 ± 10.9
LightGBM	532.1 ± 9.1	524.9 ± 9.7	519.0 ± 9.4
CatBoost	520.7 ± 12.0	520.7 ± 12.0	520.8 ± 11.8
EBM	618.2 ± 14.5	613.7 ± 11.9	612.4 ± 13.9
FastAIMLP	563.1 ± 15.0	559.4 ± 9.4	550.0 ± 8.9
TorchMLP	627.3 ± 25.8	550.5 ± 16.6	542.6 ± 15.2
RealMLP	529.6 ± 8.1	521.5 ± 7.6	513.7 ± 7.3
TabM	522.5 ± 9.2	522.6 ± 8.9	520.4 ± 8.8
MNCA	531.1 ± 11.0	523.2 ± 9.4	**512.9 ± 7.9**
TabPFNv2	-	-	-
TabDPT	535.4 ± 13.1	-	-
TabICL	-	-	-
Linear	1652.1 ± 177.2	1142.9 ± 28.3	1144.1 ± 29.7
KNN	1461.6 ± 19.2	1368.3 ± 19.4	1362.5 ± 19.5
AutoGluon	-	-	**510.5 ± 9.5**

hazelnut-spread-contaminant-detection (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.958 ± 0.006	0.959 ± 0.006	0.960 ± 0.006
ExtraTrees	0.955 ± 0.006	0.964 ± 0.005	0.964 ± 0.005
XGBoost	0.973 ± 0.005	0.975 ± 0.004	0.975 ± 0.004
LightGBM	0.973 ± 0.005	0.978 ± 0.004	0.978 ± 0.004
CatBoost	0.974 ± 0.004	0.975 ± 0.004	0.974 ± 0.004
EBM	0.971 ± 0.005	0.975 ± 0.004	0.976 ± 0.004
FastAIMLP	0.983 ± 0.003	0.986 ± 0.003	0.986 ± 0.003
TorchMLP	0.983 ± 0.003	0.987 ± 0.003	0.987 ± 0.003
RealMLP	0.984 ± 0.003	0.986 ± 0.003	0.986 ± 0.003
TabM	0.967 ± 0.005	0.984 ± 0.003	0.984 ± 0.003
MNCA	0.985 ± 0.003	0.988 ± 0.003	0.988 ± 0.003
TabPFNv2	0.988 ± 0.003	0.989 ± 0.003	0.989 ± 0.003
TabDPT	**0.992 ± 0.002**	-	-
TabICL	**0.992 ± 0.002**	-	-
Linear	0.948 ± 0.006	0.951 ± 0.006	0.952 ± 0.006
KNN	0.908 ± 0.009	0.922 ± 0.010	0.931 ± 0.008
AutoGluon	-	-	0.987 ± 0.003

healthcare_insurance_expenses (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	4889.0 ± 289.0	4641.0 ± 304.0	4630.0 ± 303.0
ExtraTrees	4845.0 ± 275.0	4607.0 ± 324.0	4610.0 ± 323.0
XGBoost	4672.0 ± 306.0	4523.0 ± 320.0	4520.0 ± 320.0
LightGBM	4610.0 ± 314.0	4525.0 ± 329.0	4512.0 ± 325.0
CatBoost	4535.0 ± 329.0	4518.0 ± 322.0	4519.0 ± 321.0
EBM	4549.0 ± 319.0	**4500.0 ± 333.0**	**4499.0 ± 326.0**
FastAIMLP	4720.0 ± 313.0	4634.0 ± 318.0	4624.0 ± 311.0
TorchMLP	4661.0 ± 343.0	4526.0 ± 329.0	4534.0 ± 333.0
RealMLP	4579.0 ± 313.0	4571.0 ± 324.0	4535.0 ± 323.0
TabM	4519.0 ± 321.0	4528.0 ± 338.0	**4507.0 ± 327.0**
MNCA	4606.0 ± 331.0	4609.0 ± 337.0	4596.0 ± 338.0
TabPFNv2	4695.0 ± 303.0	4650.0 ± 337.0	4568.0 ± 319.0
TabDPT	**4508.0 ± 295.0**	-	-
TabICL	-	-	-
Linear	6083.0 ± 276.0	6085.0 ± 276.0	6085.0 ± 276.0
KNN	12371.0 ± 504.0	11514.0 ± 472.0	11540.0 ± 478.0
AutoGluon	-	-	**4490.0 ± 332.0**

heloc (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.791 ± 0.005	0.792 ± 0.006	0.793 ± 0.005
ExtraTrees	0.790 ± 0.005	0.793 ± 0.005	0.793 ± 0.005
XGBoost	0.794 ± 0.005	0.797 ± 0.005	0.797 ± 0.005
LightGBM	0.794 ± 0.005	0.799 ± 0.005	0.799 ± 0.005
CatBoost	0.798 ± 0.004	0.798 ± 0.005	0.798 ± 0.004
EBM	0.799 ± 0.005	0.799 ± 0.005	0.799 ± 0.005
FastAIMLP	0.791 ± 0.004	0.794 ± 0.005	0.795 ± 0.005
TorchMLP	0.791 ± 0.004	0.795 ± 0.004	0.796 ± 0.004
RealMLP	0.798 ± 0.004	0.798 ± 0.004	0.800 ± 0.004
TabM	0.797 ± 0.004	0.799 ± 0.004	0.799 ± 0.004
MNCA	0.799 ± 0.004	**0.800 ± 0.004**	0.799 ± 0.005
TabPFNv2	**0.801 ± 0.003**	**0.801 ± 0.004**	**0.801 ± 0.003**
TabDPT	0.794 ± 0.004	-	-
TabICL	0.800 ± 0.004	-	-
Linear	0.786 ± 0.005	0.786 ± 0.005	0.786 ± 0.005
KNN	0.691 ± 0.007	0.747 ± 0.003	0.747 ± 0.003
AutoGluon	-	-	0.798 ± 0.005

hiva_agnostic (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.263 ± 0.025	**0.174 ± 0.001**	**0.174 ± 0.001**
ExtraTrees	0.268 ± 0.027	**0.174 ± 0.000**	**0.174 ± 0.000**
XGBoost	0.182 ± 0.002	0.179 ± 0.002	0.179 ± 0.002
LightGBM	0.175 ± 0.001	0.175 ± 0.001	0.175 ± 0.001
CatBoost	0.176 ± 0.001	0.177 ± 0.002	0.177 ± 0.002
EBM	**0.174 ± 0.001**	0.176 ± 0.001	0.175 ± 0.001
FastAIMLP	0.213 ± 0.010	0.183 ± 0.008	0.183 ± 0.004
TorchMLP	0.183 ± 0.005	0.176 ± 0.001	0.178 ± 0.003
RealMLP	0.196 ± 0.007	0.176 ± 0.002	0.179 ± 0.002
TabM	0.176 ± 0.001	0.175 ± 0.001	0.175 ± 0.001
MNCA	0.221 ± 0.013	0.176 ± 0.002	0.179 ± 0.003
TabPFNv2	-	-	-
TabDPT	0.181 ± 0.004	-	-
TabICL	-	-	-
Linear	0.448 ± 0.024	0.449 ± 0.024	0.450 ± 0.024
KNN	0.263 ± 0.025	0.264 ± 0.026	0.264 ± 0.026
AutoGluon	-	-	0.193 ± 0.027

houses (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	0.231 ± 0.002	0.231 ± 0.002	0.230 ± 0.002
ExtraTrees	0.243 ± 0.002	0.238 ± 0.002	0.238 ± 0.002
XGBoost	0.215 ± 0.003	0.215 ± 0.002	0.215 ± 0.002
LightGBM	0.217 ± 0.002	0.212 ± 0.002	0.211 ± 0.002
CatBoost	0.211 ± 0.002	0.211 ± 0.002	0.211 ± 0.002
EBM	0.231 ± 0.003	0.229 ± 0.003	0.228 ± 0.003
FastAIMLP	0.244 ± 0.001	0.236 ± 0.002	0.235 ± 0.001
TorchMLP	0.233 ± 0.003	0.228 ± 0.003	0.226 ± 0.002
RealMLP	0.223 ± 0.002	0.211 ± 0.003	0.203 ± 0.003
TabM	0.212 ± 0.002	0.208 ± 0.002	0.206 ± 0.002
MNCA	0.204 ± 0.003	0.204 ± 0.002	**0.200 ± 0.003**
TabPFNv2	-	-	-
TabDPT	0.209 ± 0.003	-	-
TabICL	-	-	-
Linear	0.325 ± 0.003	0.325 ± 0.003	0.324 ± 0.003
KNN	0.516 ± 0.004	0.480 ± 0.004	0.478 ± 0.004
AutoGluon	-	-	0.204 ± 0.002

in_vehicle_coupon_recommendation (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.812 ± 0.007	0.817 ± 0.008	0.822 ± 0.008
ExtraTrees	0.798 ± 0.007	0.804 ± 0.008	0.811 ± 0.008
XGBoost	0.832 ± 0.004	0.842 ± 0.005	0.843 ± 0.005
LightGBM	0.836 ± 0.005	0.844 ± 0.005	0.845 ± 0.005
CatBoost	0.840 ± 0.006	0.843 ± 0.005	0.844 ± 0.006
EBM	0.802 ± 0.006	0.807 ± 0.006	0.807 ± 0.006
FastAIMLP	0.810 ± 0.007	0.823 ± 0.007	0.826 ± 0.006
TorchMLP	0.825 ± 0.004	0.833 ± 0.008	0.841 ± 0.006
RealMLP	0.837 ± 0.006	0.839 ± 0.006	0.849 ± 0.006
TabM	0.848 ± 0.004	0.851 ± 0.006	**0.852 ± 0.006**
MNCA	0.812 ± 0.005	0.843 ± 0.006	0.849 ± 0.006
TabPFNv2	0.789 ± 0.008	0.806 ± 0.008	0.837 ± 0.007
TabDPT	0.798 ± 0.005	-	-
TabICL	0.846 ± 0.006	-	-
Linear	0.735 ± 0.007	0.735 ± 0.007	0.735 ± 0.007
KNN	0.500 ± 0.000	0.502 ± 0.005	0.502 ± 0.005
AutoGluon	-	-	0.847 ± 0.006

jm1 (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.752 ± 0.008	0.752 ± 0.008	0.761 ± 0.007
ExtraTrees	0.756 ± 0.007	0.758 ± 0.008	0.765 ± 0.006
XGBoost	0.748 ± 0.007	0.749 ± 0.007	0.752 ± 0.006
LightGBM	0.748 ± 0.006	0.751 ± 0.006	0.753 ± 0.006
CatBoost	0.744 ± 0.005	0.751 ± 0.005	0.749 ± 0.005
EBM	0.735 ± 0.006	0.734 ± 0.007	0.736 ± 0.007
FastAIMLP	0.728 ± 0.007	0.728 ± 0.007	0.733 ± 0.006
TorchMLP	0.728 ± 0.005	0.734 ± 0.005	0.736 ± 0.005
RealMLP	0.731 ± 0.007	0.735 ± 0.007	0.749 ± 0.007
TabM	0.735 ± 0.009	0.738 ± 0.005	0.746 ± 0.006
MNCA	0.762 ± 0.003	0.762 ± 0.006	0.769 ± 0.005
TabPFNv2	0.732 ± 0.008	0.755 ± 0.007	0.773 ± 0.006
TabDPT	0.771 ± 0.005	-	-
TabICL	**0.776 ± 0.005**	-	-
Linear	0.724 ± 0.006	0.724 ± 0.006	0.724 ± 0.007
KNN	0.648 ± 0.008	0.730 ± 0.009	0.730 ± 0.009
AutoGluon	-	-	0.760 ± 0.007

kddcup09_appetency (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.772 ± 0.016	0.822 ± 0.011	0.821 ± 0.010
ExtraTrees	0.771 ± 0.012	0.819 ± 0.012	0.821 ± 0.009
XGBoost	0.830 ± 0.012	0.833 ± 0.009	0.837 ± 0.010
LightGBM	0.798 ± 0.009	0.821 ± 0.009	0.829 ± 0.010
CatBoost	**0.846 ± 0.008**	**0.845 ± 0.008**	**0.845 ± 0.008**
EBM	0.826 ± 0.009	0.831 ± 0.010	0.833 ± 0.010
FastAIMLP	0.749 ± 0.023	0.795 ± 0.013	0.804 ± 0.014
TorchMLP	0.819 ± 0.012	0.826 ± 0.012	0.831 ± 0.013
RealMLP	0.820 ± 0.011	0.822 ± 0.011	0.832 ± 0.011
TabM	0.777 ± 0.021	0.816 ± 0.008	0.818 ± 0.009
MNCA	0.772 ± 0.016	0.813 ± 0.013	0.822 ± 0.012
TabPFNv2	-	-	-
TabDPT	0.742 ± 0.009	-	-
TabICL	0.811 ± 0.014	-	-
Linear	0.797 ± 0.013	0.797 ± 0.013	0.796 ± 0.014
KNN	0.511 ± 0.007	0.553 ± 0.010	0.551 ± 0.011
AutoGluon	-	-	**0.846 ± 0.009**

maternal_health_risk (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.479 ± 0.071	0.470 ± 0.053	0.448 ± 0.054
ExtraTrees	0.478 ± 0.069	0.453 ± 0.055	0.443 ± 0.055
XGBoost	0.470 ± 0.051	0.459 ± 0.051	0.462 ± 0.050
LightGBM	0.488 ± 0.052	0.462 ± 0.049	0.461 ± 0.045
CatBoost	0.478 ± 0.055	0.463 ± 0.053	0.459 ± 0.049
EBM	0.569 ± 0.038	0.562 ± 0.042	0.557 ± 0.040
FastAIMLP	0.650 ± 0.044	0.617 ± 0.040	0.611 ± 0.039
TorchMLP	0.606 ± 0.054	0.566 ± 0.053	0.554 ± 0.046
RealMLP	0.558 ± 0.056	0.463 ± 0.058	0.436 ± 0.049
TabM	0.516 ± 0.051	0.483 ± 0.056	0.469 ± 0.049
MNCA	0.452 ± 0.039	0.444 ± 0.050	0.428 ± 0.047
TabPFNv2	0.451 ± 0.047	0.439 ± 0.057	0.437 ± 0.057
TabDPT	**0.405 ± 0.062**	-	-
TabICL	**0.410 ± 0.058**	-	-
Linear	0.796 ± 0.037	0.795 ± 0.042	0.784 ± 0.039
KNN	1.372 ± 0.265	0.869 ± 0.149	0.490 ± 0.062
AutoGluon	-	-	0.462 ± 0.061

miami_housing (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	9676 ± 592	9332 ± 503	9302 ± 500
ExtraTrees	9482 ± 400	9195 ± 395	9166 ± 409
XGBoost	8650 ± 447	8062 ± 361	8042 ± 357
LightGBM	8563 ± 463	8124 ± 410	7961 ± 354
CatBoost	7985 ± 315	7836 ± 342	**7847 ± 329**
EBM	10420 ± 365	9882 ± 466	9823 ± 438
FastAIMLP	9034 ± 512	8855 ± 535	8664 ± 483
TorchMLP	9265 ± 455	8631 ± 511	8528 ± 466
RealMLP	8605 ± 402	8337 ± 457	8018 ± 399
TabM	8283 ± 495	8105 ± 430	8008 ± 432
MNCA	8800 ± 388	8226 ± 349	8021 ± 417
TabPFNv2	8579 ± 447	7829 ± 457	**7711 ± 442**
TabDPT	8213 ± 497	-	-
TabICL	-	-	-
Linear	19126 ± 458	17554 ± 363	17554 ± 363
KNN	14211 ± 438	13300 ± 442	13148 ± 399
AutoGluon	-	-	7873 ± 429

online_shoppers_intention (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.926 ± 0.004	0.931 ± 0.004	0.932 ± 0.003
ExtraTrees	0.918 ± 0.005	0.931 ± 0.004	0.931 ± 0.004
XGBoost	0.935 ± 0.004	0.934 ± 0.003	0.936 ± 0.003
LightGBM	0.934 ± 0.003	0.935 ± 0.004	0.935 ± 0.003
CatBoost	0.934 ± 0.003	0.933 ± 0.004	0.934 ± 0.004
EBM	0.931 ± 0.003	0.931 ± 0.003	0.931 ± 0.003
FastAIMLP	0.926 ± 0.004	0.931 ± 0.005	0.932 ± 0.004
TorchMLP	0.930 ± 0.004	0.935 ± 0.004	0.936 ± 0.003
RealMLP	0.929 ± 0.004	0.933 ± 0.003	0.934 ± 0.004
TabM	0.935 ± 0.003	0.936 ± 0.003	0.936 ± 0.003
MNCA	0.934 ± 0.003	0.935 ± 0.003	0.936 ± 0.003
TabPFNv2	0.934 ± 0.004	0.937 ± 0.003	**0.937 ± 0.003**
TabDPT	0.926 ± 0.005	-	-
TabICL	**0.937 ± 0.003**	-	-
Linear	0.913 ± 0.007	0.913 ± 0.007	0.917 ± 0.007
KNN	0.759 ± 0.009	0.828 ± 0.006	0.832 ± 0.003
AutoGluon	-	-	0.936 ± 0.003

physiochemical_protein (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	3.565 ± 0.021	3.463 ± 0.021	3.471 ± 0.021
ExtraTrees	3.548 ± 0.022	3.466 ± 0.022	3.474 ± 0.021
XGBoost	3.513 ± 0.024	3.390 ± 0.024	3.390 ± 0.024
LightGBM	3.477 ± 0.026	3.381 ± 0.027	3.384 ± 0.027
CatBoost	3.522 ± 0.026	3.395 ± 0.029	3.383 ± 0.027
EBM	4.241 ± 0.020	4.234 ± 0.019	4.226 ± 0.020
FastAIMLP	4.014 ± 0.027	3.675 ± 0.031	3.683 ± 0.034
TorchMLP	3.388 ± 0.021	3.289 ± 0.028	3.228 ± 0.018
RealMLP	3.466 ± 0.042	3.284 ± 0.029	3.125 ± 0.028
TabM	3.445 ± 0.030	3.309 ± 0.028	3.287 ± 0.030
MNCA	3.183 ± 0.041	3.136 ± 0.039	3.048 ± 0.036
TabPFNv2	-	-	-
TabDPT	**2.912 ± 0.033**	-	-
TabICL	-	-	-
Linear	5.194 ± 0.023	5.188 ± 0.022	5.142 ± 0.023
KNN	5.955 ± 0.028	5.482 ± 0.027	5.453 ± 0.026
AutoGluon	-	-	3.107 ± 0.026

polish_companies_bankruptcy (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.927 ± 0.007	0.935 ± 0.008	0.938 ± 0.007
ExtraTrees	0.874 ± 0.013	0.891 ± 0.013	0.893 ± 0.011
XGBoost	0.958 ± 0.007	0.957 ± 0.007	0.957 ± 0.008
LightGBM	0.954 ± 0.007	0.955 ± 0.008	0.957 ± 0.007
CatBoost	0.961 ± 0.008	0.961 ± 0.008	0.960 ± 0.008
EBM	0.962 ± 0.009	0.962 ± 0.010	0.964 ± 0.009
FastAIMLP	0.841 ± 0.026	0.852 ± 0.034	0.862 ± 0.022
TorchMLP	0.903 ± 0.006	0.955 ± 0.006	0.957 ± 0.005
RealMLP	0.962 ± 0.004	0.957 ± 0.006	0.963 ± 0.006
TabM	0.951 ± 0.009	0.969 ± 0.005	0.970 ± 0.004
MNCA	0.962 ± 0.007	0.968 ± 0.010	0.968 ± 0.007
TabPFNv2	0.959 ± 0.006	0.979 ± 0.003	**0.981 ± 0.002**
TabDPT	0.958 ± 0.009	-	-
TabICL	0.974 ± 0.002	-	-
Linear	0.867 ± 0.016	0.887 ± 0.013	0.891 ± 0.013
KNN	0.698 ± 0.015	0.784 ± 0.015	0.786 ± 0.017
AutoGluon	-	-	0.969 ± 0.005

qsar-biodeg (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.930 ± 0.013	0.929 ± 0.013	0.929 ± 0.013
ExtraTrees	0.932 ± 0.013	0.932 ± 0.013	0.934 ± 0.013
XGBoost	0.926 ± 0.013	0.931 ± 0.012	0.931 ± 0.012
LightGBM	0.927 ± 0.012	0.933 ± 0.012	0.933 ± 0.012
CatBoost	0.930 ± 0.012	0.931 ± 0.012	0.932 ± 0.011
EBM	0.931 ± 0.011	0.931 ± 0.011	0.933 ± 0.011
FastAIMLP	0.932 ± 0.013	0.932 ± 0.014	0.934 ± 0.013
TorchMLP	0.924 ± 0.014	0.923 ± 0.014	0.927 ± 0.014
RealMLP	0.927 ± 0.013	0.926 ± 0.015	0.934 ± 0.012
TabM	0.931 ± 0.011	0.934 ± 0.013	0.936 ± 0.012
MNCA	0.928 ± 0.011	0.928 ± 0.013	0.931 ± 0.012
TabPFNv2	0.936 ± 0.011	0.932 ± 0.013	0.936 ± 0.012
TabDPT	0.934 ± 0.012	-	-
TabICL	**0.938 ± 0.012**	-	-
Linear	0.910 ± 0.016	0.917 ± 0.014	0.918 ± 0.014
KNN	0.862 ± 0.018	0.894 ± 0.022	0.898 ± 0.019
AutoGluon	-	-	0.934 ± 0.013

seismic-bumps (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.747 ± 0.021	0.767 ± 0.022	0.765 ± 0.026
ExtraTrees	0.734 ± 0.024	0.767 ± 0.029	0.771 ± 0.025
XGBoost	0.759 ± 0.022	0.768 ± 0.024	0.771 ± 0.025
LightGBM	0.752 ± 0.027	0.770 ± 0.027	0.771 ± 0.026
CatBoost	**0.776 ± 0.027**	0.772 ± 0.026	0.767 ± 0.028
EBM	0.770 ± 0.026	0.763 ± 0.026	0.767 ± 0.025
FastAIMLP	0.728 ± 0.034	0.759 ± 0.033	0.761 ± 0.028
TorchMLP	0.763 ± 0.026	0.758 ± 0.026	0.762 ± 0.025
RealMLP	0.760 ± 0.030	0.761 ± 0.027	0.766 ± 0.027
TabM	0.768 ± 0.027	0.769 ± 0.024	0.771 ± 0.024
MNCA	0.768 ± 0.024	0.765 ± 0.026	0.738 ± 0.019
TabPFNv2	0.772 ± 0.025	0.766 ± 0.023	0.769 ± 0.024
TabDPT	**0.774 ± 0.022**	-	-
TabICL	**0.783 ± 0.024**	-	-
Linear	0.759 ± 0.024	0.757 ± 0.023	0.761 ± 0.025
KNN	0.594 ± 0.019	0.702 ± 0.026	0.701 ± 0.031
AutoGluon	-	-	0.758 ± 0.032

splice (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.317 ± 0.005	0.180 ± 0.015	0.178 ± 0.014
ExtraTrees	0.393 ± 0.007	0.175 ± 0.015	0.173 ± 0.013
XGBoost	0.119 ± 0.020	0.109 ± 0.018	0.109 ± 0.018
LightGBM	0.111 ± 0.016	0.103 ± 0.016	0.102 ± 0.015
CatBoost	0.110 ± 0.017	0.115 ± 0.020	0.111 ± 0.018
EBM	0.118 ± 0.013	0.119 ± 0.012	0.117 ± 0.012
FastAIMLP	0.118 ± 0.013	**0.105 ± 0.013**	**0.103 ± 0.014**
TorchMLP	0.157 ± 0.022	0.127 ± 0.017	0.116 ± 0.014
RealMLP	0.126 ± 0.014	0.110 ± 0.014	0.106 ± 0.013
TabM	0.111 ± 0.017	0.113 ± 0.016	0.110 ± 0.016
MNCA	0.145 ± 0.013	0.121 ± 0.012	0.122 ± 0.013
TabPFNv2	0.107 ± 0.015	0.113 ± 0.019	**0.099 ± 0.015**
TabDPT	0.267 ± 0.010	-	-
TabICL	0.148 ± 0.020	-	-
Linear	0.167 ± 0.019	0.167 ± 0.019	0.167 ± 0.019
KNN	0.317 ± 0.005	0.317 ± 0.005	0.317 ± 0.005
AutoGluon	-	-	**0.100 ± 0.017**

students_dropout_and_academic_success (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.584 ± 0.011	0.576 ± 0.014	0.574 ± 0.013
ExtraTrees	0.591 ± 0.012	0.570 ± 0.011	0.566 ± 0.013
XGBoost	0.554 ± 0.016	0.545 ± 0.015	0.546 ± 0.014
LightGBM	0.555 ± 0.015	0.543 ± 0.016	0.542 ± 0.015
CatBoost	0.552 ± 0.016	0.543 ± 0.017	0.541 ± 0.016
EBM	0.565 ± 0.014	0.561 ± 0.013	0.560 ± 0.014
FastAIMLP	0.565 ± 0.021	0.549 ± 0.015	0.540 ± 0.015
TorchMLP	0.581 ± 0.018	0.559 ± 0.016	0.552 ± 0.014
RealMLP	0.556 ± 0.015	0.553 ± 0.013	0.542 ± 0.014
TabM	0.544 ± 0.012	0.542 ± 0.013	0.538 ± 0.014
MNCA	0.555 ± 0.014	0.554 ± 0.014	0.547 ± 0.013
TabPFNv2	0.534 ± 0.012	**0.529 ± 0.015**	**0.527 ± 0.015**
TabDPT	0.561 ± 0.018	-	-
TabICL	0.550 ± 0.014	-	-
Linear	0.571 ± 0.017	0.571 ± 0.017	0.571 ± 0.016
KNN	1.957 ± 0.115	0.721 ± 0.005	0.715 ± 0.006
AutoGluon	-	-	0.536 ± 0.015

superconductivity (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	9.63 ± 0.19	9.62 ± 0.19	9.53 ± 0.19
ExtraTrees	9.43 ± 0.18	9.43 ± 0.18	9.39 ± 0.18
XGBoost	9.45 ± 0.18	9.35 ± 0.21	9.34 ± 0.20
LightGBM	9.41 ± 0.21	9.28 ± 0.22	9.26 ± 0.20
CatBoost	9.36 ± 0.19	9.34 ± 0.21	9.34 ± 0.20
EBM	10.53 ± 0.23	10.43 ± 0.22	10.21 ± 0.18
FastAIMLP	11.51 ± 0.10	10.59 ± 0.10	10.55 ± 0.12
TorchMLP	9.95 ± 0.21	9.66 ± 0.16	9.56 ± 0.16
RealMLP	9.57 ± 0.29	9.44 ± 0.23	9.22 ± 0.21
TabM	9.58 ± 0.21	9.36 ± 0.20	9.36 ± 0.20
MNCA	9.60 ± 0.15	9.49 ± 0.18	9.30 ± 0.20
TabPFNv2	-	-	-
TabDPT	**9.08 ± 0.20**	-	-
TabICL	-	-	-
Linear	17.43 ± 0.10	17.42 ± 0.09	17.27 ± 0.10
KNN	12.38 ± 0.14	10.85 ± 0.27	10.63 ± 0.21
AutoGluon	-	-	9.22 ± 0.16

taiwanese_bankruptcy_prediction (AUC ↑)

	Default	Tuned	Tuned + Ens.
RF	0.933 ± 0.011	0.932 ± 0.011	0.932 ± 0.012
ExtraTrees	0.937 ± 0.011	0.937 ± 0.008	0.938 ± 0.008
XGBoost	0.941 ± 0.008	**0.943 ± 0.008**	**0.944 ± 0.008**
LightGBM	0.938 ± 0.008	**0.943 ± 0.008**	**0.944 ± 0.007**
CatBoost	**0.944 ± 0.006**	0.944 ± 0.007	0.943 ± 0.006
EBM	0.942 ± 0.005	0.940 ± 0.005	0.941 ± 0.004
FastAIMLP	0.914 ± 0.022	0.923 ± 0.009	0.927 ± 0.010
TorchMLP	0.925 ± 0.009	0.940 ± 0.007	**0.943 ± 0.007**
RealMLP	0.941 ± 0.006	0.941 ± 0.007	**0.945 ± 0.006**
TabM	0.941 ± 0.003	0.941 ± 0.004	0.943 ± 0.004
MNCA	0.939 ± 0.004	0.938 ± 0.007	0.936 ± 0.010
TabPFNv2	0.942 ± 0.005	0.943 ± 0.007	**0.945 ± 0.008**
TabDPT	0.937 ± 0.008	-	-
TabICL	0.944 ± 0.006	-	-
Linear	0.936 ± 0.005	0.936 ± 0.005	0.936 ± 0.005
KNN	0.594 ± 0.018	0.678 ± 0.029	0.681 ± 0.027
AutoGluon	-	-	**0.946 ± 0.006**

website_phishing (logloss ↓)

	Default	Tuned	Tuned + Ens.
RF	0.312 ± 0.037	0.262 ± 0.019	0.257 ± 0.019
ExtraTrees	0.312 ± 0.036	0.258 ± 0.020	0.250 ± 0.019
XGBoost	0.260 ± 0.027	0.251 ± 0.022	0.251 ± 0.023
LightGBM	0.255 ± 0.021	0.249 ± 0.021	0.247 ± 0.021
CatBoost	0.252 ± 0.022	0.239 ± 0.022	0.239 ± 0.021
EBM	0.357 ± 0.021	0.357 ± 0.020	0.357 ± 0.020
FastAIMLP	0.334 ± 0.023	0.245 ± 0.022	0.241 ± 0.021
TorchMLP	0.289 ± 0.035	0.237 ± 0.020	0.230 ± 0.019
RealMLP	0.256 ± 0.026	0.236 ± 0.026	0.232 ± 0.021
TabM	0.245 ± 0.025	0.238 ± 0.029	0.236 ± 0.024
MNCA	0.261 ± 0.022	0.240 ± 0.021	0.238 ± 0.021
TabPFNv2	0.229 ± 0.025	**0.223 ± 0.027**	**0.222 ± 0.025**
TabDPT	0.228 ± 0.027	-	-
TabICL	0.228 ± 0.026	-	-
Linear	0.360 ± 0.021	0.361 ± 0.022	0.358 ± 0.023
KNN	0.312 ± 0.037	0.312 ± 0.037	0.312 ± 0.037
AutoGluon	-	-	0.233 ± 0.021

wine_quality (rmse ↓)

	Default	Tuned	Tuned + Ens.
RF	0.620 ± 0.021	0.616 ± 0.021	0.611 ± 0.021
ExtraTrees	0.613 ± 0.021	0.606 ± 0.025	0.603 ± 0.021
XGBoost	0.621 ± 0.021	0.610 ± 0.019	0.610 ± 0.020
LightGBM	0.628 ± 0.019	0.607 ± 0.019	0.608 ± 0.019
CatBoost	0.621 ± 0.019	0.605 ± 0.020	0.605 ± 0.020
EBM	0.679 ± 0.015	0.678 ± 0.016	0.675 ± 0.016
FastAIMLP	0.669 ± 0.019	0.668 ± 0.018	0.657 ± 0.019
TorchMLP	0.691 ± 0.019	0.653 ± 0.018	0.650 ± 0.015
RealMLP	0.625 ± 0.018	0.618 ± 0.019	0.603 ± 0.020
TabM	0.633 ± 0.020	0.620 ± 0.019	0.612 ± 0.020
MNCA	0.611 ± 0.020	0.607 ± 0.018	0.601 ± 0.020
TabPFNv2	0.692 ± 0.012	0.639 ± 0.017	0.610 ± 0.020
TabDPT	**0.590 ± 0.018**	-	-
TabICL	-	-	-
Linear	0.731 ± 0.017	0.731 ± 0.017	0.730 ± 0.016
KNN	0.809 ± 0.018	0.689 ± 0.020	0.687 ± 0.020
AutoGluon	-	-	0.599 ± 0.020