Mitigating Label Shift in Tabular In-Context Learning via Test-Time Posterior Adjustment

Seunghan Lee (LG AI Research)

LG AI Research

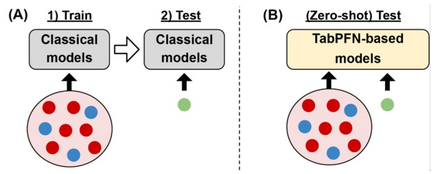
ICML International Conference On Machine Learning

1. Tabular In-Context Learning (feat. TabPFN)

- Predicts the label of a test instance by conditioning on the **entire training dataset**
- Posterior distribution of TabPFN:

$$\widehat{p}_{\text{TabPFN}}(y \mid x_j, \mathcal{D}_{\text{train}}) = \frac{\exp\left(f(x_j, \mathcal{D}_{\text{train}})[y]\right)}{\sum_{c=1}^{C} \exp\left(f(x_j, \mathcal{D}_{\text{train}})[c]\right)}$$

- Classical classification models (A) vs. TabPFN-based models (B)
  - (A): Refer to training dataset "***implicitly***"
    → via Model parameters
  - (B): Refer to training dataset "***explicitly***"
    → via Attention mechanism



2. Majority-class Bias of TabPFN-based Methods

- **Majority-class bias**: Predicts toward the **majority class in the training dataset**
- Particularly harmful when the train and test label distns differ (i.e., label shift)

Table 1. **Confusion matrices.** TabPFN exhibits severe *majority-class bias*, predicting **98.3%** of samples as the **majority class**, while DistPFN mitigates this via a simple test-time adjustment.

TabPFN-v2 (Nature 2025)			+ DistPFN (Ours)		
	$\hat{y}=0$	$\hat{y}=1$	(%)	$\hat{y}=0$	$\hat{y}=1$
$y=0$	**87.2%**	0.0%	$y=0$	81.1%	6.1%
$y=1$	**11.1%**	1.7%	$y=1$	3.0%	9.8%
Total	**98.3%**	1.7%	Total	84.1%	15.9%

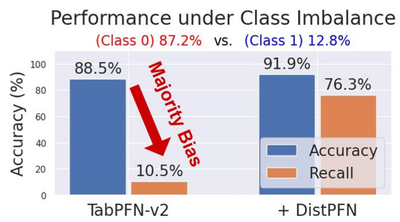


Figure 1. **Majority-class bias.** TabPFN suffers from majority-class bias, resulting in poor recall for the minority class.

3. Prior works

- Limitations of prior label-shift methods:
  - (1) Require additional training
  - (2) Depend on estimating the test label distribution

Table 3. **vs. Label shift methods.** DistPFN operates in a post-hoc manner without test prior estimation or model retraining.

	EME	Logit adj.	Bal. softmax	**DistPFN**
When applied	Inference	Training	Training	**Inference**
Test prior estimation	✓	✗	✗	✗
Model retraining	✗	✓	✓	✗

4. Proposed Method: DistPFN

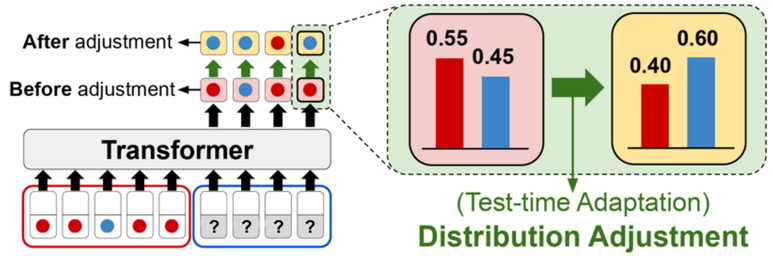
*How to handle the **majority-class bias** in **TabPFN-based models** effectively, **w/o retraining** the model?*

DistPFN Mitigates this bias via a simple **test-time adaptation** method that **rescales the predicted class probabilities** for each test instance **using its own predicted results**

$$\widetilde{p}_{\text{DistPFN}}(y) = \text{Norm}\Big(\widehat{p}_{\text{TabPFN}}(y) \cdot \underbrace{\frac{\widehat{p}_{\text{TabPFN}}(y)}{p_{\text{train}}(y)}}_{\text{Adjustment factor }(\alpha)}\Big) = \text{Norm}\Big(\frac{\widehat{p}_{\text{TabPFN}}(y)^2}{p_{\text{train}}(y)}\Big),$$

Amplifies the impact of the **observed test samples**

Downweights the influence of the **training distribution**



DistPFN-T Optimal strength of adjustment may vary depending on the deviation of test-time predictions from the training prior
→ Introduces **temperature scaling** to control the **sharpness of adjustment**, based on the **discrepancy** between the **training prior & predicted distribution**

Algorithm 1 Pseudocode for DistPFN

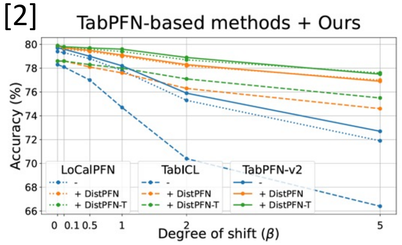
```
# x_test: test instance(s)
# D_train: training dataset
# p_train: training class prior
# f: TabPFN-based model
# alpha: adjustment factor

logits = f(x_test, D_train)
p_hat = softmax(logits, dim=0)

if method == "tabpfn":
    alpha = 1

elif method == "distpfn":
    alpha = p_hat / p_train

elif method == "distpfn-t":
    tau = cross_entropy(p_hat, p_train)
    p_hat_scaled = softmax(p_hat / tau, dim=0)
    alpha = p_hat_scaled / p_train

p_hat = normalize(alpha * p_hat)
```

5. Experiments

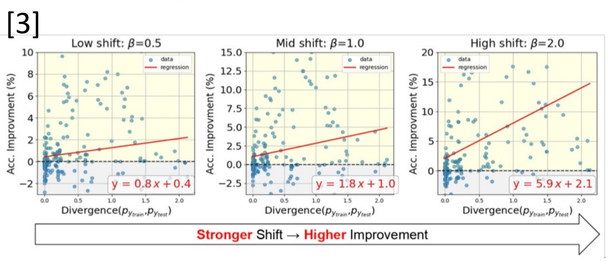
[1]

	Methods	w/o shift	Shift strength (β) 0.0	0.1	0.5	1.0	2.0	5.0	Avg.
Machine Learning	LogReg.	0.765±0.002	0.719±0.002	0.709±0.002	0.674±0.004	0.634±0.004	0.597±0.002	0.566±0.004	0.650
	+ HPO	0.771±0.003	0.697±0.005	0.687±0.003	0.653±0.003	0.616±0.006	0.586±0.003	0.550±0.003	0.631
	SVM	0.780±0.003	0.684±0.002	0.646±0.004	0.560±0.005	0.531±0.003	0.486±0.004	0.448±0.004	0.559
	+ HPO	0.784±0.003	0.731±0.001	0.689±0.008	0.626±0.003	0.597±0.005	0.570±0.005	0.541±0.007	0.626
	MLP	0.778±0.004	0.658±0.005	0.647±0.004	0.613±0.005	0.574±0.006	0.527±0.004	0.493±0.001	0.585
	+ HPO	0.795±0.005	0.706±0.006	0.688±0.006	0.654±0.008	0.615±0.006	0.580±0.005	0.545±0.005	0.631
	kNN	0.765±0.004	0.663±0.004	0.657±0.004	0.629±0.003	0.589±0.003	0.538±0.003	0.501±0.004	0.596
	+ HPO	0.783±0.002	0.693±0.002	0.684±0.003	0.644±0.004	0.588±0.003	0.540±0.003	0.498±0.002	0.608
	Random Forest	0.796±0.003	0.768±0.003	0.765±0.003	0.748±0.005	0.718±0.004	0.665±0.005	0.618±0.005	0.714
	+ HPO	0.803±0.002	0.771±0.002	0.767±0.001	0.743±0.004	0.701±0.004	0.627±0.008	0.578±0.006	0.698
	LightGBM	0.789±0.003	0.758±0.004	0.753±0.002	0.734±0.003	0.705±0.004	0.657±0.005	0.618±0.005	0.704
	+ HPO	0.790±0.006	0.726±0.008	0.661±0.005	0.655±0.008	0.608±0.008	0.577±0.015	0.551±0.004	0.630
	CatBoost	0.803±0.001	0.774±0.002	0.771±0.002	0.751±0.004	0.718±0.004	0.665±0.005	0.621±0.005	0.717
	+ HPO	0.802±0.002	0.774±0.002	0.771±0.002	0.752±0.004	0.719±0.004	0.665±0.006	0.621±0.005	0.717
Deep Learning (Non-FMs)	FT-Transformer	0.784±0.002	0.748±0.004	0.746±0.004	0.718±0.005	0.674±0.005	0.610±0.003	0.551±0.007	0.675
	TabM	0.794±0.002	0.762±0.004	0.757±0.004	0.735±0.003	0.694±0.005	0.624±0.006	0.565±0.006	0.690
	TabulaRNN	0.749±0.003	0.699±0.003	0.684±0.003	0.641±0.004	0.585±0.009	0.522±0.011	0.465±0.008	0.599
	MambaTab	0.719±0.004	0.629±0.006	0.603±0.004	0.525±0.002	0.466±0.010	0.430±0.005	0.394±0.002	0.508
	RealMLP	0.794±0.002	0.760±0.004	0.758±0.005	0.745±0.003	0.720±0.005	0.677±0.002	0.643±0.004	0.717
Deep Learning (FMs)	LoCalPFN	**0.816**±0.002	0.794±0.003	0.793±0.004	0.788±0.003	0.778±0.002	0.753±0.004	0.719±0.000	0.771
	+ DistPFN	**0.816**±0.002	0.797±0.001	0.796±0.002	0.794±0.002	0.790±0.002	0.782±0.001	0.770±0.003	0.788
	+ DistPFN-T	**0.816**±0.002	**0.798**±0.002	**0.797**±0.002	**0.796**±0.002	**0.794**±0.002	**0.787**±0.001	**0.776**±0.003	**0.791**
	TabICL	**0.806**±0.002	0.783±0.003	0.781±0.003	0.770±0.003	0.747±0.003	0.704±0.006	0.664±0.006	0.742
	+ DistPFN	**0.806**±0.002	0.786±0.002	0.786±0.002	0.781±0.002	0.776±0.002	0.763±0.002	0.746±0.004	0.773
	+ DistPFN-T	**0.806**±0.003	**0.786**±0.003	**0.786**±0.003	**0.783**±0.002	**0.780**±0.002	**0.771**±0.001	**0.755**±0.004	**0.777**
	TabPFN-v2	**0.818**±0.004	0.797±0.003	0.796±0.004	0.790±0.002	0.782±0.002	0.759±0.003	0.727±0.003	0.775
	+ DistPFN	**0.818**±0.002	**0.799**±0.001	0.797±0.002	0.795±0.002	0.791±0.003	0.783±0.003	0.769±0.003	0.789
	+ DistPFN-T	**0.818**±0.002	**0.799**±0.003	**0.798**±0.002	**0.797**±0.002	**0.796**±0.003	**0.789**±0.003	**0.775**±0.003	**0.792**

[2]



[3]



[1] Comparison w/ other tabular models

[2] Avg. Accuracy across degree of shifts

[3] (Per-dataset) Stronger shift yields higher gain

[4] Efficiency: Avg. pred time across 253 datasets

[4]

	Pred. time	Avg. Acc.
LoCalPFN	0.618	0.771
+ DistPFN	0.619	0.788
+ DistPFN-T	0.619	**0.791**
TabICL	0.620	0.742
+ DistPFN	0.622	0.773
+ DistPFN-T	0.622	**0.777**
TabPFN-v2	1.002	0.775
+ DistPFN	1.003	0.789
+ DistPFN-T	1.003	**0.792**