

---

# Supplemental Material:

## A Closer Look at TabPFN v2: Understanding Its Strengths and Extending Its Capabilities

---

Anonymous Author(s)

Affiliation

Address

email

1 In the appendix, we provide additional details, discussions, and experimental results to complement  
2 the main paper:

- 3 • Appendix A: Additional related work and discussions on methods closely related to our study (cf.  
4 Section 2 of the main paper).
- 5 • Appendix B: Detailed descriptions of the comparison methods used in our evaluation (cf. Section 4  
6 of the main paper).
- 7 • Appendix C: Implementation details and descriptions of the TabPFN v2 variants compared in  
8 Section 7 of the main paper.
- 9 • Appendix D: Additional qualitative and quantitative results for feature extraction using TabPFN v2,  
10 supplementing Section 6 of the main paper.
- 11 • Appendix E: Analysis of the impact of feature engineering and ensemble strategies in TabPFN v2,  
12 as well as a meta-feature-based analysis to identify the conditions under which TabPFN v2 performs  
13 well or poorly.
- 14 • Appendix F: Complete results corresponding to the tables and figures referenced in the main paper.
- 15 • Appendix G: Limitation and social impact of the paper.

### 16 A Additional Related Work

17 **Learning with Tabular Data.** Tabular data is prevalent across diverse fields, including healthcare,  
18 finance, and education [28, 23, 39, 1]. Tree-based models, such as XGBoost [10], LightGBM [26],  
19 and CatBoost [37], have long dominated this domain. However, recent advances in deep neural  
20 networks (DNNs) have demonstrated strong potential for tabular data [5]. Popular architectures like  
21 multi-layer perceptrons [16, 25] and Transformers [22] have been adapted to tabular tasks, alongside  
22 custom architectures designed specifically for tabular data [27, 45].

23 Deep tabular methods can be broadly categorized into two types. The first type directly processes raw  
24 features [20, 15, 49], sometimes incorporating feature-specific encoding strategies [14]. The second  
25 type tokenizes features, transforming an example into a set of tokens [43, 22, 40]. Comprehensive  
26 benchmarks have been developed to evaluate these methods across diverse datasets [17, 33, 48, 41],  
27 highlighting the strengths and weaknesses of deep tabular models in various scenarios.

28 **Variants of TabPFN.** TabPFN’s success stems from its pre-training on massive synthetic datasets,  
29 enabling strong in-context learning performance on small-scale classification tasks [18]. Motivated  
30 by its capabilities, researchers have explored a variety of applications, including tabular data gener-  
31 ation [31], anomaly detection [42], and time series forecasting [21]. [35] provided a bias-variance  
32 analysis of TabPFN, offering insight into its generalization behavior. Another line of research fo-  
33 cuses on improving scalability by addressing TabPFN’s sensitivity to context size [11, 47]. Further  
34 strategies to enhance downstream performance include context adaptation with nearest neighbor [44],  
35 partial fine-tuning [12, 29], pre-training on real-world datasets [32], scalable ensemble [30], and more

powerful and efficient pre-training on synthetic data [38]. Most of these variants remain restricted to classification tasks due to limitations in TabPFN v1.

The recently introduced TabPFN v2 [19] extends TabPFN to support regression tasks and accommodate larger context sizes. In this paper, we conduct a comprehensive evaluation of TabPFN v2, analyze its strengths, and introduce methods to overcome its scalability and applicability challenges.

## B Evaluation Details

**Experimental Compute Resources.** All experiments were conducted using 4 NVIDIA RTX 6000 Ada GPUs and 2 Intel(R) Xeon(R) Platinum 8352V CPUs.

Please refer [48] for details of the 300 small to medium datasets. For high-dimensional datasets, we selected 18 datasets with more than 2000 features from the scikit-feature repository. Detailed statistics of high-dimensional datasets and large-scale datasets are reported in Table 1 and Table 2.

We follow [48] and use different colors to represent various categories of methods in the result figures, ensuring clarity and easy comparison. In Figure 1 (right) and Figure 2 of the main paper, we compare the following methods:

- **Classical Methods** (orange): The classical methods include Dummy, Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Naive Bayes, Linear Regression, and DNNR [34], which serve as basic baselines for classification and regression tasks.
- **Tree-based Methods** (green): Tree-based methods such as Random Forest [6], XGBoost [10], LightGBM [26], and CatBoost [37] are known for their high performance on tabular data.
- **MLP variants** (red): MLP variants, including vanilla MLP, MLP-PLR, Self-Normalizing Neural Networks [27], Residual Network [16], and TabM [13] enhance the flexibility and generalization of traditional MLP architectures through advanced regularization and residual connections.
- **Special Architectures** (blue): Methods with specially designed architectures, such as DCNv2 [45], DANets [8], and TabCaps [7], focus on improving feature interaction and abstraction to capture complex relationships in tabular data.
- **Token-based Methods** (cyan): Token-based methods like AutoInt [43], TabTransformer [22], FT-Transformer [16], and ExcelFormer [9] represent features as tokens, enabling models to capture higher-order interactions through attention mechanisms.
- **Regularization-based Methods** (yellow): Regularization-based methods, including TANGOS [24], SwitchTab [46], and PTaRL [50], aim to improve model generalization by incorporating regularization techniques during training to enhance the robustness of predictions.
- **Tree-mimic Methods** (teal): Tree-mimic methods, such as NODE [36], GrowNet [4], and TabNet [3], combine the interpretability of decision trees with the power of deep learning, employing attention mechanisms to select important features.
- **Context-based Methods** (purple): Context-based methods like TabR [15] and ModernNCA [49] leverage contextual information from the training data to improve predictions by utilizing neighborhood-based and in-context learning strategies.

In addition to the aforementioned methods, for other experimental results, we will demonstrate the performance of **TabPFN v2 and its variants**, which are represented by emerald teal (dark teal), ensuring that their experimental effects are clearly distinguished from the other methods.

**Remark.** The standard checkpoint released by TabPFN employs a feature grouping size of 2, which complicates the analysis of individual feature embeddings and inter-feature relationships. To facilitate such analysis, we use a modified checkpoint with *group size=1* for the experiments in Figure 3 of the main paper, which is available at HuggingFace.

## C Details of Comparison Baselines

This section provides implementation details of the TabPFN v2 variants used in our analysis for high-dimensional, many-class, and large-scale tasks (cf. Figure 5 in the main paper).

**High-Dimensional Classification.** To address high-dimensional tasks, we introduce a baseline variant, TabPFN v2-PCA, which incorporates dimensionality reduction. Specifically, TabPFN v2-PCA reduces the feature dimension to 500 using PCA to satisfy the input constraints of TabPFN v2.

Table 1: Dataset Information for High-Dimensional Data Experiments: A collection of 18 datasets with varying numbers of instances, features, and classes used in our high-dimensional experiments.

Dataset	#Instances	#Features	#Classes	Dataset	#Instances	#Features	#Classes
BASEHOCK	1993	4862	2	lung	203	3312	5
PCMAC	1943	3289	2	warpPIE10P	210	2420	10
RELATHE	1427	4322	2	orlraws10P	100	10304	10
ALLAML	72	7129	2	Prostate_GE	102	5966	2
CLL_SUB_111	111	11340	3	SMK_CAN_187	187	19993	2
colon	62	2000	2	warpAR10P	130	2400	10
GLI_85	85	22283	2	arcene	200	10000	2
GLIOMA	50	4434	4	gisette	7000	5000	2
leukemia	72	7070	2	TOX_171	171	5748	4

Table 2: Dataset Information for Large-scale Data Experiments.

Dataset	#Instances	#Features	#Classes	Dataset	#Instances	#Features	#Classes
BNG(credit-a)	1,000,000	15	2	CDC_Indicators	253,680	21	2
Higgs	1,000,000	28	2	Smoking_signal	991,346	23	2
nomao	34,465	118	2	sf-police-incidents	2,215,023	8	2
Data_Crowdfunding	671,025	11	4	Fashion-MNIST	70,000	784	10
coverttype	581,012	54	7	jannis	83,733	54	4
poker-hand	1,025,009	10	10	volkert	58,310	180	10
Airlines_DepDelay	10,000,000	9	-	Wave_Energy_Farm	36,043	99	-
UJIndoorLoc	21,048	520	-	blogfeedback	60,021	276	-
microsoft	1,200,192	136	-	yahoo	709,877	699	-

This process is repeated multiple times with different PCA projections, and the resulting predictions are aggregated via bagging to improve robustness.

**Many-Class Classification.** We consider the following variants of TabPFN v2 to address the 10-class limit in classification tasks:

- TabPFN v2-ECOC: We use the implementation provided in the official TabPFN extensions repository, which applies Error-Correcting Output Codes (ECOC).
- TabPFN v2-DPT: We encode each class label as a  $t$ -digit decimal string and train a separate TabPFN v2 to predict each digit. For instance, a 15-class problem is decomposed into two subproblems: one for the tens digit (classes  $\{0, 1\}$ ) and one for the ones digit (classes  $\{0, \dots, 9\}$ ). The predicted digits are then decoded to recover the final class label. The same strategy is also proposed in [32].

We implement TabPFN v2 \* based on TabPFN v2-DPT for efficiency. Specifically, TabPFN v2 \* permutes the class-to-digit mapping  $\sqrt{C}$  times. For fair comparison, we also increase the number of ensembles in TabPFN v2-DPT to  $\sqrt{C}$  per digit to match the total number of predictions.

**Large-Scale Tasks.** As the size of training data increases, the benefit of pre-trained knowledge may diminish. We consider the following variants of TabPFN v2 to scale to larger datasets:

- TabPFN v2-DT: A shallow decision tree is trained with a minimum split size of 10,000. The tree partitions the dataset into smaller subsets, and a separate TabPFN v2 model is applied to each leaf node. At inference, a test instance is routed through the tree to a corresponding leaf, where it is predicted by the respective TabPFN v2 model.
- TabPFN v2-B: A bagging-based variant that randomly samples 10,000 training examples four times and aggregates their predictions.
- TabPFN v2-K: Selects a representative subset of 10,000 training examples based on proximity to KMeans-derived prototypes.

Our TabPFN v2 \*-DF is an extension of TabPFN v2-DT to a forest-based ensemble. Specifically, we sample 32 subsets from the original training data, each containing 60% of the samples (without replacement), and train a separate TabPFN v2\*-DT model on each subset. During inference, predictions from all 32 models are aggregated—*e.g.*, by majority voting or averaging—depending on the task type.

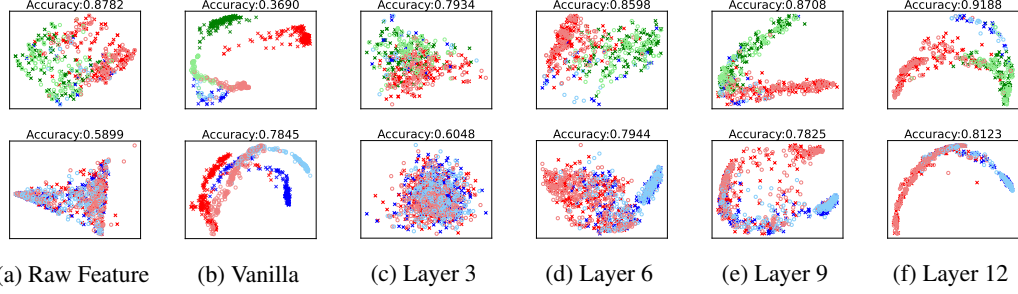


Figure 1: Visualization of the extracted instance features from two datasets: *website\_phishing* (first row, three classes) and *KDD* (second row, binary). Colors indicate classes, where Darker crosses and lighter circles denote training and test examples, respectively. (a) shows the raw input features (e.g.,  $x_i$ ), while (b) presents embeddings from the vanilla strategy. (c)-(f) display embeddings produced by our method at different layers. Classification accuracy is reported by training a linear logistic regression model on the training embeddings and evaluating on the test set.

Table 3: Average rank (lower is better) of TabPFN v2 and a linear classifier trained on the extracted embeddings across 29 classification datasets. In addition to the supervised feature extraction strategy considered in the main paper (including the vanilla one, our leave-one-fold-out, and the version based on the combined features), we compare with two *unsupervised* embedding extraction approaches by appending a column of *dummy* labels with zero values and *permuting* each column as labels, respectively.

↓	TabPFN v2	Vanilla	Dummy	Permute	Ours	Combined
Rank	2.66	5.72	4.90	3.69	2.16	<b>1.88</b>

## D Additional Feature Extraction Results

In this section, we provide further results to validate the effectiveness of our leave-one-fold-out strategy for feature extraction with TabPFN v2. We also compare this supervised approach to two unsupervised embedding extraction methods.

### D.1 Additional Results for Figure 4 in the Main Paper

Figure 1 visualizes instance embeddings extracted using our leave-one-fold-out strategy on two datasets: *website\_phishing* (first row, three classes) and *KDD* (second row, binary). Training and test examples are shown as darker crosses and lighter circles, respectively, and colored by class. These visualizations support our claim that TabPFN v2, when combined with our proposed strategy, can serve as an effective feature encoder.

### D.2 Comparison Between Supervised and Unsupervised Feature Extraction

Our leave-one-fold-out strategy explicitly incorporates label information and accounts for the distinct roles of training and test instances. To further understand the nature of the extracted embeddings, we compare this supervised strategy to two unsupervised alternatives provided by the TabPFN extensions repository.

Let  $X \in \mathbb{R}^{n \times d}$  denote a dataset with  $n$  samples and  $d$  features:

- **Unsupervised-Dummy:** Each sample is paired with a constant pseudo-target  $\mathbf{y} = \mathbf{0} \in \mathbb{R}^n$ , forming a regression task. The embedding  $E_D \in \mathbb{R}^{n \times h}$  is obtained by extracting the output tokens of the TabPFN regressor.
- **Unsupervised-Permute:** For each feature  $j \in \{1, \dots, d\}$ , we treat  $\mathbf{x}^{(j)} = X_{:,j}$  as a pseudo-target and use the remaining features  $X^{(-j)}$  as input. Depending on the type of  $\mathbf{x}^{(j)}$ , TabPFN is applied in classification or regression mode to obtain  $E^{(j)} \in \mathbb{R}^{n \times h}$ . These embeddings are concatenated to

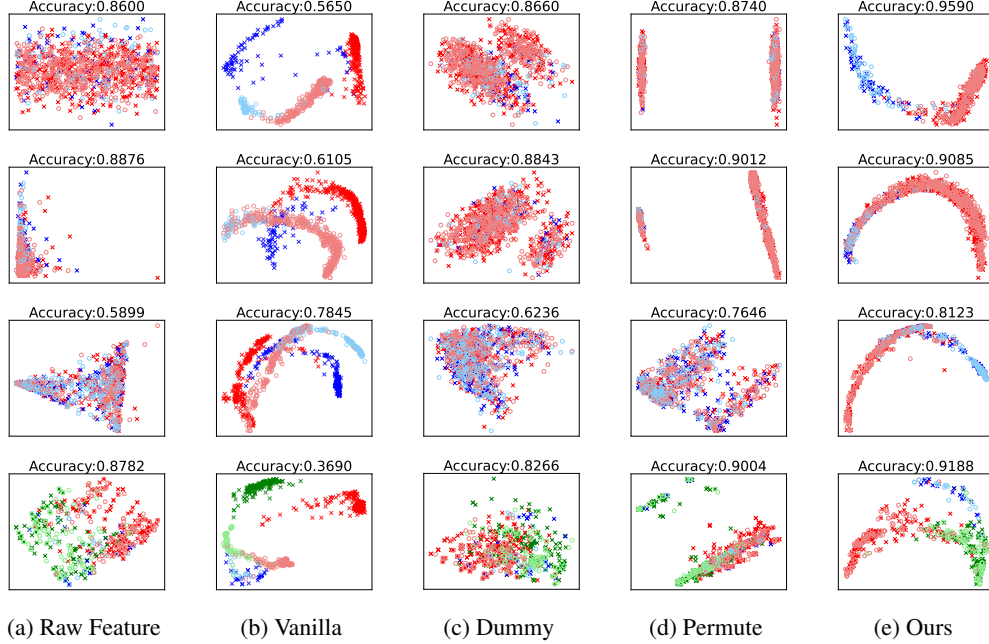


Figure 2: Comparison between unsupervised and supervised (ours) feature extraction. Visualization of extracted embeddings for four datasets: *churn* (first row, two classes), *bank* (second row, two classes), *KDD* (third row, two classes), and *website\_phishing* (fourth row, three classes). We use crosses to denote training examples and circles to denote test examples. (a) shows the raw features, while (b) presents the embeddings extracted using the vanilla strategy. (c) and (d) refer to *unsupervised* embedding extraction approaches by appending a column of dummy labels with zero values and permuting each column as labels, respectively. (e) depicts the embeddings obtained using our proposed methods. The accuracy value is calculated by training a linear model (logistic regression) over the extracted embeddings on the training set and predicting on the test set.

a high-dimensional form:

$$E_P = \text{concat}(E^{(1)}, E^{(2)}, \dots, E^{(d)}) \in \mathbb{R}^{n \times (d \cdot h)}.$$

We compare these unsupervised methods with our supervised leave-one-fold-out strategy in Table 3. Overall, unsupervised approaches underperform compared to the supervised ones and fail to recover the classification ability of TabPFN v2. This performance gap arises because label information is introduced only post hoc via a linear classifier rather than during embedding extraction. Among the unsupervised methods, the permutation-based approach performs better, likely due to its ability to encode attribute-specific structure.

Figure 2 presents a visual comparison of embeddings produced by the three methods using the same color and marker scheme as in Figure 1. Since unsupervised methods lack label supervision during embedding generation, their embeddings tend to scatter broadly without forming well-separated clusters by class. These results further highlight the contrasting goals of supervised and unsupervised strategies—class separation versus feature distribution coverage, respectively.

## E Influence of Key Modules and Meta-Feature Analysis

We investigate the influence of two key components in TabPFN v2, *i.e.*, the feature engineering that pre-processes the raw features of a given tabular dataset and the post-hoc ensembling. In addition, we analyze the conditions under which TabPFN v2 performs well or poorly through a meta-feature-based classification analysis.

**Feature engineering.** TabPFN v2 pre-processes the features of a given tabular dataset with various strategies, such as quantile, category shuffling, SVD, and power transform. Specifically, we examine the effects of adding fingerprint features (`add_fingerprint_feature`) and polynomial features

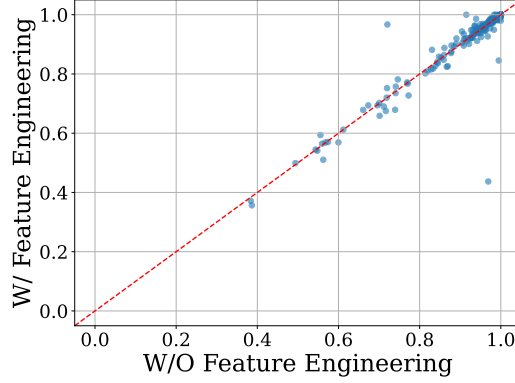


Figure 3: Scatter plot comparing the normalized Accuracy/ $R^2$  scores. The x-axis represents the normalized Accuracy/ $R^2$  scores without Feature Engineering, while the y-axis represents the normalized Accuracy/ $R^2$  scores with Feature Engineering. The red dashed line ( $y = x$ ) serves as a reference, indicating equal performance.

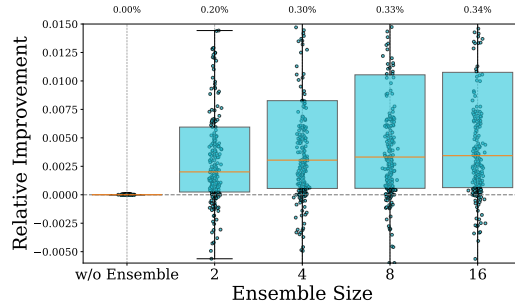


Figure 4: Box plot of relative performance improvements of TabPFN v2 with post hoc ensembling (PHE) across different ensemble sizes (2, 4, 8, and 16 base models). The relative improvement is calculated as the performance gain over the non-ensemble model, where higher values indicate stronger performance. The box plots show the median, interquartile range (IQR), and outliers for each ensemble size.

156 (polynomial\_features) to the raw tabular data. The results indicate that TabPFN v2 performs well  
 157 even without the use of these engineered features, suggesting that, for the benchmark datasets of [48],  
 158 these specific feature engineering techniques do not provide a significant improvement. This finding  
 159 highlights the robustness of TabPFN v2 and its ability to handle raw features effectively, without  
 160 the need for extensive pre-processing or feature construction. We show the influence of this step  
 161 in Figure 3.

162 **Model ensemble.** Post hoc ensembling (PHE) involves applying TabPFN v2 to the datasets multiple  
 163 times with different perturbations and aggregating the predictions of these base models at different  
 164 temperatures. We show the change of performance of TabPFN v2 w.r.t. the number of ensemble  
 165 numbers (*i.e.*, the number of base models) in Figure 4. On the benchmark of [48], we observe that,  
 166 overall, ensemble methods improve performance, with larger ensemble sizes yielding better results.  
 167 However, we also note that even without ensembling, TabPFN v2 performs exceptionally well, and  
 168 the relative performance gain from ensembling is limited. This suggests that while ensembling can  
 169 provide further improvements, the base TabPFN v2 model is already highly effective on its own.  
 170 The equivariant property described in [2] provides insight into this phenomenon. Since TabPFN v2  
 171 introduces random tokens to handle heterogeneous features, the model becomes less sensitive to the  
 172 arbitrary ordering of features, effectively enforcing equivariance in this aspect. As a result, the benefits  
 173 of ensembling through feature order permutations are less pronounced compared to TabPFN v1.

174 **Meta-Feature Analysis of TabPFN v2 Performance.** To better understand the conditions under  
 175 which TabPFN v2 performs well or poorly, we conducted a meta-learning-based classification  
 176 analysis using 300 datasets. Specifically, we used the average rank of TabPFN v2 across datasets as a  
 177 performance indicator. The threshold for classification was set at the mean rank, 6.31. Datasets where

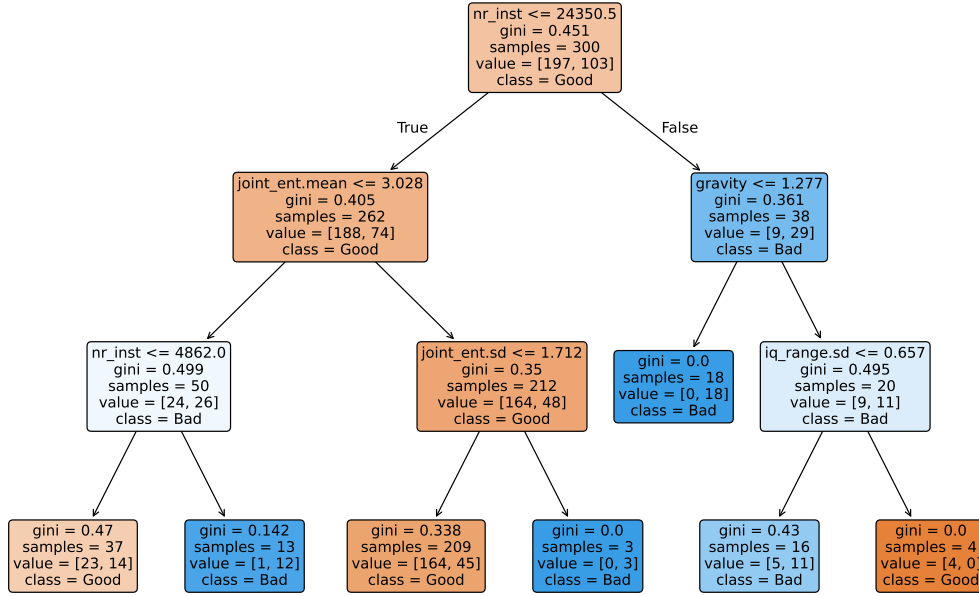


Figure 5: Decision tree for predicting TabPFN v2 performance (Good vs. Bad) based on dataset characteristics, constructed from experiments on 300 datasets. The tree splits on meta-features such as number of instances (`nr_inst`), joint entropy (`joint_ent.mean`), number of numerical features (`nr_num`), distance between minority and majority classes’ center of mass (`gravity`) and interquartile range (`iq_range`) statistics. Each split is chosen to maximize information gain. The leaf nodes indicate the predicted performance class, the Gini impurity, and class distribution. This tree provides insights into the types of datasets where TabPFN v2 is expected to perform well.

TabPFN v2 achieved a rank lower than or equal to 6.31 were labeled as “Good”, while those with a higher rank were labeled as “Bad”. We extracted meta-features from each dataset and used them to train a decision tree classifier, aiming to distinguish between the “Good” and “Bad” categories. All the meta-features utilized in this task are detailed in Table 4, accompanied by their respective explanations. A visual depiction of a simple depth-3 decision tree is shown in Figure 5, and the tree reveals key factors that influence the effectiveness of TabPFN v2. The decision tree visualizes how to predict whether TabPFN v2 performs well (“Good”) or poorly (“Bad”) on a given dataset, based on dataset meta-features: The root node splits on the number of instances (`nr_inst`), indicating that TabPFN v2 tends to perform better on datasets with fewer than 24,350 samples. For these smaller datasets, the left subtree further splits on the mean joint entropy (`joint_ent.mean`), where higher values (greater than 3.028) are associated with improved performance. For datasets with lower mean joint entropy ( $\leq 3.028$ ), TabPFN v2 also tends to perform well when the number of rows is relatively small ( $\leq 4,862$ ). In contrast, the right subtree, which represents larger datasets, reveals that a low standard deviation of the interquartile range (`iq_range.std`) across features ( $\leq 0.657$ ) is linked to poorer model performance.

## F Detailed Results

We list the detailed results of TabPFN v2 and our extensions on various benchmarks.

- We present the **main results of TabPFN v2** on 300 datasets in Table 5. The table includes accuracy for classification tasks and RMSE (Root Mean Squared Error) for regression tasks, along with the corresponding mean and standard deviation for each dataset. Notably, we excluded 27 datasets from these results in Table 5, as they were used by TabPFN v2 to select the best checkpoint. These excluded datasets, which are not shown in Figure 1 (right) and Figure 2 of the main paper, include:



Table 4: Meta-features used in the meta-feature analysis of TabPFN v2 performance.

Meta-Feature	Explanation
attr_conc	The concentration coef. of each pair of distinct attributes.
class_conc	The concentration coefficient between each attribute and class.
class_ent	The target attribute Shannon’s entropy.
inst_to_attr	The ratio between the number of instances and attributes.
mean	The mean value of each attribute.
sd	The standard deviation of each attribute.
var	The variance of each attribute.
range	The range (max - min) of each attribute.
iq_range	The interquartile range (IQR) of each attribute.
nr_attr	The total number of attributes.
sparsity	The (possibly normalized) sparsity metric for each attribute.
t_mean	The trimmed mean of each attribute.
nr_bin	The number of binary attributes.
nr_cat	The number of categorical attributes.
nr_num	The number of numeric features.
nr_norm	The number of attributes normally distributed based in a given method.
nr_cor_attr	The number of distinct highly correlated pair of attributes.
gravity	The distance between minority and majority classes’ center of mass.
nr_class	The number of distinct classes.
joint_ent	The joint entropy between each attribute and class.
attr_ent	Shannon’s entropy for each predictive attribute.
cov	The absolute value of the covariance of distinct dataset attribute pairs.
eigenvalues	The eigenvalues of covariance matrix from dataset.
eq_num_attr	The number of attributes equivalent for a predictive task.
max	The maximum value from each attribute.
min	The minimum value from each attribute.
median	The median value from each attribute.
freq_class	The relative frequency of each distinct class.
mad	The Median Absolute Deviation (MAD) adjusted by a factor.
mad	The Median Absolute Deviation (MAD) adjusted by a factor.
mut_inf	The mutual information between each attribute and target.
nr_inst	The number of instances (rows) in the dataset.
nr_outliers	The number of attributes with at least one outlier value.
ns_ratio	The noisiness of attributes.
imbalance_ratio	The ratio of the number of instances in the minority to the majority class.
attr_to_inst	The ratio between the number of attributes.

- 200 (1) ada\_prior, allbp, baseball, delta\_ailerons, eye\_movements, eye\_movements\_bin,  
201 GAMETES\_Epistasis\_2-Way\_20atts\_0.1H\_EDM-1\_1, hill-valley, JapaneseVowels,  
202 jungle\_chess\_2pcs\_raw\_endgame\_complete, led24, longitudinal-survey, page-blocks,  
203 ringnorm, rl, thyroid-ann, waveform-5000,  
204 (2) debutanizer, delta\_elevators, mauna-loa-atmospheric, puma32H, stock\_fardamento02, trea-  
205 sury, weather\_izmir, wind.

- 206 • In Table 6, we showcase the **performance of various models on 18 high-dimensional datasets**.  
207 The results display the mean accuracy of different models, including ModernNCA (MNCA), MLP,  
208 KNN, RealMLP, XGBoost (XGB), Random Forest (RForest), Logistic Regression (LogReg), and  
209 TabPFN v2 (PFN-v2), along with variants like TabPFN v2-pca and TabPFN v2\*. This highlights  
210 the ability of these models to handle high-dimensional data with many features.
- 211 • We demonstrate the **performance of various models on 12 multi-class classification tasks**  
212 with more than 10 classes in Table 7. The table provides the mean accuracy of models like  
213 KNN, TabPFN-v2\*, XGBoost (XGB), CatBoost (CatB), Random Forest (RForest), ModernNCA  
214 (MNCA), MLP, Logistic Regression (LogReg), and RealMLP, showcasing how they perform on  
215 multi-class tasks with a larger number of classes. Additionally, we compare **PFN-v2-ECOC**, a  
216 multi-class classification solution provided by [19]. This method extends TabPFN-v2 by leveraging  
217 Error-Correcting Output Codes (ECOC) to enhance multi-class classification performance.<sup>1</sup>

<sup>1</sup>[https://github.com/PriorLabs/tabpfm-community/blob/main/src/tabpfm\\_extensions/many\\_class/many\\_class\\_classifier.py](https://github.com/PriorLabs/tabpfm-community/blob/main/src/tabpfm_extensions/many_class/many_class_classifier.py)



- In Table 8, we compare the **performance of various models on 18 large-scale datasets**. The results show the mean accuracy or RMSE for MLP, Logistic/Linear Regression (LR), KNN, XGBoost (XGB), Random Forest (RForest), CatBoost (CatB), ModernNCA (MNCA), RealMLP, and different versions of TabPFN v2 (PFNV2, PNFv2 with K-means, PNFv2 with Bagging, and PNFv2\*). This illustrates the models’ performance on large-scale datasets.
- In Table 9, we show the **performance of TabPFN v2 and the extracted feature embeddings** across 29 classification datasets. The table includes average classification accuracy for each dataset when using feature embeddings from different transformer layers (Layer 6, Layer 9, Layer 12), as well as a combined approach where embeddings from multiple layers are concatenated. The “selected layers” column indicates the layers chosen based on validation set performance, offering insights into how different layers contribute to overall model performance. In addition to evaluating the performance of TabPFN v2 and the extracted feature embeddings, we also compared the results with embeddings obtained using the vanilla strategy (Vanilla).

Table 5: Main results of TabPFN v2 on 300 datasets, including accuracy (for classification tasks) and RMSE (for regression tasks), along with the corresponding mean and standard deviation for each dataset. Among the 300 datasets, 200 are classification datasets, and 100 are regression datasets. The results demonstrate the effectiveness of TabPFN v2 across both classification and regression tasks.

Dataset	Mean + Std	Dataset	Mean + Std
ASP-POTASSCO-classification	43.50 ± 1.27	Amazon_employee_access	94.22 ± 0.04
BLE_RSSI_localization	73.37 ± 0.15	BNG(breast-w)	98.56 ± 0.07
BNG(cmc)	57.69 ± 0.17	BNG(tic-tac-toe)	79.42 ± 0.26
Bank_Customer_Churn_Dataset	87.53 ± 0.12	Basketball_c	70.65 ± 0.47
California-Housing-Classification	91.47 ± 0.17	Cardiovascular-Disease-dataset	72.92 ± 0.13
Click_prediction_small	83.29 ± 0.03	Contaminant-10.0GHz	94.42 ± 0.36
Contaminant-10.5GHz	95.17 ± 0.32	Contaminant-11.0GHz	93.93 ± 0.50
Contaminant-9.0GHz	93.01 ± 0.47	Contaminant-9.5GHz	93.21 ± 0.50
Credit_c	69.98 ± 0.15	Customer_Personality_Analysis	90.03 ± 0.21
Diabetic_Retinopathy_Debrecen	72.81 ± 1.07	E-CommereShippingData	67.54 ± 0.21
Employee	84.80 ± 0.30	FICO-HELOC-cleaned	75.35 ± 0.21
FOREX_audcad-day-High	74.51 ± 0.51	FOREX_audcad-hour-High	71.01 ± 0.20
FOREX_audchf-day-High	76.66 ± 0.45	FOREX_audjpy-day-High	78.00 ± 0.28
FOREX_audjpy-hour-High	71.41 ± 0.32	FOREX_audsgd-hour-High	69.81 ± 0.39
FOREX_audusd-hour-High	69.57 ± 0.48	FOREX_cadjpy-day-High	71.68 ± 0.53
FOREX_cadjpy-hour-High	70.55 ± 0.40	Firm-Teacher-Direction	84.42 ± 0.47
Fitness_Club_c	79.67 ± 0.24	GAMETES_Epistasis	68.75 ± 0.82
GAMETES_Heterogeneity	65.90 ± 1.84	Gender_Gap_in_Spanish_WP	60.58 ± 0.24
GesturePhaseSegmentationProcessed	71.36 ± 1.15	HR_Analytics	80.02 ± 0.13
Heart-Disease-Dataset	91.23 ± 0.54	INNHOTelsGroup	87.98 ± 0.23
Indian_pines	96.41 ± 0.23	Insurance	75.75 ± 0.00
Intersectional-Bias-Assessment	94.73 ± 0.13	Is-this-a-good-customer	88.41 ± 0.00
JapaneseVowels	99.68 ± 0.08	KDD	80.14 ± 0.46
KDDCup09_upselling	81.06 ± 0.26	Long	99.88 ± 0.00
MIC	90.20 ± 0.56	MagicTelescope	88.13 ± 0.21
Marketing_Campaign	88.11 ± 0.41	Mobile_Price_Classification	97.10 ± 0.29
Nutrition_Health_Survey	83.45 ± 0.22	Performance-Prediction	73.23 ± 0.61
PhishingWebsites	96.74 ± 0.13	PieChart3	87.31 ± 0.28
Pima_Indians_Diabetes_Database	75.93 ± 0.66	PizzaCutter3	88.20 ± 0.45
Pumpkin_Seeds	87.93 ± 0.21	QSAR_biodegradation	88.50 ± 0.50
Rain_in_Australia	83.88 ± 0.11	SDSS17	97.33 ± 0.06
Shipping	68.73 ± 0.40	Telecom_Churn_Dataset	95.18 ± 0.50
UJI_Pen_Characters	45.71 ± 2.16	VulNoneVul	98.95 ± 0.00
Water_Quality_and_Potability	65.49 ± 0.50	Waterstress	71.37 ± 0.96
Wilt	99.28 ± 0.06	abalone	63.58 ± 0.38
accelerometer	73.96 ± 1.32	ada	85.40 ± 0.25
ada_agnostic	83.99 ± 0.34	ada_prior	85.32 ± 0.19
adult	85.93 ± 0.12	airlines_2000	62.28 ± 0.48
allbp	97.85 ± 0.19	allrep	98.65 ± 0.12
analcadata_authorship	99.72 ± 0.29	artificial-characters	73.90 ± 0.99
autoUniv-au4-2500	69.81 ± 1.00	autoUniv-au7-1100	41.18 ± 1.58
bank	90.86 ± 0.19	banknote_authentication	55.64 ± 0.18
baseball	93.81 ± 0.40	car-evaluation	98.29 ± 0.22
churn	96.33 ± 0.28	cmc	59.59 ± 0.49
company_bankruptcy_prediction	97.33 ± 0.07	compass	71.05 ± 0.29
connect-4	76.78 ± 0.35	contraceptive_method_choice	62.10 ± 0.37
credit	78.10 ± 0.11	credit-g	79.50 ± 0.81
customer_satisfaction_in_airline	94.79 ± 0.11	dabetes_130-us_hospitals	63.08 ± 0.07
default_of_credit_card_clients	82.63 ± 0.08	delta_aileron	95.47 ± 0.09
dis	99.07 ± 0.14	dna	97.25 ± 0.20
drug_consumption	40.32 ± 0.00	dry_bean_dataset	92.76 ± 0.10
eeg-eye-state	98.34 ± 0.12	electricity	86.57 ± 0.45
estimation_of_obesity_levels	98.66 ± 0.24	eucalyptus	72.88 ± 1.17
eye_movements	77.03 ± 1.68	eye_movements_bin	67.28 ± 2.60

first-order-theorem-proving	61.12 ± 0.70	gas-drift	99.47 ± 0.04
golf_play_dataset_extended	92.60 ± 0.44	helen	33.32 ± 0.21
heloc	72.75 ± 0.20	hill-valley	98.33 ± 0.52
house_16H	88.55 ± 0.18	htru	97.95 ± 0.06
ibm-employee-performance	100.0 ± 0.00	in_vehicle_coupon	73.20 ± 0.35
internet_firewall	92.85 ± 0.30	internet_usage	54.34 ± 2.64
jasmine	81.34 ± 0.42	jm1	81.32 ± 0.10
jungle_chess_2pcs_raw_endgame	85.97 ± 1.82	kc1	86.65 ± 0.34
kdd_ipums_la_97-small	88.50 ± 0.12	kr-vs-k	78.46 ± 1.01
kr-vs-kp	99.64 ± 0.15	kropt	77.96 ± 0.63
law-school-admission-bianry	100.0 ± 0.00	led24	73.29 ± 0.62
led7	73.99 ± 0.31	letter	97.57 ± 0.10
madeline	90.72 ± 0.48	mammography	98.71 ± 0.05
maternal_health_risk	83.28 ± 0.64	mfeat-factors	96.98 ± 0.28
mfeat-fourier	89.85 ± 0.86	mfeat-karhunen	96.42 ± 0.24
mfeat-morphological	76.63 ± 0.50	mfeat-pixel	96.10 ± 0.32
mfeat-zernike	84.10 ± 0.87	mice_protein_expression	100.0 ± 0.00
microaggregation2	62.80 ± 0.14	mobile_c36_oversampling	98.11 ± 0.08
mozilla4	93.58 ± 0.16	naticusdroid+android+permissions	96.41 ± 0.10
national-longitudinal-survey-binary	100.0 ± 0.00	okcupid_stem	74.47 ± 0.12
one-hundred-plants-margin	88.56 ± 0.74	one-hundred-plants-shape	79.52 ± 0.72
one-hundred-plants-texture	90.94 ± 0.75	online_shoppers	90.65 ± 0.10
optdigits	98.59 ± 0.12	ozone-level-8hr	94.92 ± 0.25
ozone_level	97.86 ± 0.11	page-blocks	97.67 ± 0.10
pc1	93.51 ± 0.46	pc3	88.78 ± 0.27
pc4	90.87 ± 0.36	pendigits	99.56 ± 0.06
philippine	84.20 ± 1.24	phoneme	88.47 ± 0.35
pol	98.80 ± 0.09	predict_students_dropout	78.11 ± 0.38
rice_cammeo_and_osmancik	92.74 ± 0.23	ringnorm	98.00 ± 0.13
rl	86.04 ± 0.44	satimage	92.30 ± 0.29
segment	93.91 ± 0.19	seismic+bumps	93.40 ± 0.08
semeion	92.41 ± 0.94	shill-bidding	90.31 ± 0.18
shuttle	86.97 ± 0.11	shuttle	99.86 ± 0.04
spambase	94.85 ± 0.19	splice	96.61 ± 0.22
sports_articles	84.93 ± 0.40	statlog	72.13 ± 0.97
steel_plates_faults	84.68 ± 0.55	svmguide3	85.54 ± 0.54
sylvine	97.30 ± 0.27	taiwanese_bankruptcy_prediction	97.20 ± 0.07
telco-customer-churn	80.29 ± 0.28	texture	100.0 ± 0.00
W_thyroid	99.48 ± 0.06	thyroid-ann	99.34 ± 0.08
thyroid-dis	68.75 ± 0.34	turiye_student_evaluation	51.74 ± 0.18
twonorm	97.94 ± 0.08	vehicle	84.31 ± 1.29
walking-activity	61.22 ± 0.22	wall-robot-navigation	99.44 ± 0.10
water_quality	90.12 ± 0.12	waveform-5000	86.29 ± 0.26
waveform_v1	86.59 ± 0.25	website_phishing	90.48 ± 0.48
wine	75.12 ± 0.72	wine-quality-red	58.35 ± 0.76
wine-quality-white	64.15 ± 0.69	yeast	60.18 ± 0.65

---

1000-Cameras-Dataset	607.71 ± 6.61	2dplanes	1.01 ± 0.00
RSSI_Estimation	0.00068 ± 0.00	RSSI_Estimation1	0.00092 ± 0.00
Abalone_reg	2.08 ± 0.00	Ailerons	0.00015 ± 0.00
Fiat	716.20 ± 4.05	BNG(echoMonths)	11.41 ± 0.03
BNG(lowbwt)	455.27 ± 0.78	BNG(mv)	4.63 ± 0.01
BNG(stock)	2.95 ± 0.02	Bias_correction_r	0.60 ± 0.01
Bias_correction_r_2	0.52 ± 0.01	Brazilian_houses_reproduced	0.01 ± 0.00
CPMP-2015-regression	478.02 ± 5.40	CPS1988	364.02 ± 0.24
CookbookReviews	1.52 ± 0.02	Data_Science_Salaries	60237.28 ± 102.97
Diamonds	533.30 ± 6.30	Facebook_Comment_Volume	23.16 ± 0.20
Food_Delivery_Time	7.55 ± 0.03	Goodreads-Computer-Books	0.43 ± 0.00
IEEE80211aa-GATS	0.02 ± 0.00	Job_Profitability	13.14 ± 0.02
bike_sharing_demand	68.41 ± 0.60	Laptop_Prices_Dataset	439.87 ± 3.10
Wave_Energy_Perth_100	15507.90 ± 104.31	Wave_Energy_Sydney_100	14737.67 ± 150.43
Wave_Energy_Sydney_49	4567.97 ± 64.02	MIP-2016-regression	20966.10 ± 454.90
MiamiHousing2016	83101.09 ± 507.30	Mobile_Phone_Market	714.87 ± 11.15
Moneyball	19.42 ± 0.08	NASA_PHM2008	40.24 ± 0.06
NHANES_age_prediction	15.47 ± 0.04	OnlineNewsPopularity	8606.54 ± 7.04
Parkinson_Sound_Record	14.58 ± 0.09	Parkinsons_Telemonitoring	0.60 ± 0.04
Physicochemical_r	3.45 ± 0.04	SAT11-HAND-runtime	1232.03 ± 58.01
Shop_Customer_Data	28.56 ± 0.01	Superconductivity	10.17 ± 0.07
Wine_Quality_red	0.65 ± 0.00	Wine_Quality_white	0.68 ± 0.00
airfoil_self_noise	1.16 ± 0.02	analcata_supreme	0.09 ± 0.00
archive2	342.64 ± 3.20	archive_r56_Portuguese	2.86 ± 0.02
auction_verification	1145.54 ± 146.94	avocado_sales	0.09 ± 0.00
bank32nh	0.08 ± 0.00	bank8FM	0.03 ± 0.00
boston	4.25 ± 0.19	chscase_foot	0.95 ± 0.00
colleges	0.14 ± 0.00	combined_cycle_power_plant	3.22 ± 0.05
communities_and_crime	0.13 ± 0.00	concrete_compressive_strength	4.63 ± 0.07
cpu_act	2.65 ± 0.03	cpu_small	3.06 ± 0.02
dataset_sales	4.04 ± 0.02	debutanizer	0.04 ± 0.00
delta_elevators	0.0014 ± 0.00	elevators	0.0019 ± 0.00
fifa	0.78 ± 0.00	fried	1.01 ± 0.00
garments_worker_productivity	0.13 ± 0.00	gas_turbine_emission	0.44 ± 0.00

healthcare_insurance_expenses	4716.87 $\pm$ 36.52	house_16H_reg	29631.75 $\pm$ 251.56
house_8L	28617.41 $\pm$ 202.41	house_prices_nominal	30676.02 $\pm$ 2455.48
house_sales_reduced	132655.03 $\pm$ 1847.33	houses	42559.98 $\pm$ 928.78
housing_price_prediction	1009361.62 $\pm$ 8758.05	kin8nm	0.08 $\pm$ 0.00
mauna-loa-atmospheric-co2	0.39 $\pm$ 0.01	mv	0.02 $\pm$ 0.00
pol_reg	3.84 $\pm$ 0.10	pole	3.21 $\pm$ 0.14
puma32H	0.01 $\pm$ 0.00	puma8NH	3.24 $\pm$ 0.00
qsar_aquatic_toxicity	1.05 $\pm$ 0.01	qsar_fish_toxicity	0.86 $\pm$ 0.01
satellite_image	0.65 $\pm$ 0.00	sensory	0.77 $\pm$ 0.01
socmob	19.53 $\pm$ 0.64	space_ga	0.09 $\pm$ 0.00
steel_industry_energy	0.37 $\pm$ 0.03	stock	0.65 $\pm$ 0.01
stock_fardamento02	17.57 $\pm$ 0.08	sulfur	0.03 $\pm$ 0.00
topo_2_1	0.03 $\pm$ 0.00	treasury	0.23 $\pm$ 0.00
us_crime	0.14 $\pm$ 0.00	volume	52.09 $\pm$ 0.34
weather_izmir	1.09 $\pm$ 0.01	wind	2.83 $\pm$ 0.00
wine+quality	0.72 $\pm$ 0.00	yprop_4_1	0.03 $\pm$ 0.00

Table 6: Performance of various models on 18 high-dimensional datasets. The results show the mean accuracy of different models, including ModernNCA (MNCA), MLP, KNN, RealMLP, XGBoost (XGB), Random Forest (RForest), Logistic Regression (LogReg), TabPFN v2 (PFN-v2), TabPFN v2 with PCA (v2-pca), TabPFN v2 with subsampling (v2\*), ProtoGate (ProtoG), and CatBoost (CatB). The performance is evaluated on high-dimensional datasets, with the values representing mean accuracy for each model.

Dataset	MNCA	MLP	KNN	RealMLP	XGB	RForest	LogReg	PFN-v2	v2-pca	v2*	ProtoG	CatB
CLL_SUB_111	62.90	72.46	57.39	70.43	73.04	70.14	73.91	70.14	57.68	71.59	65.51	71.59
BASEHOCK	96.31	97.01	71.88	97.46	95.29	96.73	96.99	69.09	97.41	97.36	96.32	95.87
Prostate_GE	81.27	86.98	80.00	87.94	89.52	87.94	91.43	95.24	88.57	94.29	84.13	94.92
PCMAC	88.21	88.53	66.48	90.15	91.64	92.20	87.15	92.70	90.76	90.14	88.21	92.01
GLI_85	81.57	85.49	76.47	89.80	82.35	83.92	90.59	80.39	86.27	92.55	81.96	80.78
RELATHE	88.18	90.54	75.03	90.23	87.11	87.30	90.49	86.36	87.65	89.95	89.92	90.35
SMK_CAN_187	63.51	66.84	69.47	69.82	66.49	70.70	72.11	71.05	71.75	72.10	70.71	71.40
warpPIE10P	98.41	99.05	92.38	100.0	94.92	98.57	100.0	100.0	100.0	100.0	97.79	98.89
leukemia	90.22	95.11	86.67	94.67	97.78	92.00	96.00	92.44	93.33	96.00	94.00	94.22
orlraws10P	97.67	98.33	92.00	99.00	84.33	99.00	99.00	92.00	99.33	99.67	92.67	99.00
GLIOMA	58.00	60.67	68.00	67.33	66.67	64.00	64.00	62.67	69.33	68.67	69.91	66.67
warpAR10P	83.08	85.64	53.08	97.44	81.28	87.18	97.69	90.77	95.38	96.67	90.04	87.44
TOX_171	76.00	88.19	70.86	90.48	78.10	78.67	90.29	80.95	82.48	87.24	85.52	83.05
lung	91.54	95.45	93.66	95.28	93.66	92.68	95.12	95.28	93.50	95.61	95.43	93.01
ALLAML	87.56	95.56	81.33	96.89	96.00	96.44	92.00	92.89	93.78	94.67	91.14	94.67
colon	78.46	78.97	76.92	83.08	74.87	82.56	86.15	81.54	78.46	79.49	78.46	77.95
gisette	97.21	97.57	95.04	97.86	97.55	96.82	97.51	97.35	97.26	97.23	97.18	97.78
arcene	81.67	85.50	84.50	81.00	75.00	86.83	88.00	83.67	88.33	92.00	85.33	85.00
Mean	82.86	87.11	77.29	88.83	84.76	86.87	89.36	85.25	87.29	89.73	86.37	87.48

Table 7: Performance of various models on 12 multi-class classification tasks with more than 10 classes. The results show the mean accuracy of different models, including KNN, PFN-v2\*, PFN-v2-ECOC, XGB (XGBoost), CatBoost (CatB), Random Forest (RForest), ModernNCA (MNCA), Multi-layer Perceptron (MLP), Logistic Regression (LogReg), and RealMLP. The performance is evaluated on 12 multi-class datasets with more than 10 classes, with accuracy values presented for each model on the respective datasets.

Dataset	KNN	PFN-v2*	PFN-v2-DPT	PFN-v2-ECOC	XGB	CatB	RForest	MNCA	MLP	LogReg	RealMLP
100-plants-texture	79.69	90.94	82.67	84.92	77.06	89.73	82.65	80.52	83.92	86.88	88.35
100-plants-margin	77.50	88.56	81.94	79.40	74.25	84.06	82.79	77.60	80.44	79.69	83.58
100-plants-shape	60.31	79.52	72.15	63.38	56.15	65.19	64.33	70.10	47.33	65.94	72.08
UJI_Pen_Characters	36.26	45.71	33.38	44.20	30.35	38.88	34.24	44.03	37.75	19.41	46.37
texture	98.45	100.0	99.98	100.0	98.55	99.13	96.76	99.68	99.40	99.64	99.95
letter	94.90	97.57	96.69	97.78	96.26	96.75	91.56	97.96	96.40	75.80	98.31
walking-activity	60.29	61.22	57.28	61.92	65.06	64.92	61.74	64.85	60.64	27.02	65.13
helena	28.94	33.31	28.54	19.20	32.42	37.90	33.91	36.58	37.91	33.40	38.55
internet_usage	30.17	54.34	50.51	50.86	51.08	37.90	33.91	52.09	43.00	37.73	52.23
kropt	71.22	77.96	71.44	77.11	86.95	79.26	71.77	78.27	64.45	28.08	92.03
kr-vs-k	70.78	78.46	71.54	76.29	87.26	74.81	71.60	76.83	65.03	28.03	91.85
ASP-POTASSCO	34.75	43.50	41.88	45.27	42.24	41.08	42.86	37.45	29.63	35.14	41.70
Mean	61.94	70.93	65.67	66.69	66.47	67.47	64.01	68.00	62.16	51.40	72.51

Table 8: Performance of various models on 18 large-scale datasets. The results show the mean accuracy/RMSE of different models, including MLP, Logistic Regression/Linear Regression (LR), KNN, XGBoost (XGB), Random Forest (RForest), CatBoost (CatB), ModernNCA (MNCA), RealMLP, and various versions of TabPFN v2: original TabPFN v2 (PFNV2), TabPFN v2 with K-means (PFNV2-K), TabPFN v2 with Bagging (PFNV2-B), PNFv2\* (TabPFNV2\*), PNFv2-DT (TabPFN-DT), and PNFv2-DF (TabPFNV2-DF).

Dataset	MLP	LR	KNN	XGB	RForest	CatB	MNCA	RealMLP	PFNV2	PFNV2-K	PFNV2-B	PFNV2*	PFNV2-DT	PFNV2-DF
BNG(credit-a)	90.07	85.98	87.41	90.21	89.25	91.13	89.98	90.91	89.55	89.01	89.66	89.89	90.45	90.43
CDC_Indicators	86.79	86.55	86.39	86.76	86.60	86.78	86.76	86.76	86.65	86.68	86.69	86.74	86.75	86.70
Higgs	75.53	64.29	65.16	73.33	71.87	74.81	73.28	75.36	71.64	71.56	72.01	72.13	73.53	73.62
Smoking_signal	73.90	72.53	72.36	73.87	73.08	73.99	73.63	74.00	73.47	73.37	73.55	73.69	73.74	73.84
nomao	96.19	94.59	95.20	96.92	96.07	97.03	96.68	96.37	96.08	96.29	96.12	96.18	96.75	96.34
sf-police-incidents	87.84	87.84	85.87	87.68	87.84	87.87	-	87.84	87.84	87.84	87.84	87.84	87.84	87.84
Data_Crowdfunding	96.48	67.04	93.70	96.89	95.29	96.81	96.53	96.71	94.59	91.81	94.96	95.07	96.90	96.83
Fashion-MNIST	89.54	85.69	86.00	90.03	86.57	90.24	89.36	90.25	68.40	82.82	83.89	86.26	78.91	78.82
covertype	94.01	72.54	92.76	96.30	78.30	90.77	97.31	97.38	83.54	82.95	84.16	86.85	97.38	97.44
jannis	71.99	64.60	65.67	71.83	69.19	72.26	72.57	73.00	70.24	70.26	70.59	71.31	72.57	72.50
poker-hand	99.99	50.12	54.01	99.51	64.63	97.69	76.31	99.88	41.97	38.86	36.80	54.12	91.13	92.33
vollkert	69.85	58.75	67.41	69.74	62.71	70.88	77.18	73.76	62.82	62.15	62.81	64.84	68.66	67.76
Airlines_DepDelay ( $\times 10^4$ )	2.905	2.933	3.170	2.891	2.907	2.881	-	2.482	2.937	2.933	2.937	2.915	2.900	2.897
Wave_Energy_Farm ( $\times 10^3$ )	8.199	13.19	32.29	6.917	7.294	7.173	6.148	59.05	7.214	8.375	7.063	10.506	6.616	6.785
UJIIndoorLoc ( $\times 10^0$ )	9.958	$\infty$	9.004	10.47	23.19	9.139	5.990	65.34	66.49	7.825	7.435	9.538	14.404	7.472
blogfeedback ( $\times 10^4$ )	2.387	$\infty$	2.410	2.093	2.026	2.044	1.953	2.105	3.073	2.687	2.700	2.014	1.914	1.944
microsoft ( $\times 10^{-1}$ )	7.577	7.782	8.284	7.514	7.566	7.453	7.573	5.077	7.735	7.981	7.720	7.612	7.944	7.728
yahoo ( $\times 10^{-1}$ )	7.692	7.997	8.504	7.629	-	7.514	-	5.671	8.148	8.332	8.132	7.961	16.409	8.069

## G Limitations

While this paper does not introduce a new model architecture or training paradigm, it offers a timely and principled analysis of TabPFN v2, a powerful tabular foundation model. Our contributions lie in empirically evaluating its strengths, identifying its limitations, and proposing practical extensions that enhance its applicability—particularly to large-scale, high-dimensional, and multi-class settings. A key limitation of our work is that the proposed extensions are primarily post-hoc and do not fully address the scalability constraints inherent to the original architecture. Nevertheless, by improving model usability without retraining, we reduce the computational and environmental costs typically associated with large model development.

To the best of our knowledge, this work poses no explicit ethical concerns. It provides practical guidance for applying pre-trained tabular models in real-world domains such as healthcare and finance, where transparency and efficiency are essential. Our study highlights the value of understanding and extending foundation models alongside architectural innovation.

Table 9: Performance of TabPFN v2 and the extracted feature embeddings across 29 classification datasets. The table shows the average classification accuracy for each dataset when using different layers (Layer 6, Layer 9, Layer 12) of the transformer as feature embeddings, as well as the “combined” approach, where embeddings from up to three selected layers are concatenated. The “selected layers” column indicates the specific layers chosen for each dataset based on validation set performance. “Vanilla” refers to the embeddings extracted using the vanilla strategy, which utilizes only the 12th layer of the transformer. “S” and “P” refer to *unsupervised* embedding extraction approaches by appending a column of dummy labels with zero values and permuting each column as labels, respectively, as described in Appendix D.2.

	PFN-v2	Vanilla	S	P	layer-6	layer-9	layer-12	combined	selected layers
FOREX_audchf-day-High	77.38	50.68	56.95	69.48	68.39	73.57	74.11	77.11	(5, 9, 11)
taiwanese_bankruptcy_prediction	96.99	56.45	96.77	95.75	97.14	96.77	97.07	97.14	(6)
rl	85.51	50.00	60.56	70.82	66.90	69.52	86.72	87.53	(11, 12)
pc3	89.46	10.22	89.78	86.90	90.10	88.82	88.82	88.82	(8)
eye_movements_bin	61.83	50.00	55.12	57.95	59.72	59.40	62.16	62.16	(6, 9, 12)
BNG(breast-w)	98.43	69.51	97.60	98.51	98.34	98.46	98.67	98.51	(6, 9)
FOREX_cadjpy-hour-High	69.53	51.79	66.55	71.12	62.12	64.87	70.66	70.88	(4, 5, 6)
dis	99.34	85.43	98.41	98.54	98.41	98.28	99.34	99.47	(4, 5, 6)
sylvine	97.46	85.66	72.78	95.71	92.49	93.95	97.27	96.49	(1, 11)
BNG(tic-tac-toe)	78.04	34.71	71.41	73.79	73.96	73.71	78.75	79.03	(5, 10, 12)
online_shoppers	90.59	84.51	85.93	89.46	90.02	90.11	90.63	90.02	(8)
Cardiovascular-Disease-dataset	72.84	50.86	68.73	72.60	72.96	73.06	73.14	73.09	(5, 8, 12)
credit	78.04	62.31	75.86	78.31	77.62	77.80	77.95	77.59	(4, 6, 9)
FOREX_audsgd-hour-High	67.26	51.48	65.49	70.14	57.24	61.06	69.62	70.41	(7, 10, 12)
waveform-5000	86.00	80.60	55.70	87.10	85.60	85.60	86.40	86.90	(1, 6, 11)
jungle_chess	85.65	39.60	64.12	72.14	78.55	80.44	86.66	86.85	(10, 11, 12)
BNG(cmc)	57.40	42.62	52.48	55.16	56.19	56.72	57.72	57.88	(9, 10, 12)
page-blocks	97.35	94.25	95.43	96.35	96.07	96.71	97.17	97.35	(6, 7, 12)
segment	93.07	72.29	69.26	87.23	91.99	88.10	93.51	92.64	(1, 12)
website_phishing	90.77	36.90	82.66	90.04	85.98	87.08	91.88	91.88	(7, 10)
baseball	93.66	78.73	92.54	92.16	93.28	94.03	93.66	95.15	(10, 11)
pendigits	99.50	59.75	72.40	98.18	92.81	93.04	99.41	99.45	(3, 4, 12)
Gender_Gap_in_Spanish_WP	60.84	33.68	59.47	60.84	59.68	60.32	60.53	60.84	(2, 12)
wine-quality-white	62.35	10.51	49.29	55.10	54.08	55.31	63.57	64.39	(8, 11, 12)
satimage	91.21	82.04	84.99	89.19	88.72	88.65	91.91	91.91	(8, 11, 12)
mfeat-fourier	90.00	55.50	46.75	85.75	77.75	82.25	89.50	89.50	(2, 7, 12)
VulNoneVul	98.95	1.05	98.95	98.33	98.95	98.95	98.95	98.95	(1)
law-school-admission-bianry	100.0	99.83	79.76	98.82	100.0	100.0	100.0	100.0	(6)
KDD	80.34	78.45	62.36	76.76	79.34	78.35	81.23	79.94	(1, 8, 10)

## References

- [1] Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M Pujol. Data mining methods for recommender systems. In *Recommender systems handbook*, pages 39–71. Springer, 2010.
- [2] Michael Arbel, David Salinas, and Frank Hutter. Equitabpfn: A target-permutation equivariant prior fitted networks. *CoRR*, abs/2502.06684, 2025.
- [3] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, pages 6679–6687, 2021.
- [4] Sarkhan Badirli, Xuanqing Liu, Zhengming Xing, Avradeep Bhowmik, and Sathiya S. Keerthi. Gradient boosting neural networks: Grownnet. *CoRR*, abs/2002.07971, 2020.
- [5] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions Neural Networks and Learning Systems*, 35(6):7499–7519, 2024.
- [6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [7] Jintai Chen, KuanLun Liao, Yanwen Fang, Danny Chen, and Jian Wu. Tabcaps: A capsule neural network for tabular data classification with bow routing. In *ICLR*, 2023.
- [8] Jintai Chen, Kuanlun Liao, Yao Wan, Danny Z. Chen, and Jian Wu. Danets: Deep abstract networks for tabular data classification and regression. In *AAAI*, pages 3930–3938, 2022.
- [9] Jintai Chen, Jiahuan Yan, Qiyuan Chen, Danny Ziyi Chen, Jian Wu, and Jimeng Sun. Can a deep learning model be a sure bet for tabular prediction? In *KDD*, pages 288–296, 2024.
- [10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.
- [11] Benjamin Feuer, Chinmay Hegde, and Niv Cohen. Scaling tabpfn: Sketching and feature selection for tabular prior-data fitted networks. *CoRR*, abs/2311.10609, 2023.
- [12] Benjamin Feuer, Robin Tibor Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. Tunetables: Context optimization for scalable prior-data fitted networks. In *NeurIPS*, pages 83430–83464, 2024.
- [13] Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *ICLR*, 2025.
- [14] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. In *NeurIPS*, pages 24991–25004, 2022.
- [15] Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Tabular deep learning meets nearest neighbors in 2023. In *ICLR*, 2024.
- [16] Yury Gorishniy, Ivan Rubachev, Valentin Khurlov, and Artem Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, pages 18932–18943, 2021.
- [17] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS*, pages 507–520, 2022.
- [18] Noah Hollmann, Samuel Müller, Katharina Eggenberger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *ICLR*, 2023.
- [19] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.
- [20] David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. In *NeurIPS*, pages 26577–26658, 2024.

- [21] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. The tabular foundation model tabpfm outperforms specialized time series forecasting models based on simple features. *CoRR*, abs/2501.02945, 2025.
- [22] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *CoRR*, abs/2012.06678, 2020.
- [23] Stephanie L Hyland, Martin Faltys, Matthias Hüser, Xinrui Lyu, Thomas Gumbsch, Cristóbal Esteban, Christian Bock, Max Horn, Michael Moor, Bastian Rieck, et al. Early prediction of circulatory failure in the intensive care unit using machine learning. *Nature medicine*, 26(3):364–373, 2020.
- [24] Alan Jeffares, Tennison Liu, Jonathan Crabbé, Fergus Imrie, and Mihaela van der Schaar. Tangos: Regularizing tabular neural networks through gradient orthogonalization and specialization. In *ICLR*, 2023.
- [25] Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In *NeurIPS*, pages 23928–23941, 2021.
- [26] 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 *NIPS*, pages 3146–3154, 2017.
- [27] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, pages 971–980, 2017.
- [28] Boris Kovalerchuk and Evgenii Vityaev. *Data mining in finance: advances in relational and hybrid methods*. Springer Science & Business Media, 2005.
- [29] Si-Yang Liu and Han-Jia Ye. Tabpfm unleashed: A scalable and effective solution to tabular classification problems. *CoRR*, abs/2502.02527, 2025.
- [30] Si-Yang Liu and Han-Jia Ye. Tabpfm unleashed: A scalable and effective solution to tabular classification problems. In *ICML*, 2025.
- [31] Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony L. Caterini. Tabpfgen - tabular data generation with tabpfm. *CoRR*, abs/2406.05216, 2024.
- [32] Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Hamidreza Kamkari, Alex Labach, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L. Caterini. Tabdpt: Scaling tabular foundation models. *CoRR*, abs/2410.18164, 2024.
- [33] Duncan C. 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? In *NeurIPS*, pages 76336–76369, 2023.
- [34] Youssef Nader, Leon Sixt, and Tim Landgraf. DNNR: differential nearest neighbors regression. In *ICML*, pages 16296–16317, 2022.
- [35] Thomas Nagler. Statistical foundations of prior-data fitted networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *ICML*, pages 25660–25676, 2023.
- [36] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *ICLR*, 2020.
- [37] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *NeurIPS*, pages 6639–6649, 2018.
- [38] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. In *ICML*, 2025.
- [39] Cristóbal Romero and Sebastián Ventura. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(6):601–618, 2010.



- 334 [40] Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. Revisiting pretraining  
335 objectives for tabular deep learning. *CoRR*, abs/2207.03208, 2022.
- 336 [41] Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. Tabred: A benchmark  
337 of tabular machine learning in-the-wild. In *ICLR*, 2025.
- 338 [42] Sergio Ruiz-Villafranca, José Roldán Gómez, Juan Manuel Castelo Gómez, Javier Carrillo  
339 Mondéjar, and José Luis Martínez. A tabpfn-based intrusion detection system for the industrial  
340 internet of things. *The Journal of Supercomputing*, 80(14):20080–20117, 2024.
- 341 [43] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian  
342 Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In  
343 *CIKM*, pages 1161–1170, 2019.
- 344 [44] Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims  
345 Volkovs, and Anthony L. Caterini. Retrieval & fine-tuning for in-context tabular models. In  
346 *NeurIPS*, pages 108439–108467, 2024.
- 347 [45] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong,  
348 and Ed H. Chi. DCN V2: improved deep & cross network and practical lessons for web-scale  
349 learning to rank systems. In *WWW*, pages 1785–1797, 2021.
- 350 [46] Jing Wu, Suiyao Chen, Qi Zhao, Renat Sergazinov, Chen Li, Shengjie Liu, Chongchao Zhao,  
351 Tianpei Xie, Hanqing Guo, Cheng Ji, Daniel Cociorva, and Hakan Brunzell. Switchtab:  
352 Switched autoencoders are effective tabular learners. In *AAAI*, pages 15924–15933, 2024.
- 353 [47] Derek Xu, Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompts  
354 for tabular pfns. *CoRR*, abs/2405.16156, 2024.
- 355 [48] Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at  
356 deep learning on tabular data. *CoRR*, abs/2407.00956, 2024.
- 357 [49] Han-Jia Ye, Huai-Hong Yin, De-Chuan Zhan, and Wei-Lun Chao. Revisiting nearest neighbor  
358 for tabular data: A deep tabular baseline two decades later. In *ICLR*, 2025.
- 359 [50] Hangting Ye, Wei Fan, Xiaozhuang Song, Shun Zheng, He Zhao, Dan dan Guo, and Yi Chang.  
360 Ptarl: Prototype-based tabular representation learning via space calibration. In *ICLR*, 2024.