Drift-Resilient TabPFN: In-Context Learning Temporal Distribution Shifts on Tabular Data

Kai Helli[*,1,2] **David Schnurr**[*,1,3] **Noah Hollmann**[1,4] **Samuel Müller**[1] **Frank Hutter**[1]

[1] University of Freiburg, [2] Technical University of Munich, [3] ETH Zurich,
[4] Charité University Medicine Berlin, [*] Equal contribution.
Correspondence to `kai.helli@tum.de`

Abstract While most ML models expect independent and identically distributed data, this assumption is often violated in real-world scenarios due to distribution shifts, resulting in the degradation of machine learning model performance. Until now, no method has significantly outperformed classical supervised learning, which ignores these shifts, with even smaller gains for tabular data.

To address this, we present Drift-Resilient TabPFN, a fresh approach based on In-Context Learning with a Prior-Data Fitted Network that learns the learning algorithm itself: it accepts the entire training dataset as input and makes predictions on the test set in a single forward pass. Specifically, it learns to approximate Bayesian inference on synthetic datasets drawn from a prior that specifies the model's inductive bias. This prior is based on structural causal models (SCM), which gradually shift over time. To model shifts of these causal models, we use a secondary SCM, that specifies changes in the primary model parameters.

The resulting Drift-Resilient TabPFN can be applied to unseen datasets, runs in seconds, and needs no hyperparameter tuning. Comprehensive evaluations across 18 synthetic and real-world datasets demonstrate large performance improvements over a wide range of baselines, such as XGB, CatBoost, and TabPFN. Compared to the strongest baselines, it improves accuracy from 0.688 to 0.744 and ROC AUC from 0.786 to 0.832 while maintaining stronger calibration. This approach could serve as significant groundwork for further research on out-of-distribution prediction.

1 Introduction

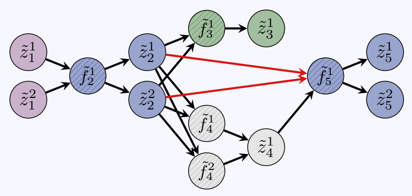
In traditional machine learning the train and test data are assumed to be sampled from the same distribution [47]. However, this assumption of independent and identically distributed (i.i.d.) data is commonly violated in real-world scenarios due to distribution shifts, resulting in performance degradation of standard ML models over time [47, 59]. Research in the area of temporal domain generalization (Temporal DG) tries to address these shifts by developing methods that perform consistently across temporal domains and generalize beyond the training regimen, i.e. into the future. In fields such as healthcare, climate science, or finance, data is most often organized in a tabular format [10, 58]. Here shifts are driven by hidden variables such as policy or climate changes, equipment updates, seasonal changes, or activity cycles, limiting model deployment [59].

Young and Steele [63] describe declining mortality quantification in a hospital system while Pasterkamp et al. [45] show deterioration of cardiovascular risk models over time, leading to increased mortality; Ganesan et al. [21] show the COVID-19 pandemic exacerbated this issue, as ICU mortality prediction models failed to adapt to the unique characteristics of COVID-19 patients; environmental models need to continually adapt to climate changes [8]; fraud detection models need to continuously adapt as the strategies of fraudsters adapt to the models themselves [37]; these feedback loops often arise in practice, where model deployment inherently causes a system change

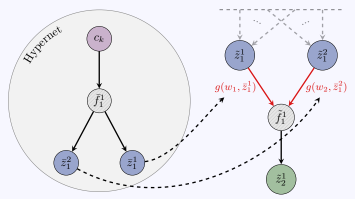
High-level overview

(a) We generate synthetic datasets by sampling Structural Causal Models (SCMs) whose edges shift over time.

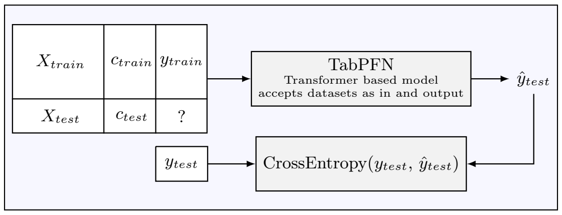
An SCM with sparse edge shifts indicated in red.



A secondary SCM, called "Hypernet" generates the strength of edge shifts depending on a time domain.



(b) TabPFN accepts entire datasets as inputs. Given millions of datasets from (a), it learns to make predictions on held-out test samples, learning the prediction algorithm itself with an inductive bias for addressing distribution shifts.


X_{train}
c_{train}
y_{train}
X_{test}
c_{test}
?
TabPFN
Transformer based model accepts datasets as in and output
$\hat{y}_{test}$
y_{test}
CrossEntropy(y_{test}, $\hat{y}_{test}$)

(c) Trained once, TabPFN can be applied to novel real-world data and makes predictions in a single forward pass, automatically detecting and extrapolating distribution shifts.

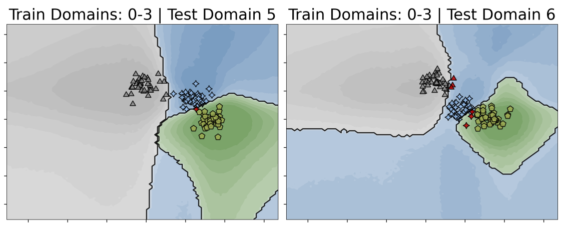


Figure 1: High-level overview of our method. We train a transformer that accepts entire datasets as input to learn the learning algorithm itself by training on millions of synthetic datasets once as part of algorithm development. The trained model can be applied to arbitrary real-world datasets. In (b), X, c, and y refer to features, time domain, and label respectively. In (c), we show predictions on test domains 4 (left) and 5 (right), where we see a distribution shift. Drift-Resilient TabPFN accurately updates decision boundaries in this example.

[2]. Robustness to such distribution shifts stands as a prominent challenge in current machine learning (ML) research [61]. So far multiple approaches have been proposed to address temporal distribution shifts using neural networks (NNs) [61, 5, 43]. However, modeling distribution shifts in tabular data presents a two-fold challenge: (i) NNs have struggled to model and extrapolate distribution shifts to date [61, 23] (ii) approaches for modeling distribution shifts have mostly employed NNs, while tree-based methods have consistently outperformed NNs in handling tabular data [10, 24, 53, 25] - leaving a wide methodological gap in addressing this common real-world scenario.

We provide a fresh perspective on predicting given distribution shifts by leveraging in-context-learning (ICL) to learn the prediction algorithm itself - bypassing many challenges encountered in this setting. Building on the foundation of Prior-Data Fitted Networks (PFNs; 42) and specifically TabPFN (27; see Section 3) . PFNs leverage large-scale machine learning and ICL techniques to approximate Bayesian inference accurately for any prior that can be sampled from. They are trained on millions of synthetic datasets sampled from this prior. For each such dataset, a supervised learning task is constructed, and the model is asked to predict on held out test samples. Then, the PFN is able to apply the principles learned on this synthetic data to real-world datasets, effectively having learned a prediction algorithm.

This paper introduces *Drift-Resilient TabPFN*, an adaptation of the TabPFN framework tailored for tabular datasets exhibiting temporal distribution shifts. Our idea is as follows: Data distribution shifts can be modeled as gradual changes to the structural causal model (SCM; 46, 48) underlying the data. By including the assumption that underlying models change over time into the approximated prior, our models learn to estimate, adapt to, and extrapolate these model changes.

2 Background

2.1 Distribution Shift Settings

Consider $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{C}$ as the sample spaces for features, labels, and domain indices, respectively, with specific instances represented by $\boldsymbol{x}$, y, and $\boldsymbol{c}$. The corresponding random variables are X, Y, and C.

A dataset is a collection of n tuples, $\mathcal{D} := \{(\boldsymbol{x}_i, y_i, \hat{\boldsymbol{c}}_k)\}_{i=1}^{n}$, each drawn from a conditional distribution $\mathbb{P}(X, Y \mid C = \boldsymbol{c}_k)$ over $\mathcal{X} \times \mathcal{Y}$. Here, $\boldsymbol{c}_k \in \mathcal{C}$ serves as a domain index conditioning the sample distribution. Since the true temporal domain is often unknown in real-world data, it is approximated as $\hat{\boldsymbol{c}}_k$ in the dataset. To isolate samples from a specific domain $\hat{\boldsymbol{c}}_k$, we define the sub-dataset $\mathcal{D}_{\hat{\boldsymbol{c}}_k} := \{(\boldsymbol{x}, y, \hat{\boldsymbol{c}}) \in \mathcal{D} \mid \hat{\boldsymbol{c}} = \hat{\boldsymbol{c}}_k\}$. This allows for a more nuanced analysis of datasets with domain shifts.

Domain Generalization (DG) aims to train a model that generalizes from source domains $\mathcal{C}^{\text{train}}$ to target domains $\mathcal{C}^{\text{test}}$ without accessing target domain samples. The objective is to learn a mapping function $f : \mathcal{X} \times \mathcal{C} \to \mathcal{Y}$ that minimizes expected loss on unseen target domains. Temporal Domain Generalization (Temporal DG), a special case of DG, has a one-dimensional domain index set $\mathcal{C}$ that follows a total ordering, $c_1 \leq c_2 \leq \ldots$. The training set is limited to source domains that precede target domains, $\mathcal{C}^{\text{train}} = \{c_1, c_2, \ldots, c_t\}$, and the objective is to learn a predictive model f that generalizes to future, unseen domains $\mathcal{C}^{\text{test}} = \{c_{t+1}, c_{t+2}, \ldots, c_n\}$.

3 Methodology

Our approach is built on PFNs, which use ICL to learn the learning algorithm itself. This approach also has a theoretical foundation as described by Müller et al. [42]: It can be viewed as approximating Bayesian prediction for a prior defined by the synthetic datasets. The trained PFN will approximate the posterior predictive distribution (PPD) and thus return a Bayesian prediction for the specified distribution over artificial datasets used during PFN training.

Hollmann et al. [27] introduce a prior based on Structural Causal Models (SCMs; 46; 48) to model complex feature dependencies and potential causal mechanisms underlying tabular data. To sample one dataset, this prior samples an SCM, which is then used to sample the examples in the dataset. In this approach, each causal representation of a sampled SCM is converted into a functional representation to enable forward computation and dataset sampling.

We extend TabPFN's prior to model distribution shifts, allowing the model to expand its posterior predictive distribution (PPD) calculations to incorporate temporal domain information. We propose modeling distribution shifts via shifting edges of the SCM over our temporal domain. Furthermore, we introduce a *hypernetwork* to model these shifts. This *hypernetwork* is itself an SCM with feature nodes specifying the magnitude of edge shifts in the base SCM's functional graph.

To do this, we sample the temporal domains $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ and for each domain c_k, we sample the number of samples n_{c_k} it contains. An illustration showing exemplary domain distributions across four datasets can be found in Figure 5 in Appendix A.6.

We then select a sparse subset of relationships in the causal representation of the SCM to undergo temporal shifts based on the evidence that sparse shifts allow for causal reasoning [47]. For these selected edges, the *hypernetwork* is used to sample shift parameters that govern the corresponding edges in the functional representation. It takes temporal domains $\mathcal{C}$ as input and, through a single forward pass on the corresponding functional representation, produces dynamic edge shifts for each edge weight $w_{i,j}$ in the original SCM that corresponds to a causal relationship that should be shifted. This design allows for Bayesian reasoning over the edge shifts and enables the *hypernetwork* to generate complex, often correlated shifts over time. For better illustration, we have visualized this approach in Figure 2 and a selection of the functions generated by a hypernet in Figure 6. A high-level outline of the sampling procedure is detailed in Algorithm 1.

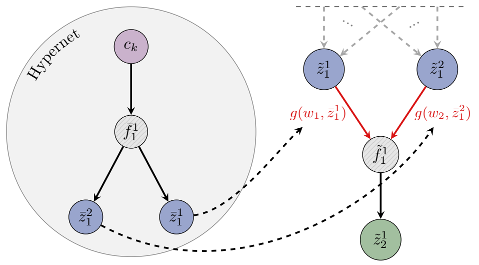


Figure 2: Diagram illustrating the integration of a hypernet for adaptive edge shifting across evolving temporal domains. On the right, the primary network $\tilde{\mathcal{G}}$ generates data samples over multiple time domains, with red arrows indicating shifted edges. On the left, the hypernet - an auxiliary network $\tilde{\mathcal{H}}$ - takes an input domain $c_k \in \mathcal{C}$ and outputs parameters to adaptively shift each edge weight w_i in the base network.

When encoded as inputs to our model, temporal domains and features are normalized, and to effectively encode temporal information , we use Time2Vec [33]. It converts each temporal domain index into an m-dimensional vector, using linear and sinusoidal functions characterized by learned parameters ω_i and φ_i. Specifically, the Time2Vec transformation for a temporal index $c_k \in \mathcal{C}$ is formulated as:

$$\mathbf{t2v}(c_k)[i] = \begin{cases} \omega_i c_k + \varphi_i, & \text{if } \ i = 0. \\ \sin(\omega_i c_k + \varphi_i), & \text{if } \ 1 \leq i < m. \end{cases} \tag{1}$$

4 Experiments

We evaluate the performance using in-distribution (ID) and out-of-distribution (OOD) splits across multiple datasets, employing state-of-the-art tabular prediction methods, including TabPFN variants. Detailed information about the evaluation strategy, metrics, datasets, baselines, and TabPFN setup is provided in Appendix A.1.

Quantitative Evaluation Our method demonstrates superior predictive performance in all metrics for OOD data across 18 test datasets, as detailed in Table 3. Compared to the strongest baseline, it improves accuracy from 0.688 to 0.744, F1 from 0.62 to 0.689, and ROC from 0.786 to 0.832. Furthermore, Drift-Resilient TabPFN shows much stronger calibration on OOD samples, improving ECE from 0.119 to 0.091. While baselines are often overconfident on OOD data, Drift-Resilient TabPFN is able to predict uncertainty accurately. Since our method focuses on enhancing OOD robustness rather than optimizing ID tasks, we find lower predictive performance on ID tasks. While performance gains are observed on both, real-world and synthetic data, we observe stronger improvements on synthetic datasets. This can be partly attributed to the, on average, stronger distribution shifts between ID and OOD data in our synthetic benchmark. Furthermore, real-world datasets often show multifaceted and complex shifts that are much more difficult to extrapolate into the future.

Qualitative Analysis. Next, we take an in-depth look at the predictions made by our method. Figure 3 illustrates the decision boundaries of our method and TabPFN$_{\text{base}}$ on the synthetic *Intersecting Blobs dataset*. In this evaluation, we restrict the training domains to $\mathcal{C}^{\text{train}} = \{0, 1, 2, 3\}$ and aim to predict samples in test domains $\mathcal{C}^{\text{test}} = \{4, 5, 6\}$ without adding additional data to the training set. This setup requires the model to extrapolate the temporal shifts into the future based solely on existing training data. In this setting, our model accurately extrapolates decision boundaries to future domains, while TabPFN$_{\text{base}}$ tends to retain its initial boundary. Our analysis reveals two key attributes of our model: (i) The model decreases prediction certainty over time, improving calibration. (ii) Our model adjusts the decision boundary dynamically, boosting accuracy.

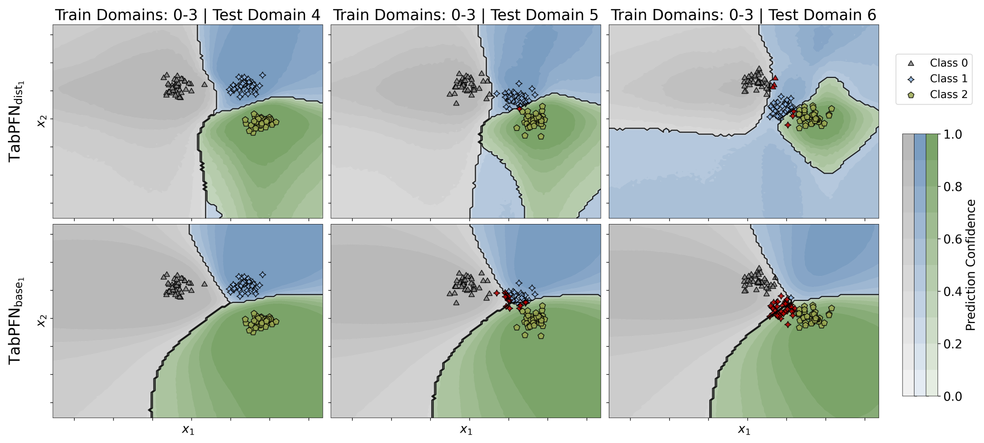


Figure 3: This figure displays the predictive behavior of $\text{TabPFN}_{\text{dist}}$ in the top row and $\text{TabPFN}_{\text{base}}$ in the bottom row on the Intersecting Blobs dataset. It illustrates how each model adapts to unseen test domains when trained on domains $\mathcal{C}^{\text{train}} = \{0, 1, 2, 3\}$. The baseline is given the domain indices as a feature in train and test. The coloring indicates the probability of the most likely class at each point. Incorrectly classified samples are highlighted in red.

5 Conclusions

In this work, we presented a Bayesian approach to address the issue of temporal domain generalization in tabular data. Specifically, we focused on enhancing TabPFN to improve its robustness to temporal distribution shifts. Within this framework, we introduced a novel approach that changes the causal relationships in the SCM prior over time, thereby enabling TabPFN to inherently adapt to these shifts. Our method outperforms all baselines on the evaluated datasets and demonstrates notable improvements both qualitatively and quantitatively, particularly on synthetic OOD datasets. Furthermore, it requires no hyperparameter tuning, is not limited to particular types of distribution shifts and takes only 10.9s for training and prediction combined.

5.1 Limitations

Despite these advancements, our methodology inherits certain limitations from the underlying TabPFN model. (1) Due to the quadratic scaling of the attention mechanism with respect to the number of samples, our method does not scale to large datasets. Here, our research will benefit from the continued improvements of TabPFN, ICL, and sequence-based models in general. (2) The TabPFN, like many transformer-based models, acts as a "black box", making it challenging to interpret the model's predictions and understand the recognized distribution shifts. (3) The underlying prior for structural causal models with sparse mechanism shifts may not accurately describe all real-world datasets.

5.2 Broader Impact

As a general method for handling distribution shifts in tabular data, Drift-Resilient TabPFN doesn't have immediate societal implications like AI systems designed to automate tasks or replace jobs. However, it offers potential benefits such as extended model longevity and enhanced decision-making. By adapting to distribution shifts, our approach prolongs the usability of ML models, reducing retraining needs, saving costs and CO2 and ensuring stable performance.

References

[1] (2023). *Proceedings of the International Conference on Learning Representations (ICLR'23)*. Published online: `iclr.cc`.

[2] Adam, G. A., Chang, C.-H. K., Haibe-Kains, B., and Goldenberg, A. (2020). Hidden risks of machine learning applied to healthcare: Unintended feedback loops between models and future data causing model degradation. In Doshi-Velez, F., Fackler, J., Jung, K., Kale, D., Ranganath, R., Wallace, B., and Wiens, J., editors, *Proceedings of the 5th Machine Learning for Healthcare Conference*, volume 126 of *Proceedings of Machine Learning Research*, pages 710–731. PMLR.

[3] Akbilgic, O. (2013). Istanbul Stock Exchange. UCI Machine Learning Repository.

[4] Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2020). Invariant risk minimization.

[5] Bai, G., Ling, C., and Zhao, L. (2023). Temporal domain generalization with drift-aware dynamic neural networks. In [1]. Published online: `iclr.cc`.

[6] Balaji, Y., Sankaranarayanan, S., and Chellappa, R. (2018). Metareg: Towards domain generalization using meta-regularization. In [7].

[7] Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors (2018). *Proceedings of the 31st International Conference on Advances in Neural Information Processing Systems (NeurIPS'18)*. Curran Associates.

[8] Beucler, T., Gentine, P., Yuval, J., Gupta, A., Peng, L., Lin, J., Yu, S., Rasp, S., Ahmed, F., O'Gorman, P. A., et al. (2024). Climate-invariant machine learning. *Science Advances*, 10(6):eadj7250.

[9] Bifet, A. and Ikonomovska, E. (2009). Data Expo competition. OpenML.

[10] Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

[11] Candanedo, L. (2016). Occupancy Detection. UCI Machine Learning Repository.

[12] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H., editors, *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*. Curran Associates.

[13] Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019). Efficient lifelong learning with A-GEM. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*. Published online: `iclr.cc`.

[14] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Krishnapuram, B., Shah, M., Smola, A., Aggarwal, C., Shen, D., and Rastogi, R., editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, pages 785–794. ACM Press.

[15] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, volume 98. Proceedings of Machine Learning Research.

[16] De Cock, D. (2011). Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19.

[17] Ding, F., Hardt, M., Miller, J., and Schmidt, L. (2021). Retiring adult: New datasets for fair machine learning. In [51].

[18] Dispenzieri, A., Katzmann, J., Kyle, R., Larson, D., Therneau, T., Colby, C., Clark, R., Mead, G., Kumar, S., Melton, L., and Rajkumar, S. (2012). Use of nonclonal serum immunoglobulin free light chains to predict overall survival in the general population. *Mayo Clinic proceedings. Mayo Clinic*, 87:517–23.

[19] Ferreira, R., Martiniano, A., and Sassi, R. (2018). Behavior of the urban traffic of the city of Sao Paulo in Brazil. UCI Machine Learning Repository.

[20] Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In Bazzan, A. L. C. and Labidi, S., editors, *Advances in Artificial Intelligence – SBIA 2004*, pages 286–295, Berlin, Heidelberg. Springer Berlin Heidelberg.

[21] Ganesan, R., Mahajan, V., Singla, K., Konar, S., Samra, T., Sundaram, S. K., Suri, V., Garg, M., Kalra, N., Puri, G. D., et al. (2021). Mortality prediction of covid-19 patients at intensive care unit admission. *Cureus*, 13(11).

[22] Garcia, S. and Herrera, F. (2008). An extension on" statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of machine learning research*, 9(12).

[23] Gardner, J., Popovic, Z., and Schmidt, L. (2024). Benchmarking distribution shift in tabular data with tableshift. *Advances in Neural Information Processing Systems*, 36.

[24] Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943.

[25] Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In [44].

[26] Harries, M. (1999). *Splice-2 comparative evaluation: Electricity Pricing*. University of New South Wales, School of Computer Science and Engineering [Sydney].

[27] Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2023). TabPFN: A transformer that solves small tabular classification problems in a second. In [1]. Published online: `iclr.cc`.

[28] Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

[29] Islam, M. M. F., Ferdousi, R., Rahman, S., and Bushra, H. Y. (2020). Likelihood prediction of diabetes at early stage using data mining techniques. In Gupta, M., Konar, D., Bhattacharyya, S., and Biswas, S., editors, *Computer Vision and Machine Intelligence in Medical Image Analysis*, pages 113–125, Singapore. Springer Singapore.

[30] Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963.

[31] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. (2018). Averaging weights leads to wider optima and better generalization. In Globerson, A. and Silva, R., editors, *Proceedings of The 34th Uncertainty in Artificial Intelligence Conference (UAI'18)*. AUAI Press.

[32] Janosi, A., Steinbrunn, W., Pfisterer, M., and Detrano, R. (1988). Heart Disease. UCI Machine Learning Repository.

[33] Kazemi, S. M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., and Brubaker, M. (2020). Time2vec: Learning a vector representation of time.

[34] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Proceedings of the 30th International Conference on Advances in Neural Information Processing Systems (NeurIPS'17)*. Curran Associates.

[35] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

[36] Kyle, R., Therneau, T., Rajkumar, S., Larson, D., Plevak, M., Offord, J., Dispenzieri, A., Katzmann, J., and Melton, L. (2006). Prevalence of monoclonal gammopathy of undetermined significance. *The New England journal of medicine*, 354:1362–9.

[37] Lucas, Y., Portier, P.-E., Laporte, L., Calabretto, S., He-Guelton, L., Oble, F., and Granitzer, M. (2019). Dataset shift quantification for credit card fraud detection. In *2019 IEEE second international conference on artificial intelligence and knowledge engineering (AIKE)*, pages 97–100. IEEE.

[38] Martiniano, A. and Ferreira, R. (2018). Absenteeism at work. UCI Machine Learning Repository.

[39] Montiel, J., Read, J., Bifet, A., and Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5.

[40] Motiian, S., Piccirilli, M., Adjeroh, D. A., and Doretto, G. (2017). Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5715–5725.

[41] Muandet, K., Balduzzi, D., and Schölkopf, B. (2013). Domain generalization via invariant feature representation. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*. Omnipress.

[42] Müller, S., Hollmann, N., Arango, S., Grabocka, J., and Hutter, F. (2022). Transformers can do bayesian inference. In *Proceedings of the International Conference on Learning Representations (ICLR'22)*. Published online: `iclr.cc`.

[43] Nasery, A., Thakur, S., Piratla, V., De, A., and Sarawagi, S. (2021). Training for the future: A simple gradient interpolation loss to generalize along time. In [51].

[44] Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors (2022). *Proceedings of the 36th International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*.

[45] Pasterkamp, G., Den Ruijter, H. M., and Libby, P. (2017). Temporal shifts in clinical presentation and underlying mechanisms of atherosclerotic disease. *Nature Reviews Cardiology*, 14(1):21–29.

[46] Pearl, J. (2009). *Causality*. Cambridge University Press, 2 edition.

[47] Perry, R., Kügelgen, J. V., and Schölkopf, B. (2022). Causal discovery in heterogeneous environments under the sparse mechanism shift hypothesis. In [44].

[48] Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

[49] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A., and Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. In [7].

[50] Ramana, B. and Venkateswarlu, N. (2012). ILPD (Indian Liver Patient Dataset). UCI Machine Learning Repository.

[51] Ranzato, M., Beygelzimer, A., Nguyen, K., Liang, P., Vaughan, J., and Dauphin, Y., editors (2021). *Proceedings of the 34th International Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*. Curran Associates.

[52] Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. (2020). Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*. Published online: iclr.cc.

[53] Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.

[54] Smith, J., Everhart, J., Dickson, W., Knowler, W., and Johannes, R. (1988). Using the adap learning algorithm to forcast the onset of diabetes mellitus. *Proceedings - Annual Symposium on Computer Applications in Medical Care*, 10.

[55] Stolfi, D. (2019). Parking Birmingham. UCI Machine Learning Repository.

[56] Strack, B., DeShazo, J. P., Gennings, C., Olmo, J. L., Ventura, S., Cios, K. J., and Clore, J. N. (2014). Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records. *BioMed Research International*, 2014:781670. Publisher: Hindawi Publishing Corporation.

[57] Sun, B. and Saenko, K. (2016). Deep coral: Correlation alignment for deep domain adaptation. In Hua, G. and Jégou, H., editors, *Computer Vision – ECCV 2016 Workshops*, pages 443–450, Cham. Springer International Publishing.

[58] van Breugel, B. and van der Schaar, M. (2024). Why tabular foundation models should be a research priority. *arXiv preprint arXiv:2405.01147*.

[59] Vela, D., Sharp, A., Zhang, R., Nguyen, T., Hoang, A., and Pianykh, O. S. (2022). Temporal quality degradation in AI models. *Scientific Reports*, 12(1):11654.

[60] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

[61] Yao, H., Choi, C., Cao, B., Lee, Y., Koh, P. W., and Finn, C. (2022a). Wild-time: A benchmark of in-the-wild distribution shift over time. In [44].

[62] Yao, H., Wang, Y., Li, S., Zhang, L., Liang, W., Zou, J., and Finn, C. (2022b). Improving out-of-distribution robustness via selective augmentation.

[63] Young, Z. and Steele, R. (2022). Empirical evaluation of performance degradation of machine learning-based predictive models–a case study in healthcare information systems. *International Journal of Information Management Data Insights*, 2(1):100070.

[64] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In Precup, D. and Teh, Y., editors, *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, volume 70. Proceedings of Machine Learning Research.

[65] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*. Published online: `iclr.cc`.

[66] Žliobaitė, I. (2011). Combining similarity in time and space for training set formation under concept drift. *Intelligent Data Analysis*, 15(4):589–611.

A Appendix

A.1 Experimental Setup

Evaluation Strategy. We evaluate analogous to the Eval-Fix setting outlined in Wild-Time [61], measuring both, in-distribution (ID) and out-of-distribution (OOD) performance. Here, each dataset $\mathcal{D}$ is split into three subsets: $\mathcal{D}^{\text{train}}$, $\mathcal{D}^{\text{ID}}$, and $\mathcal{D}^{\text{OOD}}$. Splits are based on a randomly sampled temporal domain c_k that serves as the boundary between the train and test (OOD) portion. We only use such splits, where $\mathcal{D}^{\text{train}}$ comprises between 30% and 80% of the total domains and samples. To assess ID performance, we subsample 10% of the instances in each domain of $\mathcal{D}^{\text{train}}$ as the ID test set and the remainder as the train set. An illustration of the Eval-Fix strategy is provided in Figure 8.

Each class in the training set is required to be represented in both $\mathcal{D}^{\text{ID}}$ and $\mathcal{D}^{\text{OOD}}$, and vice versa. For all datasets, we generate three random splits and average metrics across these splits. We had to limit the number of splits to three due to the constrained number of available domains and the requirement for classes to be present in both the train and test splits. Each method is trained three times, and we report the average and 95%-confidence intervals calculated across model initializations.

Metrics. We evaluate Accuracy, F1-Score (Harmonic mean of precision and recall, useful in imbalanced datasets), ROC AUC (Area under the receiver operating characteristic curve), and ECE (Expected Calibration Error; reflects the reliability of the model's probability outputs)

Datasets. Our benchmark comprises 18 test datasets, 8 synthetic and 10 real-world. In addition, 12 validation datasets, 4 synthetic and the remaining 8 real-world, were used to optimize the hyperparameters of our approach via random search. The ground truth domain indices $c_k \in \mathcal{C}$ were known for synthetic datasets. For real-world datasets, we approximated domain indices $\hat{c_k}$ based on features that encode temporal information, which we transformed into discrete intervals. Also, some real-world datasets required subsampling due to their large size, which was beyond the current architecture of TabPFN.

Baseline Setup. Our baselines include state-of-the-art methods for tabular prediction. These include advanced Gradient Boosted Decision Trees like CatBoost [49], XGBoost [14], and LightGBM [34], which have demonstrated superior performance to standard neural network approaches in handling tabular data [25]. We also include TabPFN in its unmodified form (TabPFN$_{\text{base}}$; 27). Methods from the Wild-Time benchmark are examined separately and detailed in Section A.10.6 of the Appendix. All baseline methods besides TabPFN are subject to a time budget of 1,200 seconds on 8 CPUs and 1 GPU. For each method except TabPFN, which does not require tuning, a random hyperparameter search with 3-fold time series cross-validation was used. We chose the best-performing hyperparameters based on OOD ROC AUC within the allocated time.

Among our baselines, we considered three strategies:

1. Providing the full dataset $\mathcal{D}^{\text{train}}$ along with the corresponding domain indices $\mathcal{C}^{\text{train}}$ as a feature, aiming to allow for better reasoning of the shifts in the dataset (all dom. w. ind.).
2. Using the dataset without domain indices $\mathcal{D}^{\text{train}} = \{(\boldsymbol{x}_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^{n}$ (all dom. wo. ind.).
3. Limiting the training set to samples from the last training domain c_t. In this setting, we also omit the corresponding domain indices, resulting in the set $\mathcal{D}_{\hat{c}_t}^{\text{train}} = \{(\boldsymbol{x}_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^{n_{c_t}}$ (last dom. wo. ind.). The rationale behind the last scenario is to provide only training data closest to the subsequent test distribution. This strategy is not used for distribution shift baselines.

TabPFN Setup. For the TabPFN variants, both the original and our modified method (TabPFN$_{\text{dist}}$) were pre-trained for 30 epochs across 8 GPUs. This results in a total of 30,720,000 synthetically generated datasets processed during pre-training. While this pre-training step is moderately expensive, it is done offline, in advance, and only once as part of our algorithm development. Furthermore, the preprocessing parameters of both methods were optimized once on the validation datasets by random search over 300 configurations. We chose the configurations that yielded the best OOD ROC AUC performance. The resulting model and hyperparameters are used for all datasets, resulting in, on average, 110 times faster training and prediction time on our benchmark.

A.2 Related Work

While DG has drawn considerable attention in the research community [41, 6, 40, 4, 52, 65, 62], its temporal variant remains underexplored. The following sections provide a more in-depth review of existing works, considering their approach and limitations concerning temporal DG.

Wild-Time Benchmark. Wild-Time [61] is a benchmark dedicated to studying the real-world implications of temporal distribution shifts. The benchmark employs a set of techniques such as classical supervised learning, continual learning, temporal invariant learning, self-supervised learning, and Bayesian learning, which are evaluated on five datasets.

Despite the many techniques evaluated, Wild-Time reveals a significant performance gap between in-distribution and out-of-distribution data, with none of the 13 tested methods consistently outperforming the standard Empirical Risk Minimization (ERM) approach. In our study, we employ their evaluation strategy *Eval-Fix* and benchmark our approach against 11 Wild-Time methods. A comprehensive overview of the methods in Wild-Time can be found in Appendix A.11.

Temporal DG with Drift-Aware Dynamic Neural Networks (DRAIN). DRAIN [5] employs a Bayesian framework alongside a recurrent neural network (RNN) for predicting the dynamics of model parameters across temporal domains. DRAIN's primary focus is on adaptively capturing concept shifts for the immediate next domain and , by design, cannot directly predict an arbitrary future domain without first processing the intervening domains. DRAIN implicitly assumes uniform temporal intervals, as it does not incorporate gaps between successive temporal domains to adjust the scale of the expected shifts.

Gradient Interpolation (GI). GI [43] focuses on improving temporal DG by incorporating the temporal domain as an explicit feature, utilizing a specialized Gradient Interpolation loss function, employing a time-sensitive activation called Leaky Temporal ReLU (TReLU), and enhancing domain reasoning through Time2Vec preprocessing [33]. Compared to our method, GI primarily focuses on generalization to the near future. Their loss term faces challenges in handling non-linear shifts in combination with large time gaps.

We exclude both DRAIN and Gradient Interpolation (GI) from our primary evaluation since these methods, while innovative, are not designed for extrapolation to future domains beyond the near future. Even on the Rotated Two Moons Dataset, which has been used to demonstrate the capabilities of DRAIN and GI, we show that Drift-Resilient TabPFN clearly outperforms. See Appendix A.10.3 for details.

A.3 Computational Resources

In the course of our research on Drift-Resilient TabPFN, we employed substantial computational resources across various stages of model development, training, and evaluation. The computation specifics are as follows:

1. Infrastructure: The experiments were conducted on an internal SLURM cluster equipped with RTX 2080 TI GPUs and CPUs of type AMD EPYC 7502, 32C/64T, @ 2.50-3.35GHz.

2. Baseline Experiments: Each baseline experiment utilized 8 CPUs, 1 GPU, and 62.5 GB RAM, with a hyperparameter optimization (HPO) runtime of 1200 seconds per dataset per split, repeated three times to ensure reliability.

3. Pre-training for TabPFN-base and Drift-Resilient TabPFN: These models were pre-trained three times each, requiring 64 CPUs, 8 GPUs, and 500 GB RAM, requiring approximately 7 and 8 days respectively.

4. Hyperparameter Optimization for Drift-Resilient TabPFN and TabPFN-base: We used 32 CPUs, 4 GPUs, and 250 GB RAM, running approximately 40 configurations and taking about one day per pre-training session for optimizing the hyperparameters of the novel prior-data generating mechanism of Drift-Resilient TabPFN. Both TabPFN-base and Drift-Resilient TabPFN underwent preprocessing optimization that utilized 8 CPUs, 1 GPU, and 62.5 GB RAM across 300 runs, each lasting between 0.5 to 1 hour.

Additional computational resources were allocated for method development tests and other experimental setups not detailed in the final publication. Thus the full scope of the research required more computational resources than those detailed above due to these preliminary and unreported experiments.

A.4 Ablations

Is our model's performance mostly based on Time2Vec preprocessing? To address this question, we conducted an ablation study where we trained a model with temporal domain indices normalized but not subjected to Time2Vec preprocessing (No T2V).

Table 1 presents the performance metrics of Drift-Resilient TabPFN, TabPFN-base, and our ablation model. The results indicate that while Time2Vec preprocessing slightly improves model performance, it is not the main reason for it. Rather, the substantial improvements in performance are largely due to our prior construction, used during the pre-training phase of the model.

Table 1: Comparison of Drift-Resilient TabPFN with respect to the stated ablations. Metrics include ROC AUC and accuracy for both in-distribution (ID) and out-of-distribution (OOD) data.

Model	**Variant**	**Acc. ↑**		**F1 ↑**		**ROC ↑**		**ECE ↓**	
		OOD	ID	OOD	ID	OOD	ID	OOD	ID
TabPFN$_{\text{dist}}$	all dom. w. ind.	**0.744** $_{.018}$	0.879 $_{.012}$	**0.689** $_{.028}$	0.837 $_{.022}$	**0.832** $_{.018}$	0.932 $_{.002}$	**0.091** $_{.006}$	0.074 $_{.014}$
No T2V	all dom. w. ind.	0.741	0.874	0.684	0.828	0.831	0.929	0.093	0.072
	all dom. w. ind.	0.688 $_{.01}$	**0.885** $_{.01}$	0.62 $_{.012}$	**0.847** $_{.017}$	0.786 $_{.007}$	**0.935** $_{.01}$	0.119 $_{.006}$	**0.067** $_{.005}$
TabPFN$_{\text{base}}$	all dom. wo. ind.	0.645 $_{.011}$	0.852 $_{.016}$	0.579 $_{.014}$	0.801 $_{.02}$	0.736 $_{.001}$	0.914 $_{.007}$	0.202 $_{.011}$	0.076 $_{.007}$
	last dom. wo. ind.	0.67 $_{.005}$	0.867 $_{.004}$	0.609 $_{.004}$	0.823 $_{.011}$	0.76 $_{.003}$	0.915 $_{.019}$	0.181 $_{.003}$	0.128 $_{.007}$

A.5 Plots Illustrating Types of Distribution Shifts

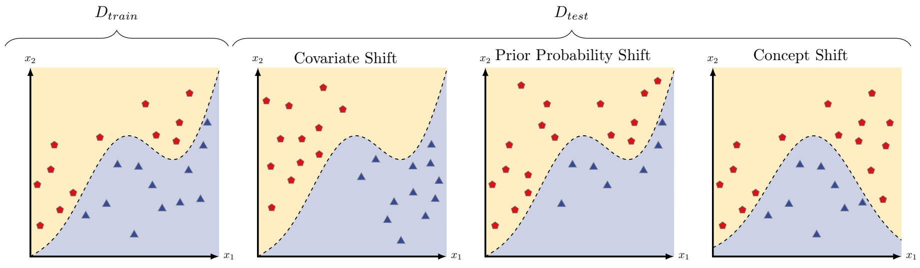


Figure 4: Illustration of initial and shifted data distributions alongside their optimal decision boundaries. The left panel depicts the initial classification dataset with two features and its true-data-optimal decision boundary. The right panel presents the dataset subjected to the three primary types of distribution shifts observed during test time.

A.6 Plots Illustrating Sampled Parameters of the Adjusted Prior

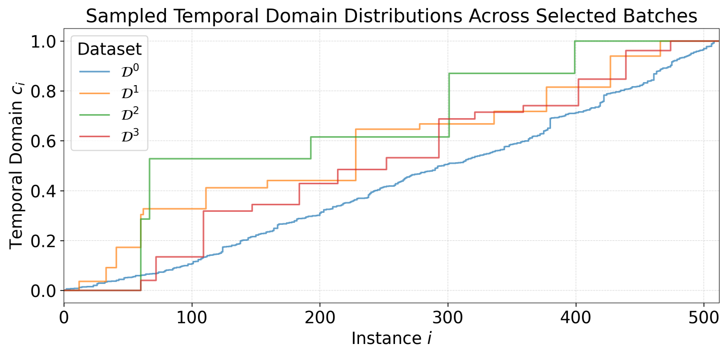


Figure 5: Share of temporal domains in exemplary datasets prior seen up to any instance i. The figure illustrates the range and structure of the sampled temporal domains $c_k \in \mathcal{C}$ across four representative datasets. It highlights variations in domain size and demonstrates the presence of arbitrary gaps, simulating irregularities in data sampling.

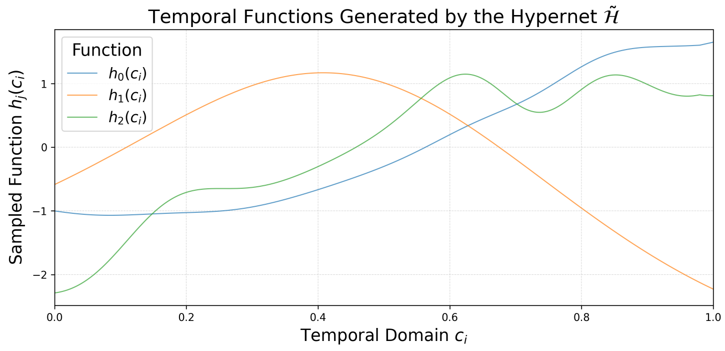
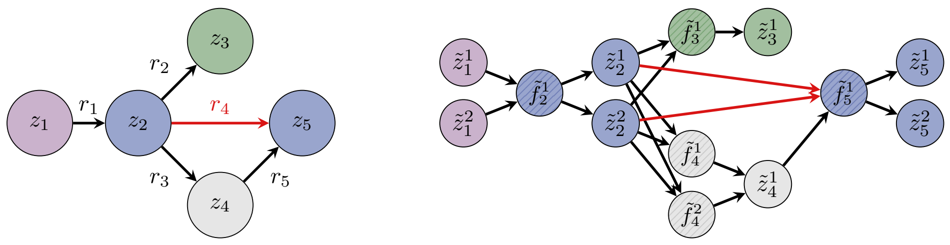


Figure 6: This figure presents three exemplary functions sampled from nodes within the network of a hypernet $\tilde{\mathcal{H}}$. In the plot, the x-axis represents the input temporal domain $c_k \in \mathcal{C}$, while the y-axis displays the corresponding node activation.

A.7 Transformation of the Causal Representation to a Functional Representation



(a) Sampled SCM

(b) Functional representation

Figure 7: Illustrative transformation of an SCM to one exemplary functional representation. Shaded nodes indicate that their activations cannot be sampled. Feature nodes are blue, the target node is green, input/noise nodes are purple, and all others are gray. The figure also shows the mapping of shifted edges between a causal relationship and its functional form in red, ensuring that shifts specifically target the intended causal relationships without affecting others.

A.8 Algorithmic Overview of Our Approach

Algorithm 1 This algorithm provides a high-level overview for generating a synthetic dataset in our prior. Although steps are depicted sequentially for clarity, many can be parallelized in actual implementation.

```
1:  procedure SampleDataset
2:      $\mathcal{G} \leftarrow$ SampleSCM()                          ▷ Sample data-generating SCM
3:      $\tilde{\mathcal{G}} \leftarrow \mathcal{G}$.Expand()                 ▷ Expand to functional representation

4:      $\mathcal{H} \leftarrow$ SampleSCM()                          ▷ Sample hypernet SCM
5:      $\tilde{\mathcal{H}} \leftarrow \mathcal{H}$.Expand()                 ▷ Expand to functional representation

6:      $\mathcal{C} \leftarrow \{c_1, c_2, \ldots, c_t\}$                ▷ Sample temporal domains
7:      $\mathcal{D} \leftarrow \emptyset$                               ▷ Initialize dataset

8:      for all $c_k \in \mathcal{C}$ do
9:          $\boldsymbol{\omega}_{c_k} \leftarrow \tilde{\mathcal{H}}$.Forward($c_k$)              ▷ Sample edge shifts
10:         $\tilde{\mathcal{G}}_{c_k} \leftarrow \tilde{\mathcal{G}}$.Update($\boldsymbol{\omega}_{c_k}$)             ▷ Update edge weights

11:         $\mathcal{D}_{c_k} \leftarrow \emptyset$                           ▷ Initialize sub-dataset
12:         for all $i \in \{1, ..., n_{c_k}\}$ do                  ▷ Sample sub-dataset
13:             $(\boldsymbol{x}_i, y_i, c_k) \leftarrow \tilde{\mathcal{G}}_{c_k}$.Forward($\epsilon_i$)
14:             $\mathcal{D}_{c_k} \leftarrow \mathcal{D}_{c_k} \cup \{(\boldsymbol{x}_i, y_i, c_k)\}$
15:         end for

16:         $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{c_k}$                        ▷ Extend dataset
17:     end for

18:     return $\mathcal{D}$                                    ▷ Return dataset
19: end procedure
```

A.9 Illustration of Adopted Evaluation Strategy

In Figure 8, we provide an illustration of the Eval-Fix evaluation strategy, originally proposed by Yao et al. [61] in the context of the Wild-Time benchmark. It should be noted that the formalism has been adapted to align with the notation used in this paper.

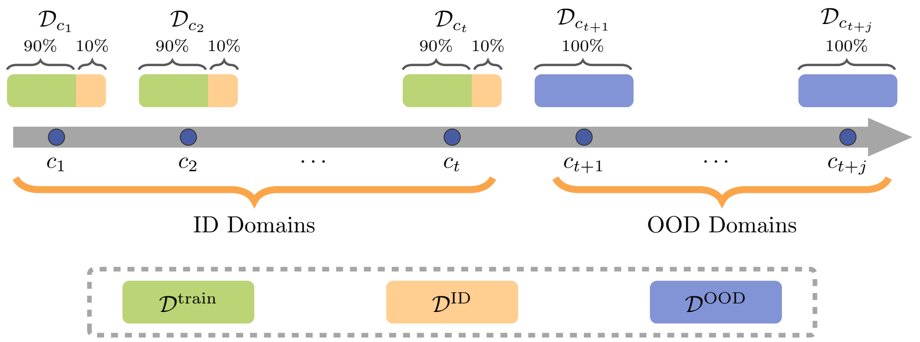
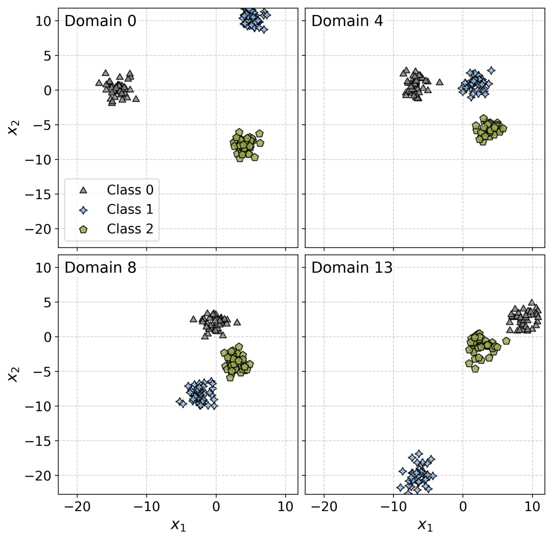


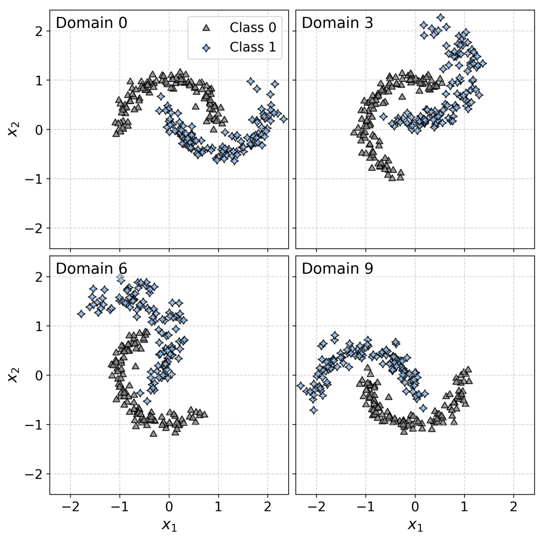
Figure 8: Adapted from the Wild-Time benchmark [61], this illustration portrays the Eval-Fix evaluation strategy employed in our study. The domain boundary is indicated by c_t, beyond which datasets are considered part of the out-of-distribution (OOD) test set $\mathcal{D}^{\text{OOD}}$. To evaluate in-distribution (ID) performance, we subsample 10% of the samples from each dataset prior to this boundary, forming the datasets $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{ID}}$.

A.10 Additional Experiments

A.10.1 Qualitative Analysis: Overview of the Shifts in the Datasets Analyzed. This section offers plots of the Intersecting Blobs and Rotated Two Moons datasets across specific temporal domains. The Intersecting Blobs dataset is visualized in Figure 9a for domains $\mathcal{C} = \{0, 4, 8, 13\}$. The Rotated Two Moons dataset is presented in Figure 9b for domains $\mathcal{C} = \{0, 3, 6, 9\}$.



(a) Intersecting Blobs



(b) Rotated Two Moons

Figure 9: This figure shows the temporal shifts of two synthetic datasets across selected domains.

A.10.2 Qualitative Analysis: Decision Boundaries on Rotated Two Moons Dataset. In addition to the qualitative analysis conducted for the Intersecting Blobs dataset in the main text of this work, this subsection provides additional illustrations of the Rotated Two Moons dataset. The corresponding visualizations for our approach as well as the TabPFN baseline are provided in Figure 10.

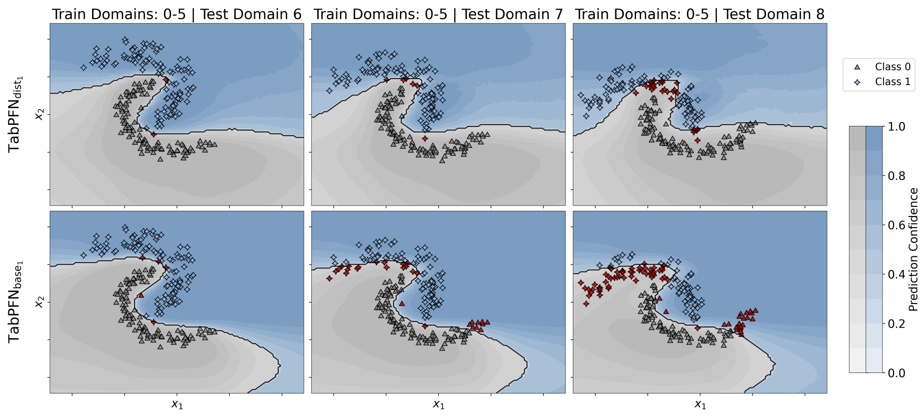


Figure 10: This figure contrasts the predictive behavior of TabPFN$_{\text{dist}}$ and TabPFN$_{\text{base}}$ on the Rotated Two Moons dataset. It illustrates how each model adapts to different testing domains when trained on domains $\mathcal{C}^{\text{train}} = \{0, 1, 2, 3, 4, 5\}$. The color shading indicates the maximum class probability at each point, with decision boundaries shown when this probability exceeds 50%. Incorrectly classified samples are highlighted in red.

A.10.3 Qualitative Analysis: Comparison Against DRAIN and GI. To show our improved performance compared to the state-of-the-art methods DRAIN [5] and GI [43], we compare our method with the qualitative analysis performed by the authors of DRAIN on the Rotated Two Moons dataset. In this setting, all domains except the last are used for training, with the final domain reserved for testing. We illustrate our method's decision boundary compared to those provided by DRAIN in Figure 11. The accuracy results are listed in Table 2. As the contour levels are unknown, we display only the pure decision boundary for clearer comparison. Our analysis shows that our model forecasts the rotation of the two moons more accurately compared to the baselines and adapts its decision boundary more precisely.

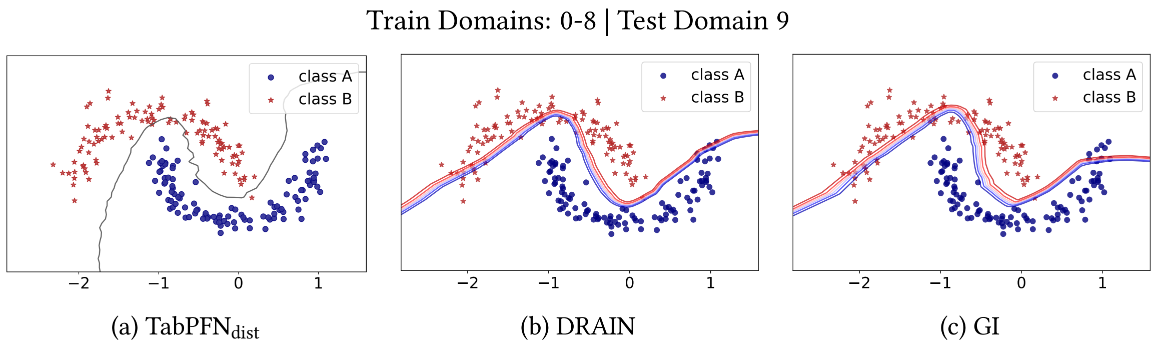


Figure 11: Comparison of our method against DRAIN [5] and GI [43] on the Rotated Two Moons dataset. The models were trained on domains $\mathcal{C} = \{0, 1, \ldots, 8\}$ and tested on domain 9. While the authors of DRAIN present different, unknown levels of the decision boundary, we present the decision boundary with 50% probability. The plots for DRAIN and GI were taken from Bai et al. [5].

Table 2: Comparison of Drift-Resilient TabPFN against DRAIN and GI on the Rotated Two Moons dataset. The metric reported is the mean out-of-distribution (OOD) accuracy along with the standard deviation. Results for DRAIN and GI are taken from Bai et al. [5].

Model	OOD Acc. ↑
TabPFN$_{\text{dist}}$	**0.98** $_{.002}$
DRAIN	0.968 $_{.012}$
GI	0.965 $_{.014}$

A.10.4 Quantitative Analysis. In the following, the quantitative results are first presented across all datasets and then separately for synthetic and real-world data.

Table 3: Comparison of Drift-Resilient TabPFN with various baselines and settings across the combined **real-world** and **synthetic** datasets. Metrics include accuracy, F1, ROC, and ECE for both in-distribution (ID) and out-of-distribution (OOD) data, averaged over three initializations and reported with 95% confidence intervals. The best mean of each metric is marked in bold. Metric arrows indicate optimization direction.

Model	Variant	Acc. ↑		F1 ↑		ROC ↑		ECE ↓	
		OOD	ID	OOD	ID	OOD	ID	OOD	ID
TabPFN$_{\text{dist}}$	all dom. w. ind.	**0.744** $_{.018}$	0.879 $_{.012}$	**0.689** $_{.028}$	0.837 $_{.022}$	**0.832** $_{.018}$	0.932 $_{.002}$	**0.091** $_{.006}$	0.074 $_{.014}$
TabPFN$_{\text{base}}$	all dom. w. ind.	0.688 $_{.01}$	**0.885** $_{.01}$	0.62 $_{.012}$	**0.847** $_{.017}$	0.786 $_{.007}$	**0.935** $_{.01}$	0.119 $_{.006}$	**0.067** $_{.005}$
	all dom. wo. ind.	0.645 $_{.011}$	0.852 $_{.016}$	0.579 $_{.014}$	0.801 $_{.02}$	0.736 $_{.001}$	0.914 $_{.007}$	0.202 $_{.011}$	0.076 $_{.007}$
	last dom. wo. ind.	0.67 $_{.005}$	0.867 $_{.004}$	0.609 $_{.004}$	0.823 $_{.011}$	0.76 $_{.003}$	0.915 $_{.019}$	0.181 $_{.003}$	0.128 $_{.007}$
CatBoost	all dom. w. ind.	0.677 $_{.006}$	0.874 $_{.007}$	0.62 $_{.005}$	0.836 $_{.01}$	0.766 $_{.003}$	0.919 $_{.011}$	0.222 $_{.007}$	0.084 $_{.009}$
	all dom. wo. ind.	0.632 $_{.003}$	0.836 $_{.013}$	0.568 $_{.005}$	0.781 $_{.009}$	0.714 $_{.012}$	0.894 $_{.014}$	0.24 $_{.02}$	0.097 $_{.015}$
	last dom. wo. ind.	0.657 $_{.002}$	0.852 $_{.014}$	0.599 $_{.004}$	0.811 $_{.024}$	0.722 $_{.005}$	0.907 $_{.01}$	0.256 $_{.006}$	0.133 $_{.012}$
XGBoost	all dom. w. ind.	0.664 $_{.005}$	0.859 $_{.004}$	0.61 $_{.013}$	0.828 $_{.003}$	0.754 $_{.006}$	0.9 $_{.019}$	0.194 $_{.018}$	0.111 $_{.02}$
	all dom. wo. ind.	0.633 $_{.035}$	0.831 $_{.024}$	0.568 $_{.033}$	0.778 $_{.031}$	0.718 $_{.033}$	0.881 $_{.028}$	0.194 $_{.054}$	0.12 $_{.042}$
	last dom. wo. ind.	0.664 $_{.01}$	0.824 $_{.023}$	0.599 $_{.016}$	0.758 $_{.054}$	0.733 $_{.009}$	0.887 $_{.016}$	0.199 $_{.024}$	0.167 $_{.011}$
LightGBM	all dom. w. ind.	0.65 $_{.009}$	0.842 $_{.024}$	0.594 $_{.008}$	0.8 $_{.024}$	0.738 $_{.008}$	0.908 $_{.008}$	0.198 $_{.009}$	0.093 $_{.016}$
	all dom. wo. ind.	0.625 $_{.02}$	0.832 $_{.019}$	0.561 $_{.018}$	0.773 $_{.018}$	0.706 $_{.009}$	0.888 $_{.01}$	0.199 $_{.009}$	0.097 $_{.005}$
	last dom. wo. ind.	0.629 $_{.02}$	0.797 $_{.009}$	0.542 $_{.031}$	0.72 $_{.028}$	0.686 $_{.006}$	0.852 $_{.018}$	0.224 $_{.011}$	0.149 $_{.016}$
Wild-Time ERM	all dom. w. ind.	0.627 $_{.036}$	0.821 $_{.032}$	0.525 $_{.052}$	0.771 $_{.044}$	0.688 $_{.028}$	0.877 $_{.025}$	0.263 $_{.054}$	0.108 $_{.017}$
	all dom. wo. ind.	0.582 $_{.028}$	0.803 $_{.01}$	0.519 $_{.028}$	0.746 $_{.018}$	0.673 $_{.02}$	0.862 $_{.008}$	0.255 $_{.019}$	0.104 $_{.015}$
	last dom. wo. ind.	0.587 $_{.026}$	0.784 $_{.03}$	0.528 $_{.034}$	0.74 $_{.011}$	0.666 $_{.018}$	0.843 $_{.02}$	0.323 $_{.038}$	0.19 $_{.025}$
Wild-Time SWA	all dom. w. ind.	0.627 $_{.047}$	0.824 $_{.003}$	0.534 $_{.069}$	0.777 $_{.014}$	0.685 $_{.036}$	0.873 $_{.015}$	0.274 $_{.053}$	0.11 $_{.007}$
	all dom. wo. ind.	0.588 $_{.025}$	0.802 $_{.007}$	0.53 $_{.029}$	0.748 $_{.008}$	0.681 $_{.03}$	0.863 $_{.008}$	0.262 $_{.036}$	0.109 $_{.01}$

Table 4: Comparison of Drift-Resilient TabPFN with various baselines and settings across the subset of **synthetic** datasets. Metrics include accuracy, F1, ROC, and ECE for both in-distribution (ID) and out-of-distribution (OOD) data, averaged over three initializations and reported with 95% confidence intervals. The best mean of each metric is marked in bold. Metric arrows indicate optimization direction.

Model	Variant	Acc. ↑		F1 ↑		ROC ↑		ECE ↓	
		OOD	ID	OOD	ID	OOD	ID	OOD	ID
TabPFN$_{\text{dist}}$	all dom. w. ind.	**0.754** $_{.032}$	0.959 $_{.011}$	**0.697** $_{.048}$	0.935 $_{.033}$	**0.844** $_{.03}$	0.987 $_{.002}$	**0.126** $_{.018}$	0.038 $_{.003}$
TabPFN$_{\text{base}}$	all dom. w. ind.	0.658 $_{.018}$	**0.963** $_{.006}$	0.567 $_{.015}$	0.935 $_{.02}$	0.749 $_{.017}$	0.986 $_{.007}$	0.164 $_{.014}$	**0.029** $_{.003}$
	all dom. wo. ind.	0.571 $_{.014}$	0.901 $_{.006}$	0.467 $_{.016}$	0.848 $_{.009}$	0.631 $_{.01}$	0.955 $_{.003}$	0.322 $_{.016}$	0.053 $_{.005}$
	last dom. wo. ind.	0.651 $_{.002}$	0.939 $_{.015}$	0.574 $_{.002}$	0.918 $_{.031}$	0.727 $_{.006}$	0.975 $_{.001}$	0.27 $_{.011}$	0.066 $_{.001}$
CatBoost	all dom. w. ind.	0.665 $_{.008}$	0.958 $_{.003}$	0.588 $_{.008}$	**0.94** $_{.009}$	0.73 $_{.01}$	**0.989** $_{.002}$	0.297 $_{.006}$	0.037 $_{.006}$
	all dom. wo. ind.	0.575 $_{.006}$	0.885 $_{.003}$	0.476 $_{.008}$	0.831 $_{.002}$	0.613 $_{.013}$	0.942 $_{.007}$	0.325 $_{.006}$	0.063 $_{.014}$
	last dom. wo. ind.	0.639 $_{.004}$	0.932 $_{.017}$	0.564 $_{.005}$	0.916 $_{.021}$	0.684 $_{.005}$	0.962 $_{.012}$	0.301 $_{.019}$	0.065 $_{.006}$
XGBoost	all dom. w. ind.	0.645 $_{.018}$	0.936 $_{.012}$	0.57 $_{.019}$	0.931 $_{.005}$	0.705 $_{.011}$	0.968 $_{.02}$	0.253 $_{.032}$	0.075 $_{.013}$
	all dom. wo. ind.	0.582 $_{.07}$	0.872 $_{.031}$	0.48 $_{.074}$	0.818 $_{.039}$	0.621 $_{.06}$	0.926 $_{.035}$	0.245 $_{.088}$	0.097 $_{.034}$
	last dom. wo. ind.	0.645 $_{.008}$	0.916 $_{.009}$	0.565 $_{.009}$	0.88 $_{.019}$	0.688 $_{.017}$	0.956 $_{.012}$	0.256 $_{.018}$	0.111 $_{.003}$
LightGBM	all dom. w. ind.	0.646 $_{.016}$	0.943 $_{.007}$	0.57 $_{.015}$	0.927 $_{.015}$	0.687 $_{.013}$	0.982 $_{.002}$	0.281 $_{.008}$	0.056 $_{.011}$
	all dom. wo. ind.	0.581 $_{.01}$	0.884 $_{.005}$	0.482 $_{.007}$	0.829 $_{.005}$	0.617 $_{.016}$	0.935 $_{.001}$	0.273 $_{.01}$	0.069 $_{.017}$
	last dom. wo. ind.	0.629 $_{.004}$	0.917 $_{.003}$	0.553 $_{.006}$	0.892 $_{.018}$	0.662 $_{.005}$	0.958 $_{.007}$	0.288 $_{.012}$	0.077 $_{.007}$
Wild-Time ERM	all dom. w. ind.	0.648 $_{.046}$	0.945 $_{.017}$	0.489 $_{.092}$	0.906 $_{.026}$	0.65 $_{.042}$	0.973 $_{.021}$	0.304 $_{.038}$	0.041 $_{.006}$
	all dom. wo. ind.	0.576 $_{.021}$	0.885 $_{.04}$	0.487 $_{.035}$	0.837 $_{.049}$	0.621 $_{.03}$	0.943 $_{.015}$	0.282 $_{.012}$	0.058 $_{.024}$
	last dom. wo. ind.	0.632 $_{.006}$	0.921 $_{.017}$	0.566 $_{.007}$	0.9 $_{.035}$	0.688 $_{.01}$	0.962 $_{.005}$	0.282 $_{.018}$	0.094 $_{.016}$
Wild-Time SWA	all dom. w. ind.	0.636 $_{.05}$	0.923 $_{.034}$	0.489 $_{.088}$	0.877 $_{.06}$	0.651 $_{.035}$	0.958 $_{.03}$	0.313 $_{.046}$	0.054 $_{.015}$
	all dom. wo. ind.	0.573 $_{.032}$	0.887 $_{.019}$	0.48 $_{.042}$	0.837 $_{.017}$	0.631 $_{.043}$	0.943 $_{.012}$	0.3 $_{.055}$	0.059 $_{.002}$

Table 5: Comparison of Drift-Resilient TabPFN with various baselines and settings across the subset of **real-world** datasets. Metrics include accuracy, F1, ROC, and ECE for both in-distribution (ID) and out-of-distribution (OOD) data, averaged over three initializations and reported with 95% confidence intervals. The best mean of each metric is marked in bold. Metric arrows indicate optimization direction.

Model	Variant	Acc. ↑		F1 ↑		ROC ↑		ECE ↓	
		OOD	ID	OOD	ID	OOD	ID	OOD	ID
TabPFN$_{\text{dist}}$	all dom. w. ind.	**0.736** $_{.007}$	0.814 $_{.014}$	**0.682** $_{.012}$	0.759 $_{.015}$	**0.822** $_{.01}$	0.887 $_{.006}$	**0.062** $_{.004}$	0.103 $_{.026}$
TabPFN$_{\text{base}}$	all dom. w. ind.	0.712 $_{.012}$	**0.822** $_{.013}$	0.661 $_{.022}$	**0.777** $_{.018}$	0.816 $_{.006}$	**0.894** $_{.013}$	0.083 $_{.001}$	0.097 $_{.007}$
	all dom. wo. ind.	0.704 $_{.01}$	0.813 $_{.023}$	0.668 $_{.014}$	0.764 $_{.03}$	0.82 $_{.006}$	0.882 $_{.011}$	0.106 $_{.019}$	**0.095** $_{.011}$
	last dom. wo. ind.	0.685 $_{.008}$	0.809 $_{.016}$	0.637 $_{.005}$	0.746 $_{.036}$	0.787 $_{.008}$	0.867 $_{.036}$	0.109 $_{.005}$	0.177 $_{.012}$
CatBoost	all dom. w. ind.	0.687 $_{.005}$	0.807 $_{.012}$	0.646 $_{.003}$	0.753 $_{.017}$	0.796 $_{.004}$	0.862 $_{.019}$	0.161 $_{.016}$	0.122 $_{.019}$
	all dom. wo. ind.	0.677 $_{.008}$	0.797 $_{.025}$	0.642 $_{.005}$	0.741 $_{.015}$	0.794 $_{.011}$	0.856 $_{.022}$	0.172 $_{.032}$	0.125 $_{.037}$
	last dom. wo. ind.	0.671 $_{.002}$	0.788 $_{.023}$	0.627 $_{.004}$	0.728 $_{.05}$	0.752 $_{.007}$	0.863 $_{.022}$	0.221 $_{.017}$	0.188 $_{.023}$
XGBoost	all dom. w. ind.	0.68 $_{.008}$	0.797 $_{.011}$	0.642 $_{.01}$	0.745 $_{.004}$	0.793 $_{.01}$	0.845 $_{.02}$	0.147 $_{.027}$	0.141 $_{.027}$
	all dom. wo. ind.	0.674 $_{.025}$	0.798 $_{.019}$	0.639 $_{.004}$	0.746 $_{.026}$	0.795 $_{.011}$	0.845 $_{.024}$	0.153 $_{.028}$	0.138 $_{.049}$
	last dom. wo. ind.	0.679 $_{.014}$	0.75 $_{.041}$	0.626 $_{.034}$	0.66 $_{.109}$	0.769 $_{.008}$	0.833 $_{.022}$	0.154 $_{.028}$	0.212 $_{.021}$
LightGBM	all dom. w. ind.	0.654 $_{.015}$	0.761 $_{.042}$	0.614 $_{.01}$	0.698 $_{.031}$	0.778 $_{.008}$	0.848 $_{.013}$	0.133 $_{.017}$	0.123 $_{.02}$
	all dom. wo. ind.	0.66 $_{.029}$	0.79 $_{.031}$	0.624 $_{.028}$	0.728 $_{.029}$	0.778 $_{.003}$	0.849 $_{.018}$	0.14 $_{.009}$	0.12 $_{.018}$
	last dom. wo. ind.	0.629 $_{.036}$	0.701 $_{.019}$	0.533 $_{.055}$	0.582 $_{.046}$	0.706 $_{.008}$	0.768 $_{.03}$	0.172 $_{.014}$	0.206 $_{.025}$
Wild-Time ERM	all dom. w. ind.	0.61 $_{.037}$	0.721 $_{.056}$	0.553 $_{.046}$	0.664 $_{.059}$	0.719 $_{.017}$	0.8 $_{.042}$	0.23 $_{.07}$	0.161 $_{.035}$
	all dom. wo. ind.	0.587 $_{.033}$	0.737 $_{.014}$	0.545 $_{.028}$	0.673 $_{.013}$	0.714 $_{.012}$	0.798 $_{.024}$	0.233 $_{.043}$	0.14 $_{.008}$
	last dom. wo. ind.	0.551 $_{.041}$	0.674 $_{.045}$	0.498 $_{.067}$	0.612 $_{.03}$	0.648 $_{.039}$	0.749 $_{.037}$	0.355 $_{.061}$	0.267 $_{.034}$
Wild-Time SWA	all dom. w. ind.	0.62 $_{.049}$	0.745 $_{.028}$	0.57 $_{.055}$	0.697 $_{.046}$	0.712 $_{.039}$	0.805 $_{.007}$	0.242 $_{.059}$	0.155 $_{.021}$
	all dom. wo. ind.	0.6 $_{.019}$	0.733 $_{.017}$	0.57 $_{.019}$	0.677 $_{.004}$	0.721 $_{.029}$	0.798 $_{.008}$	0.232 $_{.021}$	0.15 $_{.017}$

A.10.5 Quantitative Analysis: Critical Difference Diagrams. In addition to our quantitative results, we present a critical difference diagram for each of our evaluated metrics using the Wilcoxon-Holm method [60, 28, 22, 30].

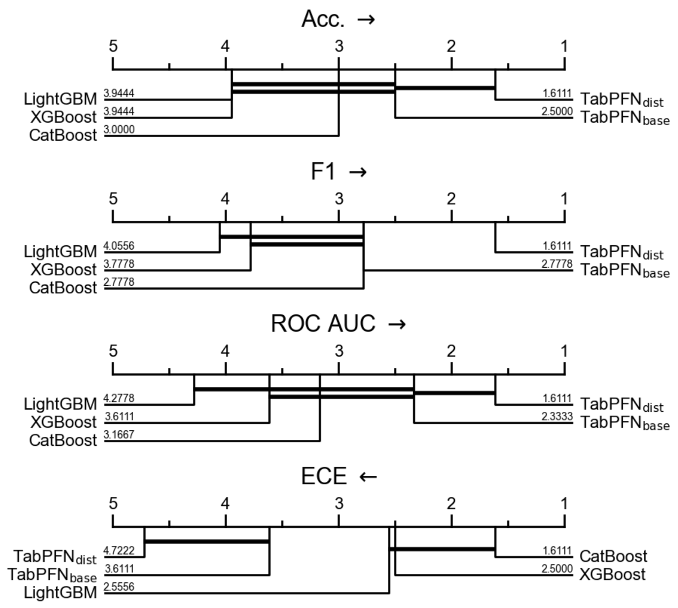


Figure 12: This figure presents critical difference diagrams of our evaluated metrics on OOD data, analyzed using the Wilcoxon-Holm method [60, 28, 22, 30] across the best performing OOD models evaluated in this work. The diagrams indicate significant differences of Drift-Resilient TabPFN against the tree-based methods. For the F1 metric our method shows significant differences against all top performing baselines. Arrows indicate optimization direction.

A.10.6 Quantitative Analysis: Comparison Against Wild-Time Methods. This section offers a quantitative evaluation of our model against methods found in the Wild-Time benchmark [61]. We focus exclusively on methods applicable to tabular data, omitting SimCLR and SwAV which are specifically designed for image datasets. Detailed explanations of these methods are available in Section A.11.

As a base model, we used a Multilayer Perceptron (MLP) optimized through Hyperparameter Optimization (HPO).

The findings of this quantitative comparison are presented in Table 6. Notably, none of the evaluated Wild-Time methods demonstrated performance equal to our approach or the other baseline methods on OOD data. This discrepancy is likely due to the small number of instances in the datasets used in our evaluations, which affects the generalization of these deep-learning techniques. Among the Wild-Time methods, SWA emerged as the most effective in handling OOD data.

Table 6: Comparison of Drift-Resilient TabPFN with the applicable baselines of the Wild-Time benchmark [61] across the combined **real-world** and **synthetic** datasets. Metrics include accuracy, F1, ROC, and ECE for both in-distribution (ID) and out-of-distribution (OOD) data, averaged over three initializations and reported with 95% confidence intervals. The best mean of each metric is marked in bold. Metric arrows indicate optimization direction.

Model	Variant	Acc. ↑		F1 ↑		ROC ↑		ECE ↓	
		OOD	ID	OOD	ID	OOD	ID	OOD	ID
TabPFN$_{\text{dist}}$	all dom. w. ind.	**0.744** $_{.018}$	**0.879** $_{.012}$	**0.689** $_{.028}$	**0.837** $_{.022}$	**0.832** $_{.018}$	**0.932** $_{.002}$	**0.091** $_{.006}$	**0.074** $_{.014}$
ERM	all dom. w. ind.	0.627 $_{.036}$	0.821 $_{.032}$	0.525 $_{.052}$	0.771 $_{.044}$	0.688 $_{.028}$	0.877 $_{.025}$	0.263 $_{.054}$	0.108 $_{.017}$
	all dom. wo. ind.	0.582 $_{.028}$	0.803 $_{.01}$	0.519 $_{.028}$	0.746 $_{.018}$	0.673 $_{.02}$	0.862 $_{.008}$	0.255 $_{.019}$	0.104 $_{.015}$
	last dom. wo. ind.	0.587 $_{.026}$	0.784 $_{.03}$	0.528 $_{.034}$	0.74 $_{.011}$	0.666 $_{.018}$	0.843 $_{.02}$	0.323 $_{.038}$	0.19 $_{.025}$
FT	all dom. w. ind.	0.51 $_{.031}$	0.645 $_{.014}$	0.422 $_{.046}$	0.559 $_{.033}$	0.594 $_{.027}$	0.693 $_{.039}$	0.357 $_{.013}$	0.256 $_{.007}$
	all dom. wo. ind.	0.544 $_{.038}$	0.677 $_{.034}$	0.481 $_{.046}$	0.604 $_{.03}$	0.65 $_{.025}$	0.756 $_{.031}$	0.33 $_{.03}$	0.231 $_{.061}$
EWC	all dom. w. ind.	0.498 $_{.026}$	0.621 $_{.034}$	0.405 $_{.038}$	0.522 $_{.042}$	0.569 $_{.027}$	0.668 $_{.006}$	0.342 $_{.006}$	0.239 $_{.034}$
	all dom. wo. ind.	0.518 $_{.023}$	0.686 $_{.045}$	0.456 $_{.032}$	0.617 $_{.051}$	0.603 $_{.012}$	0.74 $_{.026}$	0.309 $_{.048}$	0.195 $_{.005}$
SI	all dom. w. ind.	0.478 $_{.057}$	0.63 $_{.006}$	0.383 $_{.089}$	0.526 $_{.015}$	0.566 $_{.061}$	0.673 $_{.023}$	0.37 $_{.046}$	0.238 $_{.012}$
	all dom. wo. ind.	0.495 $_{.019}$	0.674 $_{.035}$	0.425 $_{.036}$	0.588 $_{.035}$	0.597 $_{.022}$	0.744 $_{.03}$	0.346 $_{.002}$	0.214 $_{.049}$
A-GEM	all dom. w. ind.	0.513 $_{.031}$	0.677 $_{.038}$	0.405 $_{.04}$	0.576 $_{.045}$	0.594 $_{.061}$	0.741 $_{.038}$	0.367 $_{.024}$	0.223 $_{.021}$
	all dom. wo. ind.	0.486 $_{.035}$	0.684 $_{.023}$	0.403 $_{.061}$	0.583 $_{.05}$	0.58 $_{.035}$	0.747 $_{.019}$	0.364 $_{.073}$	0.221 $_{.054}$
CORAL-T	all dom. w. ind.	0.579 $_{.051}$	0.777 $_{.02}$	0.481 $_{.085}$	0.714 $_{.041}$	0.637 $_{.036}$	0.837 $_{.016}$	0.252 $_{.058}$	0.143 $_{.025}$
	all dom. wo. ind.	0.569 $_{.032}$	0.771 $_{.035}$	0.495 $_{.029}$	0.702 $_{.039}$	0.643 $_{.016}$	0.837 $_{.021}$	0.232 $_{.027}$	0.15 $_{.003}$
GroupDRO-T	all dom. w. ind.	0.58 $_{.025}$	0.779 $_{.014}$	0.503 $_{.038}$	0.723 $_{.009}$	0.642 $_{.036}$	0.84 $_{.007}$	0.285 $_{.019}$	0.125 $_{.01}$
	all dom. wo. ind.	0.576 $_{.025}$	0.775 $_{.023}$	0.514 $_{.023}$	0.725 $_{.036}$	0.658 $_{.008}$	0.843 $_{.006}$	0.264 $_{.076}$	0.135 $_{.04}$
IRM-T	all dom. w. ind.	0.577 $_{.021}$	0.746 $_{.014}$	0.49 $_{.051}$	0.689 $_{.013}$	0.633 $_{.021}$	0.819 $_{.016}$	0.259 $_{.03}$	0.12 $_{.013}$
	all dom. wo. ind.	0.559 $_{.018}$	0.742 $_{.023}$	0.498 $_{.031}$	0.678 $_{.047}$	0.644 $_{.005}$	0.822 $_{.013}$	0.252 $_{.031}$	0.134 $_{.011}$
Mixup	all dom. w. ind.	0.617 $_{.028}$	0.8 $_{.023}$	0.521 $_{.016}$	0.702 $_{.008}$	0.681 $_{.029}$	0.859 $_{.021}$	0.239 $_{.014}$	0.14 $_{.007}$
	all dom. wo. ind.	0.574 $_{.034}$	0.782 $_{.027}$	0.492 $_{.025}$	0.688 $_{.013}$	0.68 $_{.025}$	0.85 $_{.022}$	0.212 $_{.021}$	0.142 $_{.017}$
LISA	all dom. w. ind.	0.621 $_{.041}$	0.822 $_{.033}$	0.517 $_{.079}$	0.76 $_{.063}$	0.679 $_{.005}$	0.881 $_{.012}$	0.272 $_{.046}$	0.116 $_{.02}$
	all dom. wo. ind.	0.583 $_{.013}$	0.804 $_{.015}$	0.512 $_{.009}$	0.739 $_{.024}$	0.672 $_{.025}$	0.859 $_{.024}$	0.258 $_{.025}$	0.108 $_{.023}$
SWA	all dom. w. ind.	0.627 $_{.047}$	0.824 $_{.003}$	0.534 $_{.069}$	0.777 $_{.014}$	0.685 $_{.036}$	0.873 $_{.015}$	0.274 $_{.053}$	0.11 $_{.007}$
	all dom. wo. ind.	0.588 $_{.025}$	0.802 $_{.007}$	0.53 $_{.029}$	0.748 $_{.008}$	0.681 $_{.03}$	0.863 $_{.008}$	0.262 $_{.036}$	0.109 $_{.01}$

A.11 Detailed Overview of Wild-Time Methods

A.11.1 Classical Supervised Learning.

Empirical Risk Minimization (ERM). ERM - a fundamental approach in supervised learning - focuses on minimizing the average loss over the training dataset. In Wild-Time ERM is defined as typical supervised learning without making use of any temporal information.

A.11.2 Continual Learning.

Fine-Tuning (FT). FT extends the ERM approach by training on the data of each successive temporal domain separately, allowing the model to adapt to new distributions but risking catastrophic forgetting of past tasks.

Elastic Weight Consolidation (EWC). EWC [35] counters catastrophic forgetting by adding a regularization term that constrains the changes to important model parameters, thus preserving knowledge from previous tasks.

Synaptic Intelligence (SI). SI [64] captures a synaptic strength metric over time for each model parameter. This metric is used as a regularizer to limit changes to important parameters during learning.

Averaged Gradient Episodic Memory (A-GEM). A-GEM [13] maintains a small episodic memory and computes gradients not just for the current task but also the average of the gradients over several past tasks stored in the episodic memory.

A.11.3 Temporally Invariant Learning.

Deep Correlation Alignment (Deep CORAL). Initially developed for domain adaptation, Deep CORAL [57] aims to align the second-order statistics of features between the source and target domains to minimize distribution shift. In the Wild-Time benchmark, this original purpose is modified to align features across different temporal domains within the training set, thus converting it into a DG method. For handling temporal shifts, it is extended into CORAL-T, which employs sliding windows to segment the data stream into temporal substreams, treating each as a separate domain for alignment. [61]

Group Distributionally Robust Optimization (GroupDRO). Originally designed at optimizing on the worst-performing group within the training data, GroupDRO [52] aims to learn a model that is robust across varying group distributions. In the Wild-Time benchmark, this method is adapted to the temporal context as GroupDRO-T. It utilizes sliding window-based segmentation to create temporal substreams, treating each as a separate group for distributionally robust optimization. [61]

Invariant Risk Minimization (IRM). IRM [4] aims to identify a data representation that is consistently predictive across different domains. In the context of Wild-Time, the method is adapted to temporal shifts and named IRM-T. It employs sliding window-based segmentation to create temporal substreams, which are then treated as individual domains for invariant risk minimization. [61]

Mixup. Mixup [65] is an interpolation-based data augmentation technique that creates new training examples by blending the features and labels of existing samples. This technique aims to enhance the model's ability to generalize across domains by diversifying the training data. It replaces the original training samples with these newly generated interpolations for more robust training.

Learning with Selective Augmentation (LISA). LISA [62], motivated by Mixup, employs selective interpolation to neutralize domain-specific information in the training data. It comes in two variants: intra-label LISA, which interpolates examples from different domains but having the same label, and intra-domain LISA, which interpolates examples within the same domain but having different labels. In Wild-Time, only intra-label LISA is used [61].

A.11.4 Self-Supervised Learning.

Simple Framework for Contrastive Learning of Visual Representations (SimCLR). SimCLR [15] employs contrastive learning to maximize the agreement between different augmentations of the same image, thereby enhancing the quality of learned visual representations. The approach benefits from learnable nonlinear transformations and optimized contrastive loss parameters.

Swapping Assignments between multiple Views of the same image (SwAV). SwAV [12] employs a clustering approach within the contrastive learning framework. It enforces consistency between cluster assignments across different augmentations of the same image. This obviates the need for pairwise feature comparisons, offering computational efficiency.

A.11.5 Bayesian Learning.

Stochastic Weight Averaging (SWA). SWA [31] averages multiple parameter values along the stochastic gradient descent (SGD) trajectory to improve in-distribution generalization. It operates with minimal computational overhead and aims to approximate the posterior distribution over model parameters, reflected in the flatness of the learned optima. [61]

A.12 Datasets

A.12.1 Validation Datasets.

Synthetic Datasets.

Dataset 1 (**Shifting Sin Classification**). The Shifting Sin Classification dataset is a synthetic, binary classification dataset of 1,500 instances evenly distributed across 10 domains. Each domain is differentiated by a unique shift in the offset of the sinusoid wave function, creating distinct decision boundaries for classification. The dataset contains two features corresponding to the x and y coordinates of each instance. Instances are labeled as 1 if they lie above the sine curve and 0 otherwise in their respective domains.

Dataset 2 (**Rotated Five Blobs**). The Rotated Five Blobs dataset is a synthetically generated dataset consisting of five blobs rotated sequentially counterclockwise -20° around a central point in each domain. It comprises two numerical features representing the x and y coordinates of each data point. Each blob consists of 40 samples, resulting in 200 samples per domain, for a total of 2,000 samples across the 10 domains represented.

Dataset 3 (**Moving Square**). The Moving Square is a synthetically generated dataset designed for a multi-class classification task. It encompasses two features and is divided into six domains, each containing 200 instances, thereby leading to a total of 1,200 samples. In the construction of this dataset, each of the four clusters—representing distinct classes—is initially located on one corner of a square. As we transition through the six domains, each cluster progressively moves along the edge of the square to the next corner.

Dataset 4 (**Moving Diagonal Line**). The Moving Diagonal Line dataset is a synthetic dataset, generated using the sklearn blobs function. It comprises 1,200 instances, divided across 6 domains, with each domain holding 200 instances. There are two features, corresponding to the x and y coordinate of each instance. In this dataset, there are two clusters, each representing a class, following a diagonal line that moves with each domain. Thereby, both clusters move in opposite directions along parallel diagonal next to each other. Each domain in this context represents different stages of the diagonal movement.

Real World Datasets.

Dataset 5 (**Indian Liver Patient Dataset**). The Indian Liver Patient Dataset (ILPD), referenced from Ramana and Venkateswarlu [50], is tailored for the binary classification task of identifying liver disease. It contains 583 records featuring 10 attributes, including age, gender, and diverse biochemical measurements. Originating from Andhra Pradesh, India, it comprises 416 liver patient records and 167 non-liver patient records. In our settings, every 5-year age interval is considered as an individual domain. The dataset, sourced from the UCI Machine Learning Repository, is geared towards supporting the diagnosis of liver disease.

Dataset 6 (**Istanbul Stock Exchange Returns**). The Istanbul Stock Exchange Returns dataset sourced from the UCI Machine Learning Repository provided by Akbilgic [3] includes 536 instances of returns from the Istanbul Stock Exchange and seven international indices from June 2009 to February 2011. The eight attributes represent various market return indices. The dataset is thereby used to predict the changes in the Istanbul stock exchange given all the other indices. The target was thereby discretized into 9 categories. The data was processed by dropping the USD column of the ISE and converting dates into a monthly domain feature, introducing a time-based distribution shift.

Dataset 7 (Diabetes 130-US Hospitals). The Diabetes 130-US Hospitals dataset provided by Strack et al. [56] encapsulates a decade (1999-2008) of diabetes care across 130 US hospitals, detailing 50 features related to patient demographics and hospitalization details. Criteria for inclusion are inpatient and diabetic encounters, with stays ranging from 1 to 14 days, where both lab tests and medications were administered. Features include patient identifiers, race, gender, age, admission type, duration of stay, attending physician's specialty, lab test counts, HbA1c results, diagnoses, medication details, and counts of healthcare visits prior to admission. The target of the prediction is to determine whether and in what time frame a patient will be readmitted. It is categorized into `<30 days`, `>30 days`, or `No` for no readmission. The original dataset, with 101,766 instances and 50 features, has been subsampled to 964 instances.

Dataset 8 (Airlines Delay). The Airlines Delay dataset, sourced from OpenML and provided by Bifet and Ikonomovska [9], contains 539,383 instances, each with 7 features. The task is to predict flight delays based on the scheduled departure information. Features include Airline, Flight, AirportFrom, AirportTo, DayOfWeek, and Time of departure. Notably, departure time, discretized to an interval of full hours, will be our distribution shift domain. The dataset was subsampled, thereby we sampled at most 60 samples per discrete time step. This resulted in 1,380 instances.

Dataset 9 (Pima Indians Diabetes). The Pima Indians Diabetes dataset from the National Institute of Diabetes and Digestive and Kidney Diseases, referenced by Smith et al. [54], contains 768 instances and 8 medical diagnostic features. These data represent female Pima Indian patients aged 21 or older. The task involves binary classification for predicting diabetes onset. We categorize each successive 2-year age interval as a separate domain, highlighting shifts in the dataset across age groups.

Dataset 10 (Diabetes Prediction through Questionaire). This dataset, collected from Sylhet Diabetes Hospital in Bangladesh and provided by Islam et al. [29], aims to predict early-stage diabetes. It comprises 520 instances and 16 features, representing symptoms and demographic information of patients. The task is binary classification, predicting whether a patient has diabetes or not. Age groups of every successive 5-year interval are considered as different domains, providing 14 age-based domains.

Dataset 11 (Room Occupancy Detection). The dataset is sourced from the UCI Machine Learning Repository and provided by Candanedo [11]. The preprocessed dataset, reduced to 1800 instances from the original 20,560, is used for binary classification of room occupancy based on Temperature, Humidity, Light, and CO2 levels. After removing 'date' and 'Id' features, 'day' and 'hour' were added. Thereby the 'day' was used as the temporal domain.

Dataset 12 (Sao Paulo Urban Traffic Behavior). The Sao Paulo Urban Traffic Behavior dataset, sourced from the UCI Machine Learning Repository provided by Ferreira et al. [19], captures records of urban traffic behavior in Sao Paulo, Brazil, from December 14 to 18, 2009, and tries to predict the slowness in traffic. The dataset contains 135 instances each with 18 attributes.

Attributes are various traffic indicators such as Hour, Immobilized bus, Broken Truck, Vehicle excess, Accident victim, and more. We have discretized the target "Slowness in traffic (%)" into intervals of 7.5 percent. Each day represents a different domain, thus introducing a time-based shift in the dataset.

A.12.2 Test Datasets.

Synthetic Datasets.

Dataset 13 (Rotated Two Moons). The Rotated Two Moons dataset as stated by Nasery et al. [43] and Bai et al. [5], is a derivative of the 2-entangled moons dataset and includes 220 instances in each of the 10 domains. Each domain is differentiated by counter-clockwise rotations of 18°, resulting in a rotation of $18 \cdot i°$ in domain i. Also, the distribution of a subset of the instances varies across domains.

Dataset 14 (Intersecting Blobs). The Intersecting Blobs Dataset is a synthetically created set tailored for complex, binary classification tasks. The dataset contains two features per sample and 120 samples per domain in a total of 14 domains. Each domain comprises three classes, each represented by 40 samples appearing as blobs. These blobs move and vary, getting quite close to one another, almost intersecting. This dynamic creates sudden shifts in decision boundaries, increasing the difficulty and complexity of the classification tasks. The continuous shifts in blobs' positions across domains are implemented by adjusting their centers and standard deviations.

Dataset 15 (Binary Label Shift). The Binary Label Shift Dataset is a synthetic dataset tailored for binary classification in an environment with prior probability shifts. The dataset consists of 10 unique domains, with each containing 200 samples and each sample consisting of two features. The key feature of this dataset is the systematic manipulation of class probabilities across domains. This is realized by starting with a high probability of 0.95 for one class in the first domain, which progressively diminishes to 0.05 in the final domain. Simultaneously, the probability for the other class increases from 0.05 to 0.95, following the opposite direction. This dynamic essentially portrays a "fade out and fade in" pattern of the classes across the domains, representing the prior probability shift.

Dataset 16 (Rotating Hyperplane). The Rotating Hyperplane binary classification dataset is artificially generated based on the package `scikit-multiflow` provided by Montiel et al. [39]. It consists of five features, of which three shift over time. The dataset is divided into 15 domains of 100 instances each, providing a total of 1500 samples. As the name implies, the key aspect of this dataset is about a rotating hyperplane. In other words, the decision boundary - or hyperplane - shifts as we navigate from one domain to the next, making the classification task increasingly difficult.

Dataset 17 (RandomRBF Drift). The RandomRBF Drift binary classification dataset is synthetically generated based on the package `scikit-multiflow` provided by Montiel et al. [39]. This dataset has been constructed by introducing drifts in the data using Radial Basis Functions. It is characterized by the motion of cluster centroids, which are responsible for creating data drift. This movement can be visualized as clusters that change their positions, altering the distribution of data over time. The dataset consists of 8 features, 15 domains with 100 samples in each domain, for a total of 1,500 samples.

Dataset 18 (Rotating Segments). The Rotating Segments dataset is a synthetically generated dataset tailored for a binary classification task. The dataset visualizes a circle partitioned into four segments, similar to the slices of a cake. The data points are thereby labeled alternately. As we traverse through the ten domains, these segments undergo a rotation. Each domain contains 150 samples, accumulating to 1500 samples in total.

Dataset 19 (Sliding Circle). The Sliding Circle dataset is a synthetically generated dataset and represents a binary classification task. It comprises two features, the dataset is partitioned into ten domains, each possessing 200 samples, summing up to a total of 2,000 samples. The unique aspect of this dataset is its visual representation: a smaller circle slides around the inner perimeter of a larger circle. Within the larger circle, points are classified based on whether they lie inside the smaller sliding circle or outside of it. As we traverse through the ten domains, the position of the smaller circle changes, causing a shift in the classification of the points.

Dataset 20 (Shifting Two Spirals). The Shifting Two Spirals dataset is designed for binary classification tasks. The dataset visually represents two intertwined spirals. As we move from one domain to the next, the classification boundary in the spirals evolves. Specifically, one spiral gradually transitions its labels from the center towards the outer end, while the other spiral does the exact opposite, transitioning its labels from the outer end towards the center. This dynamic showcases a fascinating interplay of domain adaptation across the ten domains. Each domain has 200 samples, with 100 samples from each spiral, summing up to a total of 2,000 samples.

Real World Datasets.

Dataset 21 (Free Light Chain Mortality). The free light chain dataset comprises data from a study investigating the link between serum free light chain (FLC) and mortality. It includes 1125 stratified samples per domain and target from an original pool of 7,874, featuring residents of Olmsted County, Minnesota aged 50 or more.

The task involves predicting mortality based on 9 features, which include age, sex, year of blood sample, FLC portions (kappa and lambda), the FLC group, serum creatinine, MGUS diagnosis, and days from enrollment to death or last follow-up. The feature 'chapter' was omitted because it is direct information on whether someone died or not. We treat "sample.yr", the year in which the sample was taken, as a domain, resulting in a total number of 9 domains. We suspect that both the measurements themselves and the selection of participants have changed over time. The dataset is sourced from studies by Dispenzieri et al. [18] and Kyle et al. [36].

Dataset 22 (Electricity). This dataset as used by Harries [26] and Gama et al. [20] contains electricity demand in the Australian New South Wales Electricity Market. Since the prices are not fixed, they fluctuate depending on supply and demand. It has 45,312 instances and 5 features. For reproducibility, the dataset includes additional features such as the New South Wales electricity price, which was used to form the target class according to the original paper, and the Victoria electricity price, which was not used in the original paper. The dataset features the demand of electricity in to provinces as well as the transfer between those for periods of 30 minutes. The task is a binary classification, which requires predicting whether the price in the current period is higher or lower than the average of the last 24 hours. The dataset contains seasonal data due to varying demand for electricity. The effects of long-term price trends on the class label are removed by the 24-hour moving average. We consider one-week periods as a single domain. To comply with TabPFN sequence length limits, we keep only two hourly intervals for each day and subsample 15 weeks of the whole time period.

Dataset 23 (Absenteeism at Work). The Absenteeism at Work dataset, sourced from the UCI Machine Learning Repository provided by Martiniano and Ferreira [38], comprises 740 instances across 21 features. It captures various attributes of employees and their working conditions, such as the reason for absence, day of the week, seasons, distance to work, and more, with the target feature being the absenteeism time in hours which was discredited into 4-quantiles.

The primary shift in this dataset is supposed to be seasonal. Thereby each consecutive season is treated as a different domain. Furthermore, no significant preprocessing or subsampling was required due to the manageable size of the dataset.

Dataset 24 (Heart Disease Dataset). The Heart Disease dataset, sourced from the UCI Machine Learning Repository and provided by Janosi et al. [32], targets the classification task of identifying the presence (values 1,2,3,4) or absence (value 0) of heart disease in patients. We treat the task as a binary classification, predicting only whether a patient has heart disease or not. It includes 303 instances, each with 13 health-related features such as age, sex, resting blood pressure, cholesterol levels, etc. In this context, each consecutive 4-year age interval is viewed as a single domain. Rows with missing values have been omitted.

Dataset 25 (**Parking Birmingham**). The Parking Birmingham dataset, provided by Stolfi [55] via the UCI Machine Learning Repository, initially comprises the capacity and occupancy rates(target) of multiple car parks. We have processed it to only include the car park labeled 'Others-CCCPS133', thereby reducing the number of instances to 1,294 from 35,717. The original 'LastUpdated' attribute has been transformed into 'day', 'week', and 'month' features, with the 'week' serving as the temporal domain. The 'Occupancy' target, originally an absolute figure, is now presented as a percentage of the parking space utilization, discretized into 25% intervals.

Dataset 26 (**Ames Housing Prices**). The Ames Housing Dataset, curated by De Cock [16], consists of 1460 instances each with 79 features detailing various aspects of residential homes in Ames, Iowa. We use only the training portion of this dataset due to the absence of ground truth targets in the test data. We have discretized the house price, which was originally a continuous variable, into a categorical variable. This transformation was achieved by partitioning the price data into intervals. Specifically, the intervals are defined as: $[0, 125k]$, $(125k, 300k]$, $(300k, \infty)$. The task is to predict the price range of a home based on its features. We treat the 'YearBuilt' attribute, divided into 15-year periods, as our domain to capture changes in housing trends over time. It is to be expected that the data set shows temporal shifts, as the price distribution between older and more modern houses differs.

Dataset 27 (**Folktables US Census**). The folktables datasets, derived from the US Census Public Use Microdata Sample (PUMS) data and published by Ding et al. [17] consists of demographic and socioeconomic data between 2015 and 2021. Each year within this timeframe represents a distinct domain for a series of tasks: ACSIncome, ACSPublicCoverage, and ACSEmployment. We purposefully limited our focus to the state Maryland, to limit the shifts to the temporal domain. The size of the datasets necessitated a stratified subsample for the target per year to reach a total of approximately 1300 instances per dataset and meet the TabPFN model requirements. This subsampling ensured a representative yet computationally manageable sample.

The tasks are as follows:

- **ACSIncome**: The task predicts whether an individual's income surpasses $50,000, narrowing the ACS PUMS data to individuals over 16 who reported at least one working hour per week in the past year and a minimum income of $100. The task consists of 10 features accross the 7 domains.
- **ACSPublicCoverage**: The objective is to predict if an individual is covered by public health insurance. The dataset is filtered to include only individuals under 65 with an income below $30,000, focusing the prediction on low-income individuals ineligible for Medicare. The task consists of 19 features accross the 7 domains.
- **ACSEmployment**: The task is to predict whether an individual is employed. The dataset is filtered to include only individuals between 16 and 90 years of age. The task consists of 16 features accross the 7 domains.

Dataset 28 (**Chess**). The Chess dataset published by Žliobaitė [66] is derived from recorded chess games, aiming to predict game outcomes (draw, lost, won) through a multi-class classification task. It consists of nine features which provide insights into the game details and player attributes, including move sequences, player side (white or black), current rating, opponent's rating, type of game, speed, and the date of the game (broken down into year, month, and day). The dataset is segmented into 27 domains, where each domain represents 20 consecutive games. This segmentation helps capture the evolution of a player's progress over time. The dataset, in its entirety, holds 533 instances. It has been constructed based on games played between 7 December 2007 and 26 March 2010.

A.13 Hyperparameter Search Spaces

Table 7: Hyperparameter search spaces we used for Wild-time baselines.

Underlying MLP (+ ERM, A-GEM, FT)	
Parameter	Values
train_update_iter	500, 1000, 2000, 3000, 4000, 5000, 6000, 7000
lr	$e^{\mathcal{U}(-14,-4)}$
use_scheduler	True, False
ft_scheduler_gamma	$\mathcal{U}(0.9, 1.0)$
weight_decay	$0.0, 1e-5, 1e-2$
early_stopping	True, False
early_stop_holdout	$0.1, 0.15, 0.2$
early_stop_patience	$\mathcal{U}\{10, 11, \ldots, 30\}$
LISA	
Parameter	Values
mix_alpha	$e^{\mathcal{U}(0.5,4.0)}$
cut_mix	True, False
Mixup	
Parameter	Values
mix_alpha	$e^{\mathcal{U}(-5,0)}$
EWC	
Parameter	Values
gamma	$e^{\mathcal{U}(1.0,2.0)}$
ewc_lambda	$e^{\mathcal{U}(0.5,2.0)}$
GroupDRO-T	
Parameter	Values
group_size	$\mathcal{U}\{1, 2, \ldots, 6\}$
non_overlapping	True, False
group_loss_adjustments	None, 0.1, 0.5, 1.0
group_loss_btl	True, False
SWA	
Parameter	Values
swa_portion	$\mathcal{U}(0.5, 0.9)$
swa_lr_factor	$\mathcal{U}\{1, 2, \ldots, 6\}$
IRM-T	
Parameter	Values
group_size	$\mathcal{U}\{1, 2, \ldots, 6\}$
non_overlapping	True, False
irm_lambda	$\mathcal{U}\{1, 2, \ldots, 100\}$
irm_penalty_anneal_iters	0, 250, 500, 750, 100
SI	
Parameter	Values
si_c	$\mathcal{U}(0.05, 0.2)$
epsilon	$\mathcal{U}(0.0005, 0.002)$
CORAL-T	
Parameter	Values
group_size	$\mathcal{U}\{1, 2, \ldots, 6\}$
non_overlapping	True, False
coral_lambda	$\mathcal{U}(0.1, 1.0)$

Table 8: Hyperparameter search spaces we used for our baselines.

XGBoost	
Parameter	Values
learning_rate	$e^{\mathcal{U}(-7,0)}$
max_depth	$\mathcal{U}\{1,2,\ldots,10\}$
subsample	$\mathcal{U}(0.2,1)$
colsample_bytree	$\mathcal{U}(0.2,1)$
colsample_bylevel	$\mathcal{U}(0.2,1)$
min_child_weight	$e^{\mathcal{U}(-16,5)}$
alpha	$e^{\mathcal{U}(-16,2)}$
lambda	$e^{\mathcal{U}(-16,2)}$
gamma	$e^{\mathcal{U}(-16,2)}$
n_estimators	$\mathcal{U}\{100,101,\ldots,4000\}$
LightGBM	
Parameter	Values
num_leaves	$\mathcal{U}\{5,6,\ldots,50\}$
max_depth	$\mathcal{U}\{3,4,\ldots,20\}$
learning_rate	$e^{\mathcal{U}(-3,0)}$
n_estimators	$\mathcal{U}\{50,51,\ldots,2000\}$
min_child_weight	1e-5, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4
subsample	$\mathcal{U}(0.2,0.8)$
colsample_bytree	$\mathcal{U}(0.2,0.8)$
reg_alpha	0, 1e-1, 1, 2, 5, 7, 10, 50, 100
reg_lambda	0, 1e-1, 1, 5, 10, 20, 50, 100
CatBoost	
Parameter	Values
learning_rate	$e^{\mathcal{U}(-5,0)}$
random_strength	$\mathcal{U}\{1,2,\ldots,20\}$
l2_leaf_reg	$e^{\mathcal{U}(0,log(10))}$
bagging_temperature	$\mathcal{U}(0.0,1.0)$
leaf_estimation_iterations	$\mathcal{U}\{1,2,\ldots,20\}$
iterations	$\mathcal{U}\{100,101,\ldots,4000\}$

Table 9: Preprocessing search spaces for Drift-Resilient TabPFN and TabPFN-base.

Parameter	**Search Space**
model_type	single
N_ensemble_configurations	16, None
preprocess_transforms	See Table 10
softmax_temperature	log(0.75), log(0.8), log(0.9), log(0.95)
use_poly_features	True, False
max_poly_features	50
remove_outliers	-1, 7.0, 9.0, 12.0
add_fingerprint_features	True, False
subsample_samples	0.9, 0.99, -1

Table 10: Parameters and values for enumerate_preprocess_transforms function.

Parameter	**Values**
names	["safepower"], ["quantile_uni_coarse"], ["quantile_norm_coarse"], ["adaptive"], ["norm_and_kdi"], ["quantile_uni"], ["none"], ["robust"], ["kdi_uni"], ["kdi_alpha_0.3"], ["kdi_alpha_3.0"], ["safepower", "quantile_uni"], ["kdi", "quantile_uni"], ["none", "power"]
categorical_name	["numeric", "ordinal_very_common_categories_shuffled", "onehot", "none"]
append_original	[True, False]
subsample_features	[-1, 0.99, 0.95, 0.9]
global_transformer	[None, "svd"]