Fine-Tuning the Retrieval Mechanism for Tabular Deep Learning

Felix den Breejen[1], **Sangmin Bae**[1], **Stephen Cha**[1], **Tae-Young Kim**[2], **Seoung Hyun Koh**[2], **Se-Young Yun**[1†]

{felixdenbreejen, bsmn0223, jooncha, yunseyoung}@kaist.ac.kr, {kim.taeyoung, seounghyun.koh}@kt.com

[1]KAIST AI [2]KT

Abstract

While interests in tabular deep learning has significantly grown, conventional tree-based models still outperform deep learning methods. To narrow this performance gap, we explore the innovative retrieval mechanism, a methodology that allows neural networks to refer to other data points while making predictions. Our experiments reveal that retrieval-based training, especially when fine-tuning the pretrained TabPFN model, notably surpasses existing methods. Moreover, the extensive pretraining plays a crucial role to enhance the performance of the model. These insights imply that blending the retrieval mechanism with pretraining and transfer learning schemes offers considerable potential for advancing the field of tabular deep learning.

1 Introduction

Tabular data has been widely utilized in the industry, leading to an increased interest in the research field of deep learning for tabular data. However, even with the significant advancements of deep learning in vision and language domains, previous studies [12, 17, 25] have confirmed that tree-based methods, such as Random Forests [3] and XGBoost [5], still prevail over deep learning techniques in tabular domains. In order to narrow the performance gap, recent research has introduced architectures that leverage the decision tree algorithm [22, 15] or the self-attention mechanism [1, 14, 11, 26]. Moreover, several works have focused on transfer learning to exploit the benefits of deep neural networks by pretraining on extensive datasets [28, 2, 13, 29].

Among various methodologies, a *retrieval* mechanism, which retrieves data points as the reference for prediction, has opened up new avenues in the field of tabular deep learning [16, 26, 23, 13, 10]. These neural networks make predictions by learning relationships between features and labels but also by referring to other samples, which resembles models like decision trees and nearest neighbors [7]. One specific application involves using input that contains a support set, $\mathcal{D} = (\boldsymbol{X}_{support}, \boldsymbol{y}_{support}, \boldsymbol{X}_{query})$, to predict the label of a query sample.

Specifically, TabPFN [13] builds on prior-data fitted networks [18] to pretrain on synthetically generated tabular prior datasets. Their goal is to approximate the posterior predictive distribution of a test sample, which is conditioned on the set of training samples (see Figure 1). While the trained models exhibit strong performance on unseen real-world tabular datasets in a single forward pass, this in-context learning [4] approach could have led to overlooking the further potential of retrieval-based training, potentially constraining the ability to transfer knowledge gained during pretraining.

In this paper, our comprehensive experiments, grounded on TabPFN and tabular benchmarks [12], reveal crucial insights emphasizing the significance of retrieval-based training. We demonstrate that

†Corresponding Author
Table Representation Learning Workshop at NeurIPS 2023.

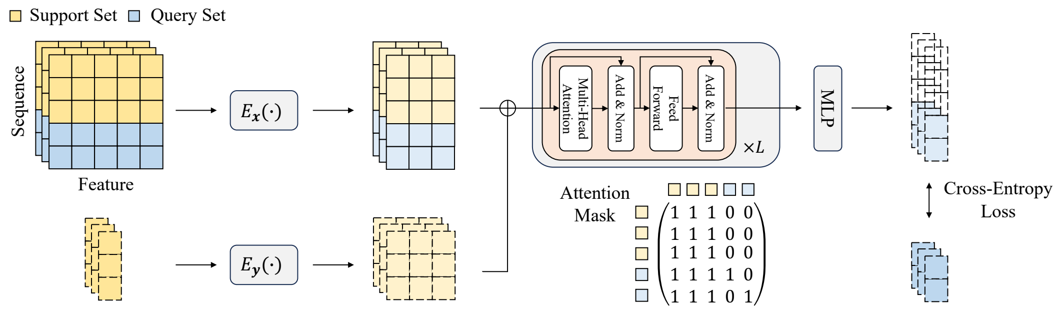


Figure 1: The overview of TabPFN training framework. Synthetically generated datasets are randomly divided into support and query sets during the pretraining phase, while train and test sets of tabular benchmarks are utilized during the evaluation.

the retrieval mechanism outperforms previous methods by further fine-tuning the pretrained TabPFN model. Moreover, a larger number of retrieved data points contributes to higher performance gains by offering enhanced references for prediction. We also find that pretraining on extensive datasets plays an important role when comparing to models trained from scratch. Consequently, we believe that integrating the retrieval mechanism with pretraining and transfer learning schemes will significantly influence the future landscape of tabular deep learning.

2 Related Work

2.1 Retrieval Mechanism in Tabular Data

Recent retrieval mechanism that implicitly or explicitly retrieves data points as the reference for prediction have shown significant improvements for tabular datasets. Non-Parameteric Transformers [16] take the entire dataset as input and learn relationships through self-attention between datapoints. Similarly, intersample attention from SAINT [26] associates the target row with other rows in the table. On the other hand, there are explicit retrieval-based models, like RIM [23] and TabR [10], that initially retrieve relevant instances using search engine techniques and a retrieval module, respectively. Subsequently, they utilize the features and labels of the selected rows to make the final prediction for the target row. Our work is primarily inspired by the TabPFN method [13], which let the token representations of test samples attend feature and label representations of training sets. While they only demonstrated the zero-shot evaluation performance, we fine-tuned pretrained models on actual tabular benchmarks, surpassing the previous deep learning approaches for tabular data.

2.2 Tabular Transfer Learning

Leveraging transfer learning methods from computer vision [6] and natural language processing [8] is a promising approach for tabular deep learning. Striving for generalization to unseen tables, various works in the tabular deep learning literature have attempted to pretrain on a large collection of different tables to exploit cross-table information [29, 27], harness biases from LLMs [19, 27] and adopt self/semi-supervised learning [20, 19]. XTab [29] processes entire tables using data featurizers and uses transformers for knowledge transfer. CT-BERT [27], similar to XTab, employs a pretrained BERT for natural language and translates tables into "[column name] is [value]" strings. STUNT [20] uses few-shot semi-supervised learning on self-generated tasks from unlabeled data, emphasizing column correlations. SPROUT [19] learns from unlabeled tables through LLMs, focusing on columns correlated with target labels, and applies in-context learning on generated prompts. The retrieval mechanism as used in this paper, can be seen as an alternative way of enabling transfer learning.

3 Methodology

TabPFN is a transformer-based architecture designed for zero-shot inference on small classification tasks. It is built on a basic transformer architecture that takes $(\boldsymbol{X}_{support}, \boldsymbol{y}_{support}, \boldsymbol{X}_{query})$ as input. Each observation in the dataset is tokenized during the embedding process. For observation $\boldsymbol{x}_i \in \mathbb{R}^{d_f}$

with corresponding class label $y_i \in \mathbb{Z}$ and d_f features, the embedding is calculated as follows:

$$\boldsymbol{z}_i^{support} = \boldsymbol{W}_x \boldsymbol{x}_i + \boldsymbol{w}_y y_i \tag{1}$$

$$\boldsymbol{z}_i^{query} = \boldsymbol{W}_x \boldsymbol{x}_i \tag{2}$$

with weight matrices $\boldsymbol{W}_x \in \mathbb{R}^{d \times d_f}$ and $\boldsymbol{w}_y \in \mathbb{R}^{d \times 1}$ for hidden dimension d. These embedded tokens $\boldsymbol{z}_i$ are then processed through a basic transformer where attention is applied between the observations, with the exception of attention between pairs of inference observations. In the final layer, tokens corresponding to the inference observations are passed through a classification head. Figure 1 illustrates the overview of the TabPFN architecture.

The features $\boldsymbol{x}_i$ are all variable-wise quantile transformed and scaled by d_f/d_f^i, where d_f^i is the effective number of features of observation i. The target y_i is kept as an unnormalized integer representing the class index. Categorical features are not one-hot encoded before the quantile transformation. Under the pretraining settings outlined in the TabPFN paper, the model accomodates up to $d_f = 100$ features per observation and a maximum of 10 classes.

For fine-tuning TabPFN, we use the checkpoint from the pretrained TabPFN model. Given a real-life test dataset $\mathcal{D} = (\boldsymbol{X}_{train}, \boldsymbol{y}_{train}, \boldsymbol{X}_{test}, \boldsymbol{y}_{test})$, we create a distinct randomized 80% split of $\boldsymbol{X}_{train}$ and $\boldsymbol{y}_{train}$ to make a new training dataset $\mathcal{D}_{train} = (\boldsymbol{X}_{train}^{support}, \boldsymbol{y}_{train}^{support}, \boldsymbol{X}_{train}^{query}, \boldsymbol{y}_{train}^{query})$ at every training step. We fine-tune on these splits and evaluate by predicting on $(\boldsymbol{X}_{train}, \boldsymbol{y}_{train}, \boldsymbol{X}_{test})$.

4 Experiments

We evaluate our fine-tuned TabPFN on tabular benchmarks [12] consisting of approximately 50 datasets categorized based on feature type (numerical or numerical and categorical) and tasks (classification or regression). We focus on the medium-size benchmarks containing about 10,000 observations and less than a hundred features. To ensure consistency to benchmarks, we follow the same procedures for data splitting, optimizer selection, learning rate scheduling, early stopping, and accuracy or R^2 score normalization.

The hyperparameters are as follows. We use a learning rate of 1.0×10^{-5} and no weight decay for fine-tuning. When training from scratch, we use a learning rate of 1.0×10^{-4} and weight decay of 1.0×10^{-5}. We use a support set size of 10,000 or 1,000. For a support set size of 10,000, the entire training dataset is processed through the transformer, while for 1,000, we randomly sample 1,000 samples from the training set and ensemble 10 times for each test observation. For all TabPFN variants, we run default settings without random search, as the search showed little benefit.

4.1 Main Results

We compare our finetuned TabPFN with three tree-based methods (XGBoost [5], Random Forest [3], and GradientBoostingTree [9]) and four neural network-based methods (MLP, Resnet, SAINT [26], and FT-Transformer [11]). Since TabPFN pretraining relies on synthetically generated data, we consider this a fair comparison in the quantity of real-world data used.

Figure 2 presents the overall comparison of classification accuracy. Here, we fine-tuned TabPFN with 10,000 number of retrieved samples. Utilizing default settings for fine-tuning TabPFN outperforms all neural network-based tabular data methods, even after extensive random hyperparameter searches for baselines. Furthermore, it also outperforms the tree-based methods under default settings, although tree-based methods tend to perform better on average after about 10 random searches. Additional dataset-specific results are available in Appendix B.

4.2 Ablation Study

Table 1 displays results for various TabPFN variations on mixed and numerical classification tasks, referring to the settings in Figure 2. There is a significant gap in normalized accuracy between fine-tuning on 10k samples and learning from scratch, with Zeroshot performance falling in between. We believe that these results imply the importance of fine-tuning phase after pretraining.

We also examine the size of the support set. TabPFN is pretrained on datasets smaller than 1,000 samples. The classification ICL and Fine-tuning results indicate that processing 10,000 samples is

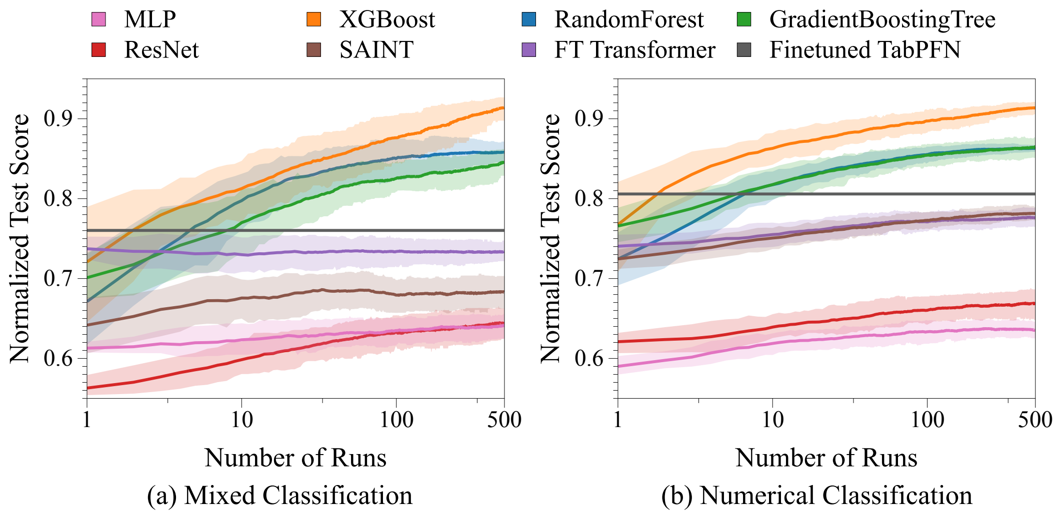


Figure 2: Comparison between fine-tuning TabPFN and baselines using the full 10,000 samples.

Table 1: Results on TabPFN variants for classification accuracy and regression R^2 score. The pretraining and fine-tuning are conducted on synthetic datasets and actual tabular datasets, respectively. ICL denotes the in-context learning, Scratch denotes training from scratch.

				classification		regression	
method	pretrain	fine-tune	# retrieval	mixed	numerical	mixed	numerical
Scratch	✗	✗	10k	0.5008	0.6057	**0.2199**	**0.2906**
ICL	✓	✗	1k	0.4451	0.5421	0.0000	0.0000
ICL	✓	✗	10k	0.5989	0.6609	0.0000	0.0000
Fine-tune	✓	✓	1k	0.6221	0.7073	0.0677	0.0877
Fine-tune	✓	✓	10k	**0.7598**	**0.8057**	0.0353	0.0913

preferable to ensembling 1,000-sample segments. For large datasets with over one million samples, the best approach appears to be utilizing as many observations as the GPU memory allows.

In regression tasks, fine-tuned TabPFN underperforms, as the expected R^2 should be around 0.8 (see Appendix A). This results comes as no surprise, as TabPFN is not pretrained for regression. The model's embedding expects integer class labels, but we provide quantile-transformed regression targets. This suggests the need for pretraining on regression or exploring alternative methods.

5 Conclusion

Our findings suggest that fine-tuning tabular data transformers pretrained using a retrieval mechanism is a compelling avenue for improving neural network based tabular data prediction. While research was predominantly centered on training neural networks from scratch, our results suggest that adopting the transfer learning paradigm holds significant potential. Pretrained transformers, unlike small tabular data architectures, can scale effectively in the size of the network. We imagine that large-scale companies could take this concept by developing a billion parameter transformer fine-tunable on downstream datasets. Such an approach, we believe, has the potential to decisively surpass the performance of all tree-based methods.

Our analysis identifies several areas for future research, with one pressing concern the scalability in the number of observations. Models like GPT-4 [21] are limited by a context length of 32k tokens and CodeLlama [24] by context length of 100k, but real-world tabular datasets can have millions of observations. Further research areas include refining the architecture to tailor it specifically to tabular data, exploring the limitations of synthetic data, and enabling regression within the model. In essence, our study opens doors to a range of exciting possibilities for enhancing tabular data prediction through transfer learning, paving the way for general tabular data solver.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)). This work was also supported by the "KT-KAIST Open R&D" project funded by KT Corporation.

References

[1] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687, 2021. Issue: 8.

[2] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. SCARF: Self-Supervised Contrastive Learning using Random Feature Corruption, March 2022. arXiv:2106.15147 [cs].

[3] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] 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*, pages 785–794, August 2016. arXiv:1603.02754 [cs].

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[7] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. Publisher: Institute of Mathematical Statistics.

[10] Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Unlocking the power of retrieval-augmented tabular deep learning. *arXiv preprint arXiv:2307.14338*, 2023.

[11] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[12] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? In *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv, July 2022. arXiv:2207.08815 [cs, stat].

[13] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *International Conference on Learning Representations (ICLR)*. arXiv, October 2022. arXiv:2207.01848 [cs, stat].

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

[15] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 384–394, 2019.

[16] Jannik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Thomas Rainforth, and Yarin Gal. Self-Attention Between Datapoints: Going Beyond Individual Input-Output Pairs in Deep Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 28742–28756. Curran Associates, Inc., 2021.

[17] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When Do Neural Nets Outperform Boosted Trees on Tabular Data?, May 2023. arXiv:2305.02997 [cs, stat].

[18] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

[19] Jaehyun Nam, Woomin Song, Seong Hyeon Park, Jihoon Tack, Sukmin Yun, Jaehyung Kim, and Jinwoo Shin. Semi-supervised tabular classification via in-context learning of large language models. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.

[20] Jaehyun Nam, Jihoon Tack, Kyungmin Lee, Hankook Lee, and Jinwoo Shin. STUNT: Few-shot Tabular Learning with Self-generated Tasks from Unlabeled Tables. In *International Conference on Learning Representations (ICLR)*. arXiv, March 2023. arXiv:2303.00918 [cs].

[21] OpenAI. Gpt-4 technical report, 2023.

[22] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

[23] Jiarui Qin, Weinan Zhang, Rong Su, Zhirong Liu, Weiwen Liu, Ruiming Tang, Xiuqiang He, and Yong Yu. Retrieval & interaction machine for tabular data prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1379–1389, 2021.

[24] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2023.

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

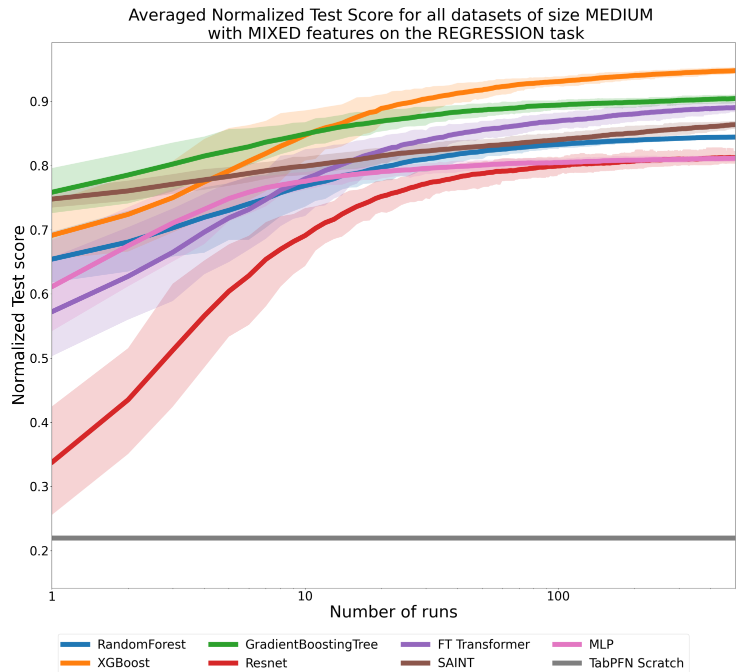
[26] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. In *NeurIPS Workshop: Table Representation Learning*. arXiv, June 2021. arXiv:2106.01342 [cs, stat].

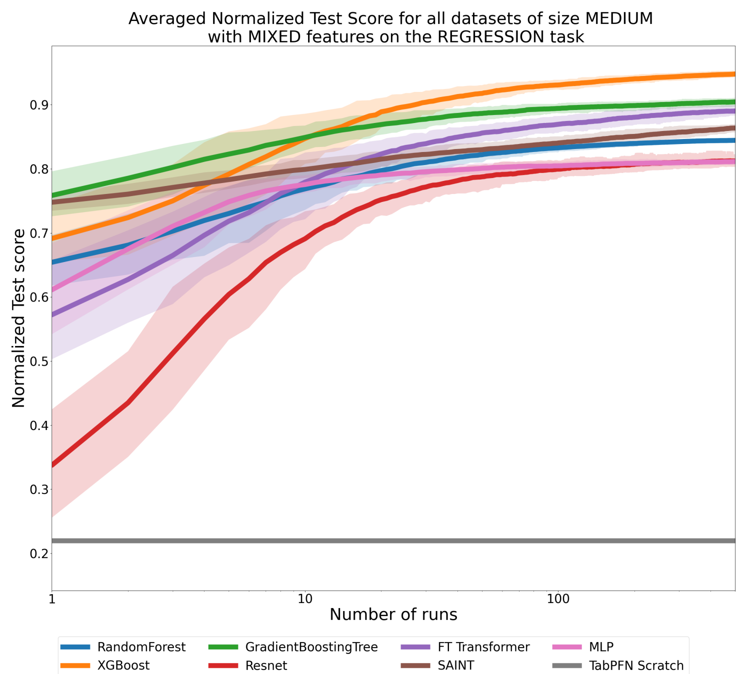
[27] Chao Ye, Guoshan Lu, Haobo Wang, Liyao Li, Sai Wu, Gang Chen, and Junbo Zhao. Ct-bert: Learning better tabular representations through cross-table pre-training. *arXiv preprint arXiv:2307.04308*, 2023.

[28] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain. In *Advances in Neural Information Processing Systems*, volume 33, pages 11033–11043. Curran Associates, Inc., 2020.

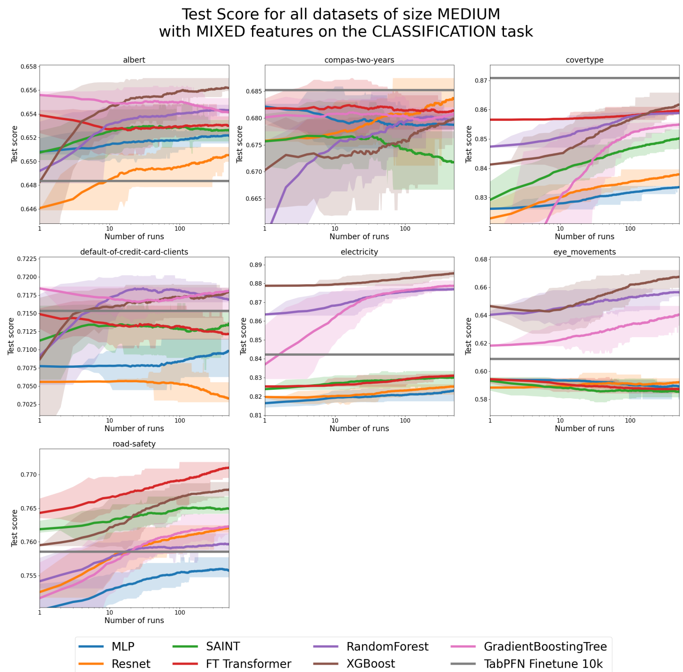
[29] Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. XTab: Cross-table Pretraining for Tabular Transformers. In *International Conference on Learning Representations (ICLR)*, June 2023.

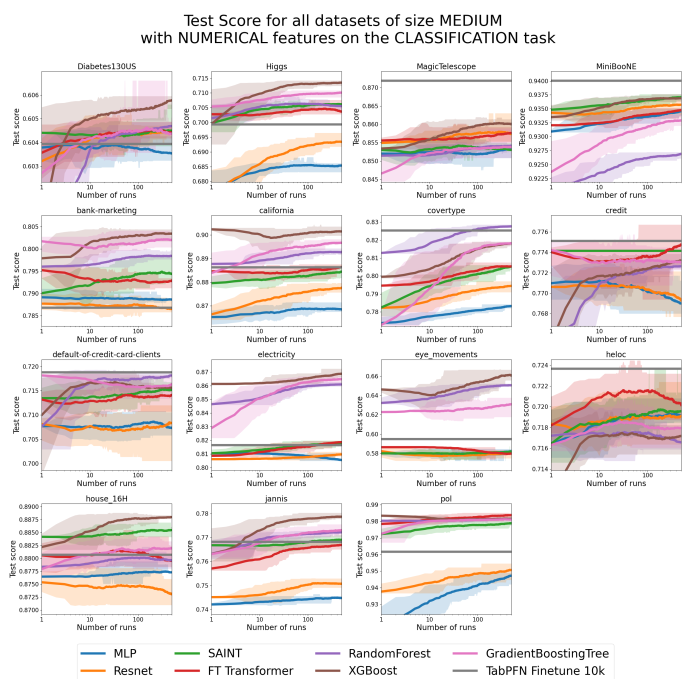
A Normalized Graphs for Regression from Scratch

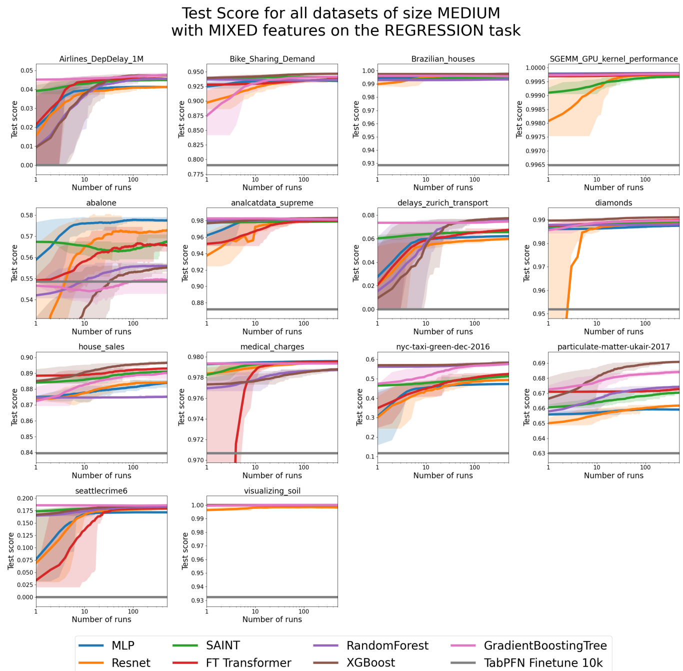


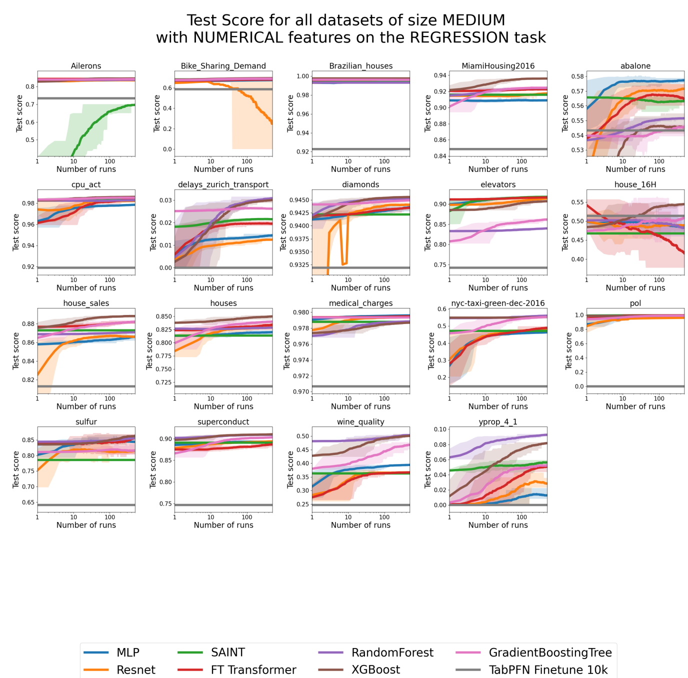


B Dataset Graphs






Test Score for all datasets of size MEDIUM
with MIXED features on the REGRESSION task
Airlines_DepDelay_1M
Bike_Sharing_Demand
Brazilian_houses
SGEMM_GPU_kernel_performance
abalone
analcatdata_supreme
delays_zurich_transport
diamonds
house_sales
medical_charges
nyc-taxi-green-dec-2016
particulate-matter-ukair-2017
seattlecrime6
visualizing_soil
Test score
Number of runs
MLP
Resnet
SAINT
FT Transformer
RandomForest
XGBoost
GradientBoostingTree
TabPFN Finetune 10k


Test Score for all datasets of size MEDIUM
with NUMERICAL features on the REGRESSION task
Ailerons
Bike_Sharing_Demand
Brazilian_houses
MiamiHousing2016
abalone
cpu_act
delays_zurich_transport
diamonds
elevators
house_16H
house_sales
houses
medical_charges
nyc-taxi-green-dec-2016
pol
sulfur
superconduct
wine_quality
yprop_4_1
Test score
Number of runs
MLP
Resnet
SAINT
FT Transformer
RandomForest
XGBoost
GradientBoostingTree
TabPFN Finetune 10k