When Do Boosted Trees Outperform Neural Nets on Tabular Data?

Duncan McElfresh[*1], **Sujay Khandagale**[2], **Jonathan Valverde**[3], **Vishak Prasad C**[4], **Ganesh Ramakrishnan**[4], **Micah Goldblum**[5], **Colin White**[1]

[1] Abacus.AI, [2] Pinterest, [3] University of Maryland, [4] IIT Bombay, [5] New York University

Abstract

Tabular data is one of the most commonly used types of data in machine learning. Despite recent advances in neural nets for tabular data, there is still an active discussion on whether or not neural nets generally outperform gradient-boosted decision trees (GBDTs) on tabular data. This is an important question, because neural nets are often not practical in low-resource settings. In this work, we take a completely different approach by focusing on what *properties* of a dataset make neural nets or simpler approaches better-suited to perform well. To accomplish this, we conduct the largest tabular data analysis to date, by comparing 22 approaches across 171 datasets, while testing 965 metafeatures. We find that, for a surprisingly high number of datasets, either the performance difference between GBDTs and neural nets is negligible, or light hyperparameter tuning on a GBDT is more important than selecting the best neural net. Furthermore, we identify metafeatures of datasets that are predictive of certain algorithms, and algorithm families, performing well. Based on these insights, we present a guide for practitioners to decide whether or not they need to run a neural net to reach top performance on their dataset.

1 Introduction

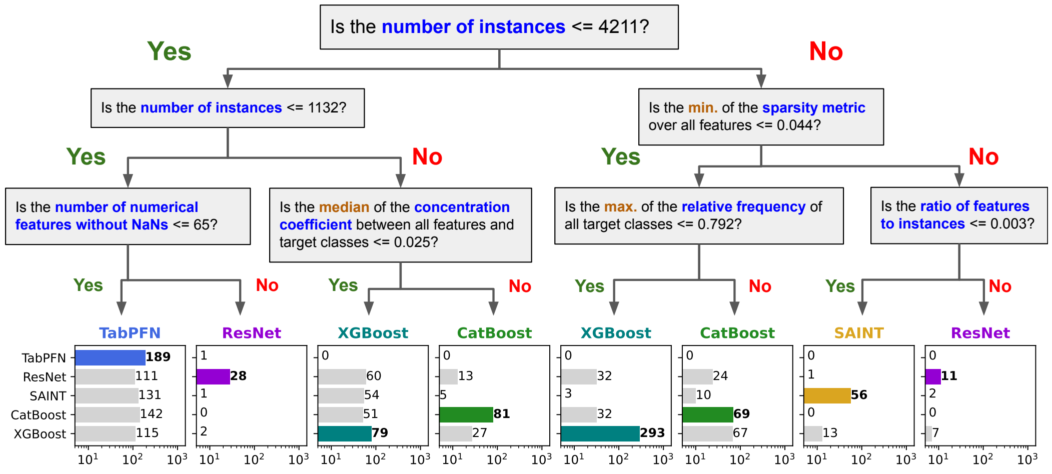
Tabular datasets—data organized into rows and columns consisting of continuous, categorical, and ordinal features—are the oldest and among the most ubiquitous dataset types in machine learning in practice (Shwartz-Ziv & Armon, 2022; Borisov et al., 2021), due to their numerous applications across medicine (Johnson et al., 2016; Ulmer et al., 2020), finance (Arun et al., 2016; Clements et al., 2020), online advertising (Richardson et al., 2007; McMahan et al., 2013; Guo et al., 2017), and many other areas (Chandola et al., 2009; Buczak & Guven, 2015; Urban & Gates, 2021).

Despite several advances in designing neural nets for tabular data (Arik & Pfister, 2021; Popov et al., 2020), there is still an active debate over whether or not deep learning methods generally outperform gradient-boosted decision trees (GBDTs) on tabular data, with multiple works arguing either for (Kadra et al., 2021; Arik & Pfister, 2021; Popov et al., 2020; Rubachev et al., 2022) or against (Shwartz-Ziv & Armon, 2022; Borisov et al., 2021; Gorishniy et al., 2021) neural networks. This is an important question, because neural nets are often not practical in low-resource settings. Many prior studies on tabular data use fewer than 50 datasets or do not properly tune baselines (Tunguz, 2022; Lipton & Steinhardt, 2019), putting the generalizability of these findings into question. Furthermore, the bottom line of many prior works is to determine which method performs the best (in terms of the average rank across datasets), without searching for more fine-grained insights.

In this work, we take a completely different approach by focusing on what *properties* of a dataset indicate that neural nets or GBDTs better-suited to perform well. We take a data-driven approach to answer this question, conducting the largest tabular data analysis to date, by comparing 22 algorithms with up to 30 hyperparameter settings, across 171 datasets, while testing over 965 metafeatures. We use 10 train/validation folds for each dataset, in order to further reduce the uncertainty of our results.

Our first insight is that there is no clear winner: for about 25% of the datasets, a neural network model performs better than any non-neural model, and for 27%, a GBDT model performs better than

[*]Correspondence to: `{duncan, colin}@abacus.ai`.



Is the number of instances <= 4211?
Yes
No
Is the number of instances <= 1132?
Is the min. of the sparsity metric over all features <= 0.044?
Yes
No
Yes
No
Is the number of numerical features without NaNs <= 65?
Is the median of the concentration coefficient between all features and target classes <= 0.025?
Is the max. of the relative frequency of all target classes <= 0.792?
Is the ratio of features to instances <= 0.003?
Yes
No
Yes
No
Yes
No
Yes
No
TabPFN
ResNet
XGBoost
CatBoost
XGBoost
CatBoost
SAINT
ResNet
TabPFN
ResNet
SAINT
CatBoost
XGBoost
189
111
131
142
115
1
28
1
0
2
0
60
54
51
79
0
13
5
81
27
0
32
3
32
293
0
24
10
69
67
0
1
56
0
13
0
11
2
0
7
10^1
10^2
10^3

Figure 1: Decision tree for picking the best algorithm, based on our experiments across 171 datasets. The decision tree chooses among the five best-performing algorithms from our experiments: ResNet, SAINT, TabPFN, CatBoost, and XGBoost. When faced with a new dataset, we calculate the numerical properties of the dataset at each node. IQR denotes interquartile range, and CanCor (Lindner & Studer, 1999) denotes canonical correlation.

any non-GBDT model. We also show that for a surprisingly high fraction of datasets, either a simple baseline method performs on par with all other methods, or light hyperparameter tuning on a GBDT increases performance more than choosing the best technique. These results show that for many tabular datasets, training the latest neural network is not necessary. Next, we run analyses to discover what properties of datasets explain which methods, or families of methods, do or do not succeed. For example, we find that the most predictive property to determine whether a GBDT outperforms neural nets is having a *low homogeneity of covariance* among the features, and the most predictive property to determine whether a baseline performs well is having homogeneity in the skewness of features. Based on these analyses, we present a guide for practitioners to follow when faced with a new dataset, and we also identify failure modes and promising directions for future work. Our codebase and all raw results are available at `https://anonymous.4open.science/r/tabzilla`.

Related Work. GBDTs are a powerful technique to model tabular data, which work by building an ensemble of decision trees, incrementally updated using gradient descent. Due to their strong performance, many high-performing instantiations have been proposed, such as XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018).

There are three types of tabular data (Borisov et al., 2021). Data transformation methods (Yoon et al., 2020; Hancock & Khoshgoftaar, 2020) seek to encode the data into a format that is better-suited for neural nets. Architecture-based methods design specialized architectures for tabular data (Popov et al., 2020; Guo et al., 2017; Chen et al., 2022), a large sub-class of which are transformer-based architectures (Arik & Pfister, 2021; Gorishniy et al., 2021; Huang et al., 2020; Somepalli et al., 2021). Regularization-based methods seek to use tailored regularization methods to reduce the extreme flexibility of deep learning (Shavitt & Segal, 2018; Kadra et al., 2021). Several recent works have compared GBDTs to neural nets on tabular data, finding that neural nets (Kadra et al., 2021; Gorishniy et al., 2021) or GBDTs (Shwartz-Ziv & Armon, 2022; Borisov et al., 2021; Grinsztajn et al., 2022) perform best. However, none considered more than 40 datasets, compared to our 171 datasets.

2 Analysis of Algorithms for Tabular Data

We present results for 22 algorithms, including popular recent techniques and common baselines. The methods include three GBDTs: CatBoost (Prokhorenkova et al., 2018), LightGBM (Ke et al., 2017), and XGBoost (Chen & Guestrin, 2016); 14 neural networks: DANet (Chen et al., 2022), DeepFM (Guo et al., 2017), FTTransformer (Gorishniy et al., 2021), two MLPs (Gorishniy et al., 2021), NAM (Agarwal et al., 2021), NODE (Popov et al., 2020), ResNet (Gorishniy et al., 2021), SAINT (Somepalli et al., 2021), STG (Yamada et al., 2020), TabNet (Arik & Pfister, 2021), TabPFN (Hollmann et al., 2022), TabTransformer (Huang et al., 2020), and VIME (Yoon et al., 2020); and five baselines: Decision Tree (Quinlan, 1986), KNN (Cover & Hart, 1967), Logistic Regression (Cox, 1958), Random Forest (Liaw et al., 2002), and SVM (Cortes & Vapnik, 1995). These algorithms were chosen because of their popularity and high performance.

We run the algorithms on 171 classification datasets from OpenML (Vanschoren et al., 2014). Our aim is to include most classification datasets from popular recent papers that study tabular data (Kadra et al., 2021; Borisov et al., 2021; Shwartz-Ziv & Armon, 2022; Gorishniy et al., 2021), including all tabular datasets from the OpenML-CC18 suite Bischl et al. (2017) and the AutoML benchmark Gijsbers et al. (2019). To the best of our knowledge our 171 datasets and 22 algorithms are the largest number of *either* datasets *or* algorithms considered by recent tabular dataset literature, and the largest number available to run with popular algorithms in a single open-source repository.

Table 1: Ranking of 21 algorithms over all 171 datasets, according to test accuracy. Rank columns show min, max, and mean ranks over all datasets, and mean acc. indicates the mean normalized accuracy.

Algorithm	Rank			Mean
	min	*max*	*mean*	Acc.
CatBoost	1	18	5.21	0.87
XGBoost	1	19	5.61	0.87
ResNet	1	20	6.85	0.78
LightGBM	1	20	6.96	0.82
SAINT	1	19	7.15	0.78
NODE	1	20	7.48	0.75
RandomForest	1	19	8.09	0.77
FTTransformer	1	17	8.10	0.75
SVM	1	19	8.31	0.74
DANet	1	20	8.65	0.76
MLP-rtdl	1	19	9.57	0.67
DeepFM	1	21	10.69	0.63
TabNet	1	21	11.04	0.63
MLP	1	20	11.37	0.61
DecisionTree	1	21	11.41	0.60
TabTransformer	1	21	11.42	0.57
STG	1	21	11.47	0.60
LinearModel	1	20	12.19	0.51
KNN	1	21	12.80	0.53
VIME	1	21	14.45	0.41
NAM	1	21	15.68	0.34

For each dataset, we use the ten train/test folds provided by OpenML (Vanschoren et al., 2014). Appendix Table 2 shows summary statistics for all 1710 training splits used in our experiments. For each algorithm, and for each dataset-split, we ran the algorithm for up to 10 hours. During this time we trained and evaluated the algorithm with at most 30 hyperparameter sets (one default set and 29 random sets). In line with prior work, our main metric of interest is *accuracy*. For nearly all of our experiments, for each algorithm and dataset fold pair, we report the test performance of the hyperparameter setting that had the maximum performance on the validation set.

To compute metafeatures, we extracted general, statistical, information theoretic, landmarking, and model-based metafeatures using the PyMFE Alcobaça et al. (2020). See Appendix A for more details of our experimental design.

2.1 Relative Algorithm Performance

In this section, we investigate the research question, "How do individual techniques, and families of techniques, perform across a large set of datasets?" Specifically, we will look at which algorithms perform well on average across all datasets, which families of algorithms perform well (especially, GBDTs vs. neural nets), and whether the differences between algorithm families is significant.

We start by comparing the average rank of 21 algorithms across 171 datasets (we exclude TabPFN, since it cannot run on large datasets). Table 1 shows the min (best), max (worst) and average rank of each algorithm, according to accuracy, across 171 datasets; ties are assigned the min (best) rank. We see that, surprisingly, *every algorithm ranks first on at least one dataset, and nearly last on at least one other dataset*. The fact that the best out of 21 algorithms, CatBoost, only achieved an average rank of 5.21, shows that there is not a single approach that dominates across most datsets. In Appendix Table 3, we compute the same table for the 63 smallest datasets, so that we can include TabPFN (Hollmann et al., 2022) in our rankings (which can only run on at most 2000 datapoints). We find that TabPFN achieves the best average performance of all algorithms.

GBDTs vs. neural nets. Although Table 1 tells us which *individual* methods perform best on average, now we ask the age-old question, "are GBDTs better than neural networks for tabular data?" We split the 22 algorithms into three *families*: `GBDTs` (CatBoost, XGBoost, LightGBM), `neural nets` (the 14 listed above), and `baselines` (DecisionTree, KNN, LinearModel, RandomForest, SVM). We analyze each dataset-split separately.

To compare algorithm performance across dataset-splits, we use min-max scaling to normalize each algorithm's tuned accuracy to be on $[0, 1]$, and we say that an algorithm "wins" for a dataset-split if it achieves a normalized accuracy of at least 0.99. For each of the 1705 dataset-splits, we determine which algorithm families (tree, neural net, baseline) have a winning algorithm. Figure 2 shows the number of dataset splits where each subset of algorithm classes win, according to accuracy. Surprisingly, neural networks are the *sole* winner for only 25.3% of datasets, showing that for most datasets, a GBDT or baseline suffices!

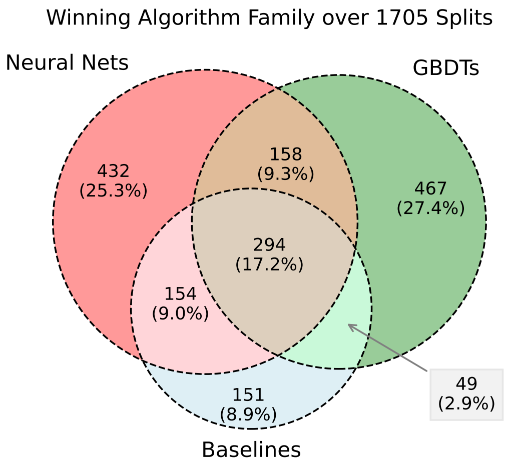


Figure 2: Venn Diagram of the number of "wins" for each algorithm class, over all 1705 dataset splits. An algorithm wins if its normalized accuracy is at least 0.99 (out of 1) on the test set.

Algorithm selection vs. tuning. Next, we investigate whether it is more important to select the best possible algorithm family, or to simply run light hyperparameter tuning on CatBoost. For each dataset, we calculate two quantities: *(1)* the *impact of algorithm selection*, measured by the performance difference between the best tuned tree method and the best tuned neural net; and *(2)* the *impact of hyperparameter tuning*, measured by the performance difference between CatBoost with default hyperparameters vs. CatBoost tuned via 30 iterations of random search on the validation set. Surprisingly, we find that for nearly 50% of datasets, light hyperparameter tuning results in a greater increase in performance than algorithm selection. Once again, this suggests that for a large fraction of datasets, it is not necessary to use a neural net. In the next section, we explore *why* a dataset might be more amenable to a neural net or a GBDT.

2.2 Metafeature analysis

In this section, we investigate the research question, "What properties of a dataset are associated with certain techniques, or families of techniques, outperforming others, and how can practitioners make use of this information?" Specifically, we take a meta-learning approach to ask *(1)* can we predict which algorithm will perform best on a given dataset, and *(2)* can we predict whether GBDTs, neural nets, and/or baselines perform well on a given dataset?

We assess the *difference* in performance between neural nets and GBDTs, calculated as the difference in normalized accuracy between the best neural net and the best GBDT method, which we refer to as Δ_{acc}. First, we observe that many dataset properties are correlated with Δ_{acc} over all 1705 dataset splits. Appendix Table 4 shows the dataset properties with the largest absolute correlation with Δ_{acc}, and Appendix Figure 6 plots Δ_{acc} with three of these most-correlated properties. In order to show that these metafeatures are *predictive*, we train and evaluate a meta-learning model using a leave-one-out approach: one dataset is held out for testing, while the remaining 170 datasets are used for training, averaged across all 171 test sets. Appendix Table 5 shows the performance accuracy reaches 0.67 for decision trees, and 0.74 for an XGBoost model.

A guide for practitioners. Now we present a concrete guide for practitioners to follow when presented with a new tabular dataset. Specifically, we present decision trees created using all 171 datasets that we study. First, we train a decision tree to predict which of the top five algorithms works best: SAINT, ResNet, CatBoost, XGBoost, or TabPFN. The result is a simple heuristic to show what approach to run, by just computing three simple tests on the new dataset. See Figure 1. We also compute similar heuristics to predict whether a baseline method will perform just as well as the best GBDT and neural net. Appendix Figure 7b shows when baseline methods perform well. Finally, Appendix Figure 7a shows a decision tree trained to identify datasets where the only winning algorithms are neural nets.

2.2.1 FAILURE MODES

In this section, we investigate the research question, "what properties of a dataset are associated with a certain method performing better or worse?" In other words, we ask when *individual* algorithms perform well.

We focus on the top-performing algorithms from previous sections: CatBoost, XGBoost, ResNet, LightGBM, SAINT, and NODE, and we focus on their *relative* performance among all 22 algorithms. First, we calculate the correlation between each dataset metafeature and normalized accuracy of these algorithms; Appendix Table 6 shows the metafeature with the largest absolute correlation with normalized accuracy.

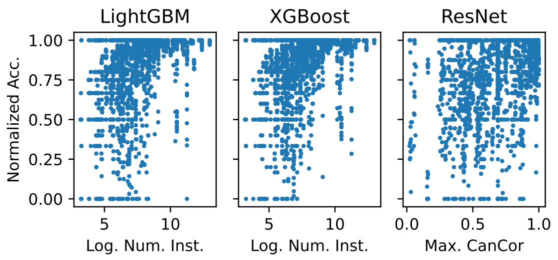


Figure 3: Normalized accuracy of three algorithms plotted with its metafeature most-correlated with performance.

Figure 3 plots the normalized accuracy for the three algorithms in Table 6 with the greatest absolute correlation: XGBoost, CatBoost, and ResNet. Our analysis shows, for each algorithm, the one dataset property that is most associated with the algorithm having strong or poor performance. For example, somewhat surprisingly, LightGBM and XGBoost perform relatively poorly for smaller datasets. Our hope is that this analysis is a starting point for the tabular data research community to develop improvements, ensembles, and even better methods for tabular data.

3 CONCLUSIONS AND FUTURE WORK

In this work, we studied what *properties* of a dataset make neural nets or trees better-suited to perform well. We conducted the largest tabular data analysis to date, by comparing 22 approaches across 171 datasets. We found that, for a surprisingly high number of datasets, either the performance difference between GBDTs and neural nets is negligible, or light hyperparameter tuning is more important than selecting the best method. By analyzing 965 metafeatures, we presented a guide for practitioners to decide which method or family of methods to use on a new dataset, and we looked at what metafeatures are correlated with performance for a given algorithm or family of algorithms.

Future Directions. Our findings raise several questions for future inquiry. First, there are many datasets where the best neural net approach outperforms the best GBDT, and many other cases where the reverse is true. This suggests that *both* neural nets and GBDTs are worthwhile approaches to tabular datasets, and future work should refine these methods to improve performance. In particular, our failure mode analysis shows the types of datasets on which a particular method struggles to perform well. Second, baseline methods perform well for a surprisingly large number of datasets. Since these baselines are often simpler and less costly to train and test than modern neural nets and GBDTs, it is worth asking how much additional performance can be achieved through tuning strong baseline algorithms. Since no algorithm or family of algorithms *clearly dominates* for tabular data, it is likely that a diverse ensemble including multiple algorithm families will outperform any single algorithm—or family. Since we find that several dataset metafeatures are related to algorithm performance, an ensembling approach might use these metafeatures to build an effective ensemble for a particular dataset.

References

Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Edesio Alcobaça, Felipe Siqueira, Adriano Rivolli, Luís P. F. Garcia, Jefferson T. Oliva, and André C. P. L. F. de Carvalho. Mfe: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111):1–5, 2020. URL `http://jmlr.org/papers/v21/19-348.html`.

Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Kumar Arun, Garg Ishan, and Kaur Sanmeet. Loan approval prediction based on machine learning approach. *IOSR J. Comput. Eng*, 18(3):18–21, 2016.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *arXiv preprint arXiv:1708.03731*, 2017.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

Jintai Chen, Kuanlun Liao, Yao Wan, Danny Z Chen, and Jian Wu. Danets: Deep abstract networks for tabular data classification and regression. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Jillian M Clements, Di Xu, Nooshin Yousefi, and Dmitry Efimov. Sequential deep learning for credit risk monitoring with tabular financial data. *arXiv preprint arXiv:2012.15330*, 2020.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 1995.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 1967.

David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1958.

Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, and Joaquin Vanschoren. An open source automl benchmark. *arXiv preprint arXiv:1907.00909*, 2019.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI*, 2017.

John T Hancock and Taghi M Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):1–41, 2020.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3): 18–22, 2002.

Guido Lindner and Rudi Studer. Ast: Support for algorithm selection with a cbr approach. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '99, pp. 418–423, Berlin, Heidelberg, 1999. Springer-Verlag. ISBN 3540664904.

Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship: Some ml papers suffer from flaws that could mislead the public and stymie future research. *Queue*, 2019.

H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1222–1230, 2013.

Donald Michie, D. J. Spiegelhalter, C. C. Taylor, and John Campbell (eds.). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, USA, 1995. ISBN 013106360X.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1986.

Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pp. 521–530, 2007.

Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.

Ira Shavitt and Eran Segal. Regularization learning networks: deep learning for tabular datasets. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Bojan Tunguz. Trouble with hopular, 2022. URL `https://medium.com/@tunguz/trouble-with-hopular-6649f22fa2d3`.

Dennis Ulmer, Lotta Meijerink, and Giovanni Cinà. Trust issues: Uncertainty estimation does not enable reliable ood detection on medical tabular data. In *Machine Learning for Health*, pp. 341–354. PMLR, 2020.

Christopher J Urban and Kathleen M Gates. Deep learning: A primer for psychologists. *Psychological Methods*, 2021.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 2014.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

Table 2: Summary statistics for all 1710 training splits used in our experiments; each training split accounts for 80% of the full dataset. Left columns show the number of instances, number of features, and number of target classes. Right columns show the number of feature types.

	# Inst.	# Feats.	# Classes	# Feature Types		
				Num.	Bin.	Cat.
Mean	14930.6	222.4	5.4	205.2	25.2	17.1
Min.	24	2	2	0	0	0
25%	455.2	9.0	2.0	4.0	0.0	0.0
50%	1600.0	21.0	2.0	10.0	0.0	0.0
75%	8239.0	60.0	6.0	51.0	3.0	8.0
Max.	431507	7200	100	7200	1555	1555

A Additional Results

A.1 Details of experimental setup

Metafeatures. We extracted metafeatures using the Python library PyMFE Alcobaça et al. (2020). Within this library, metafeatures are divided into groups. We used metafeatures from as many groups as possible, which includes: general (such as number of classes and number of numeric features), statistical (such as the mean and range of each feature), information theoretic (such as the Shannon entropy of the target), landmarking (running a baseline such as Naive Bayes or 1-Nearest Neighbor on a subsample of the dataset), and model-based (summary statistics for some model fit on the data, such as number of leaf nodes in a decision tree model). Several of the metafeature extraction processes result in distributions, and we aggregate these using several summary statistics such as the mean, standard deviation, and interquartile range. Since some of these features have long-tailed distributions, we also include the log of each strictly-positive metafeature in our analysis.

Experimental design. For each dataset, we use the ten train/test folds provided by OpenML (Vanschoren et al., 2014), which allows our results on the test folds to be compared with other works that used the same OpenML datasets. Since we also needed validation splits in order to run hyperparameter tuning, we used a procedure to split the training folds into train and validation splits. Given a dataset, for each fold i, we keep the test set the same as in the original OpenML split. We divide the remaining 9 folds into a validation set (fold $i + 1$ modulo 10), and the remaining folds are added to a training set. In this way, all samples are included in training, testing, and validation splits at the same rate. Henceforth we refer to a complete OpenML dataset as a *dataset* and we refer to a single (training, validation, test) split as a *dataset-split*. Of the 171 datasets there are 1710 dataset-splits; five of these produced errors in all algorithms and were removed from our analysis, leaving 1705 dataset-split remaining. Table 2 shows summary statistics for all 1710 training splits used in our experiments. Roughly half of our datasets have a binary classification target (and as many as 100 target classes), and roughly half of all training sets have fewer than 2000 instances—though many datasets have tens of thousands of instances.

For each algorithm, and for each dataset-split, we ran the algorithm for up to 10 hours. During this time we trained and evaluated the algorithm with at most 30 hyperparameter sets (one default set and 29 random sets). Each parameterized algorithm is given at most two hours to complete a single train/evaluation cycle. This controls for runtimes of the algorithms – the faster algorithms are able to search more hyperparameter settings. For nearly all of our experiments, for each algorithm and dataset fold pair, we use the performance tuned on the validation set; that is, we report the test performance of the hyperparameter setting that had the maximum performance on the validation set.

A.2 Dataset statistics

Table 2 shows summary statistics for all 1710 training splits used in our experiments. Roughly half of our datasets have a binary classification target (and as many as 100 target classes), and roughly half of all training sets have fewer than 2000 instances—though many datasets have tens of thousands of instances.

A.3 Experiments on small datasets

TabPFN (Hollmann et al., 2022) is a recently released meta-learning approach for tabular data, which can perform supervised classification in less than a second. However, its current implementation is limited to datasets with size at most 1000 datapoints, at most 100 features, and at most 10 target classes. Of the 171 datasets used in our main analysis, 63 fit these criteria. Table 3 shows the average ranking of all 22 algorithms on these 63 datasets. We find that TabPFN achieves the best average performance of all algorithms, even beating CatBoost. However, with an average rank of 4.92, it still does not dominate all other approaches across most datasets. It is notable that XGBoost performed considerably worse on these smaller datasets, compared to its performance across all 171 datasets.

Table 3: Ranking of 22 algorithms over all 63 datasets where TabPFN can be run, according to test accuracy. Lower ranks indicate higher accuracy. Rank columns show min, max, and mean ranks over all datasets, and mean acc. indicates the mean normalized accuracy across all datasets.

Alg.	Rank			Mean Acc
	min	*max*	*mean*	(normalized)
TabPFN	1	20	4.92	0.86
CatBoost	1	19	5.68	0.85
ResNet	1	21	7.48	0.76
SAINT	1	20	7.81	0.76
FTTransformer	1	18	8.32	0.76
RandomForest	1	19	8.37	0.76
NODE	1	21	8.48	0.75
XGBoost	1	20	8.86	0.75
DeepFM	1	22	9.43	0.71
MLP-rtdl	1	19	10.10	0.67
SVM	1	20	10.24	0.69
LinearModel	1	21	10.65	0.64
LightGBM	1	21	10.89	0.68
DANet	1	21	11.40	0.68
MLP	1	20	12.19	0.60
TabTransformer	1	22	12.34	0.60
STG	1	22	12.38	0.60
DecisionTree	1	21	12.41	0.62
KNN	1	22	14.00	0.49
NAM	1	22	15.26	0.45
TabNet	3	22	15.83	0.42
VIME	1	22	15.84	0.37

A.4 Critical difference diagrams

In this section we compare performance of all algorithms across all datasets, to determine whether rankings are statistically significant. First we use a Friedman test to determine whether performance differences between each algorithm are significant. If these differences are significant according to the Friedman test ($p < 0.05$), then we use a Wilcoxon signed-rank test to determine which pairs of algorithms have significant performance differences ($p<0.05$). A Holm-Bonferroni correction is used to account for multiple comparisons.

Figure 4 shows a critical difference plot indicating significant differences between each algorithm. In this figure, the average rank of each algorithm is shown on the horizontal axis; if differences between algorithms are *not significant* ($p \geq 0.05$), then algorithms are shown connected by a horizontal bar.

Next, we compare the performance of each algorithm *type* (GBDT, neural networks, and baselines). We use the same methodology as in the previous plot. See Figure 5.

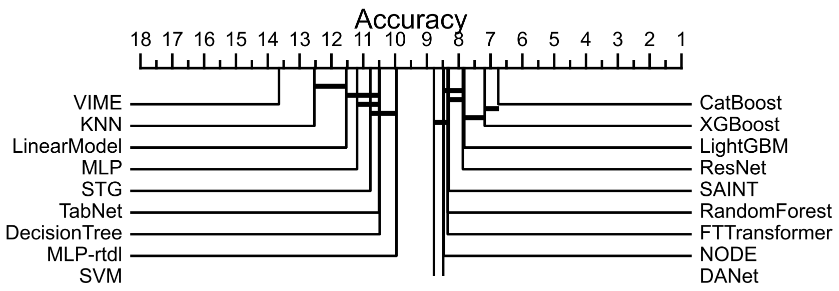


Figure 4: Critical Difference plot comparing all algorithms according to accuracy. Each algorithm's average rank is shown as a horizontal line on the axis. Algorithms which are *not significantly different* are connected by a horizontal black bar.

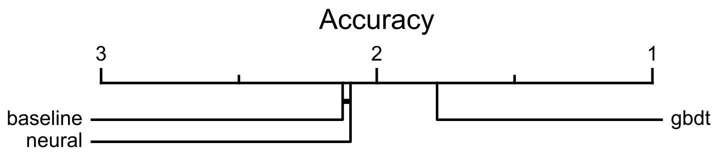


Figure 5: Critical Difference plot comparing three algorithm types, according to accuracy. Each algorithm's average rank is shown as a horizontal line on the axis. Algorithms which are *not significantly different* are connected by a horizontal black bar.

A.5 METAFEATURE ANALYSIS

Table 4 shows the dataset properties with the largest absolute correlation with Δ_{acc}, and Figure 6 plots Δ_{acc} with three of these most-correlated properties.

To evaluate the predictive power of dataset properties, we train several decision tree models using the train/test procedure above, with a binary outcome: 1 if $\Delta_{\text{acc}} > 0$ (the best neural net beats the best GBDT), and 0 otherwise. Table 5 shows the performance accuracy of decision trees trained on this task, with varying depth levels; we also include an XGBoost model for comparison.

We compute similar heuristics to predict whether a baseline method will perform just as well as the best GBDT and neural net. Figure 7b shows when baseline methods perform well. Finally, Figure 7a shows a decision tree trained to identify datasets where the only winning algorithms are neural nets.

We calculate the correlation between each dataset metafeature and normalized accuracy of these algorithms; Table 6 shows the metafeature with the largest absolute correlation with normalized accuracy.

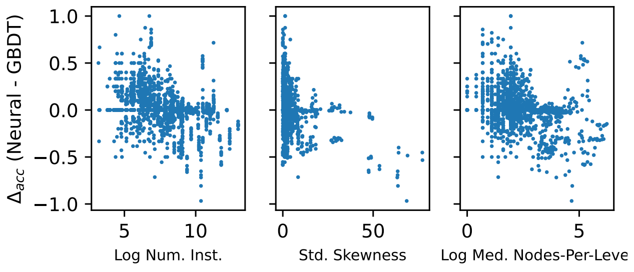


Figure 6: Difference in normalized accuracy between the best neural net and GBDT method (Δ_{acc}) by three dataset properties, for all 1705 dataset-splits. Each dataset property here is negatively correlated with Δ_{acc}, meaning that a higher value of the property corresponds to a higher (lower) accuracy for the best GBDT (neural net) algorithm. All dataset properties are plotted on a log scale.

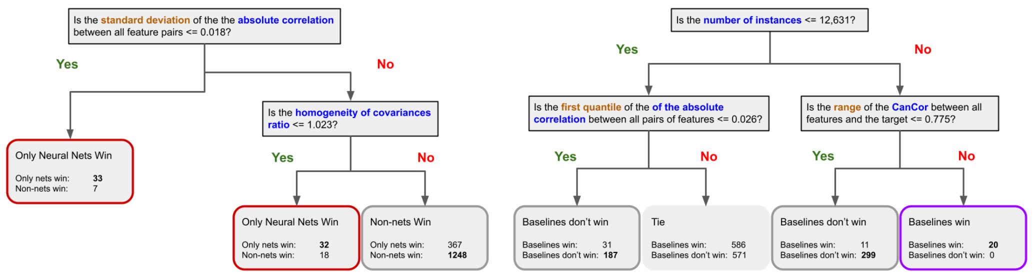
Table 4: Selected metafeatures with the largest absolute correlation with the difference in normalized accuracy between the best neural net and GBDT method (Δ_{acc}), over all 1705 dataset splits.

Description	**Corr. with Δ_{acc}**
Log of the number of instances in the dataset.	-0.332
Standard deviation of the skewness of all features	-0.329
Log of the median number of nodes per level of a decision tree trained on the dataset.	-0.321
Mean of the kurtosis of all features.	-0.307
Log of the max. test accuracy of a naive bayes classifier trained using 10-fold CV on the dataset.	0.296

Table 5: The test accuracy of tree models for predicting whether the best neural network will outperform the best GBDT model on a tabular dataset. Results are aggregated over 171 train/test splits with one dataset family held out for testing in each split. The number of dataset properties used by any model of each model type is listed in the leftmost column. Top rows include decision trees (DT$\leq n$) with maximum depth n; the bottom row is an XGBoost model for comparison.

Model	Test Accuracy (mean $\pm$ stddev)	Num. Metafeatures
DT$= 1$	0.54 ± 0.28	3
DT≤ 3	0.60 ± 0.29	21
DT≤ 5	0.59 ± 0.28	183
DT≤ 7	0.64 ± 0.29	382
DT≤ 9	0.66 ± 0.27	423
DT≤ 11	0.66 ± 0.30	529
DT≤ 13	0.65 ± 0.29	561
DT≤ 15	0.68 ± 0.30	594
DT$\leq \infty$	0.67 ± 0.30	685
XGBoost	0.74 ± 0.33	675

Discussion. Now, we give high-level takeaways from our raw metafeature analyses. First, our work corroborates some findings from prior work (Grinsztajn et al., 2022): our work (Figure 1) verifies that NNs are comparatively worse at handling uninformative features. Furthermore, ResNet also performs



(a) Do only neural net methods "win"?

(b) Does any baseline method "win"?

Figure 7: Decision trees illustrating two questions for model builders. Left: do *only* neural net algorithms "win"? Right: Does any baseline method "win"? In the left figure, homogoneity of covariance ratio is calculated as in Michie et al. (1995). In the right figure, CanCor indicaes the canonical covariance between any feature and the one-hot encoded target.

Table 6: Metafeatures most-correlated with the performance of each top-performing algorithm. The rightmost column shows the correlation between the metafeature and normalized accuracy over all 1705 dataset-splits.

Alg.	Metafeature Description	Corr.
CatBoost	Min. absolute value of the correlation between any two features	-0.29
XGBoost	Number of instances	0.36
LightGBM	Number of instances	0.41
ResNet	Max. canonical correlation between any numeric feature and the target	0.34
SAINT	Max. canonical correlation between any numeric feature and the target	0.33
NODE	The 1st quartile of the performance of a naive Bayes classifier over 10 folds of the dataset	0.32

particularly badly in the presence of features with outliers, which is new but related to prior work (Grinsztajn et al., 2022).

Overall, CatBoost is much more robust at handling datasets that are less "regular", compared to ResNet. Specifically, it is comparatively better on: datasets with a higher average skewness across features, datasets with less-balanced classes, and datasets where the range of feature values (after normalization) is high. We suspect that this is due to the nature of CatBoost (an ensemble) versus ResNet (a fixed MLP-like architecture).

Additionally, SAINT performs comparatively better than CatBoost and XGBoost on datasets with fewer features. We hypothesize that this is because SAINT performs attention across all pairs of features, and cannot generalize as well on datasets with many features. Finally, SAINT is correlated with higher performance compared to GBDTs, when the dataset has fewer features.

A.6 Ranking of default and tuned algorithms

We compute rankings of all algorithms according to accuracy, while including both tuned algorithms and algorithms parameterized with their default hyperparameters. See Table 7.

A.7 Training Time Analysis

In this section we analyze the relative training time required by each algorithm. Here we only consider algorithms with their *default* hyperparameters, so no tuning is used. Table 8 shows a ranking of all algorithms, according to the total training time per 1 000 training samples. We compute this table on all datasets where TabPFN can be run, becuase the results over all datasets is similar. These rankings are calculated by first taking the average training time per 1 000 samples over all 10 folds of all 171 datasets, and then ranking each algorithm for each dataset according to this average train time.

Next, we plot all algorithms according to both normalized accuracy and runtime. Figure 8 plots a point for each algorithm, where the x-axis is median runtime per 1 000 training samples, and the y-axis is median normalized accuracy—over all dataset splits.

Table 7: Ranking of 21 algorithms over all 171 datasets, according to test accuracy, including algorithms parameterized with default hyperparameters. Rank columns show min, max, and mean ranks over all datasets, and mean acc. indicates the mean normalized accuracy.

Algorithm	Rank			Mean
	min	*max*	*mean*	Acc.
CatBoost	1	30	8.05	0.91
CatBoost (default)	1	31	10.12	0.87
XGBoost	1	30	10.77	0.87
ResNet	1	34	11.51	0.84
XGBoost (default)	1	34	12.02	0.85
NODE	1	33	12.17	0.82
SAINT	1	34	12.27	0.81
FTTransformer	1	31	12.78	0.83
RandomForest	1	33	13.00	0.83
LightGBM	1	34	13.19	0.85
ResNet (default)	1	35	13.94	0.79
SVM	1	32	14.24	0.78
LightGBM (default)	1	34	14.38	0.80
SAINT (default)	1	35	14.48	0.76
NODE (default)	1	34	14.61	0.77
DANet	1	33	15.08	0.83
RandomForest (default)	1	35	16.22	0.76
MLP-rtdl	1	34	16.28	0.74
FTTransformer (default)	1	35	18.01	0.71
STG	1	34	18.73	0.70
DecisionTree	1	35	18.93	0.73
SVM (default)	1	35	19.63	0.64
MLP	1	34	19.66	0.69
LinearModel	1	35	19.95	0.64
MLP-rtdl (default)	1	35	20.11	0.63
DANet (default)	1	33	20.19	0.71
TabNet	1	35	20.56	0.68
DecisionTree (default)	1	35	21.88	0.63
KNN	1	35	22.88	0.59
TabNet (default)	1	35	23.52	0.60
MLP (default)	1	35	23.78	0.56
KNN (default)	1	35	24.61	0.54
VIME	3	34	25.16	0.53
STG (default)	1	35	26.17	0.44
VIME (default)	6	35	30.78	0.23

Table 8: Ranking of all 22 algorithms over all datasets where TabPFN can be run, according to average training time per 1 000 samples. Lower ranks indicate lower training times. Rank columns show min, max, and mean ranks over all datasets. Right columns show average training time per 1 000 samples over all 10 training folds, and the number of datasets considered for each algorithm.

Alg.		Rank		Mean Train Time
	min	*max*	*mean*	(s/1000 samples)
TabPFN	1	3	1.16	<0.01
KNN	1	3	2.02	<0.01
DecisionTree	1	4	2.86	0.02
LinearModel	3	5	4.27	0.06
SVM	4	8	5.21	0.22
LightGBM	5	12	6.46	0.82
RandomForest	5	8	6.65	0.71
XGBoost	5	8	7.46	1.42
CatBoost	8	19	9.65	10.41
DeepFM	9	11	9.79	6.08
MLP-rtdl	9	18	12.13	19.69
ResNet	10	17	12.86	23.30
VIME	9	17	13.33	17.83
STG	9	17	13.46	19.01
MLP	9	18	14.02	25.70
FTTransformer	11	19	14.63	28.98
TabNet	10	20	16.02	30.75
TabTransformer	12	20	16.32	33.25
DANet	15	22	18.73	53.09
NODE	14	22	18.95	62.21
SAINT	16	22	19.73	135.43
NAM	17	22	20.57	212.99

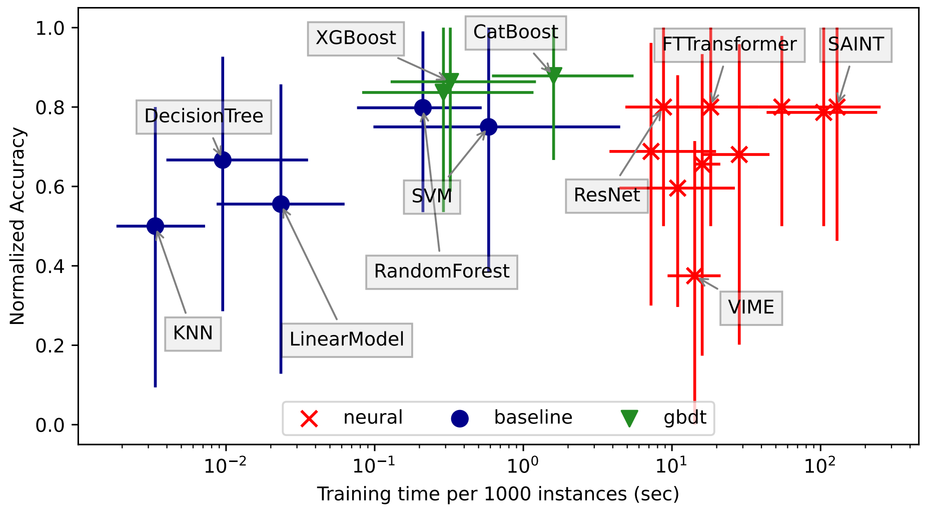


Figure 8: Median runtime vs. median normalized accuracy for each algorithm, over all 1040 dataset splits (for all 104 datasets). Error bars indicate the 20th and 80th percentile over all dataest splits.