Universiteit Leiden

Master Computer Science

Tabular Foundation Models in Risk Management: Navigating Sparse, Noisy, and Limited Data

Name:	[Hanwen Huang]
Student ID:	[S4261100]
Date:	[01/05/2026]
Specialisation:	[Artificial Intelligence]
1st supervisor:	[Marc Hilbert]
2nd supervisor:	[Yingjie Fan]

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Credit risk management relies on application information, converting model predictions into automated decisions through modeling. However, in actual business operations, data sparsity trends: including missing values and data quality issues, these questions emerge, diminishing the predictive capability of models. Based on data provided by **De Lage Landen International B.V.**, this study utilizes actual credit application data from the company's North American region to construct an end-to-end evaluation and deployment framework. This study assesses the generalization capability of the base model from four dimensions: randomness, sample size, label noise, and missing disturbance. Building upon this foundation, this study integrates global importance screening with attribution fusion techniques, mapping continuous prediction probabilities onto three decision intervals. Simultaneously, by leveraging surrogate models and shallow tree distillation methods, IF-ELSE rule fragments are generated that can be directly integrated into operational systems. Experimental results demonstrate that the TabPFN model maintains stability even under conditions of small sample sizes and noisy labels, with predictive capabilities comparable to traditional machine learning approaches. Concurrently, the risk control decision strategy achieves high automation coverage, reducing manual review ratios to 43.3%. This framework provides a reliable evaluation and interpretable deployment solution for tabular foundational models in highly sparse financial risk control environments.

Contents

1 Introduction

Credit risk management is a core function of financial services, protecting an organization's assets and cash flows by identifying, assessing, and mitigating potential risks associated with extending credit to customers or counterparties. Its primary activities involve predicting the likelihood of borrower default and supporting automated credit decisions. Credit risk represents the potential loss arising from a borrower's refusal or inability to repay outstanding balances in full and on time. In modern credit risk management systems, the objective is to optimize automated decision-making within specified risk thresholds, thereby reducing manual intervention and enhancing operational efficiency[1]. More specifically, the system generates automated decisions based on existing application information: Accept, Refer, and Decline. Subsequently, the manual review phase consolidated the results into two final categories: **'Accept'** and **'Decline'**. From a modeling perspective, this task can be formalized as a supervised learning binary classification problem[2].

Unlike fields such as natural language processing (NLP) or computer vision, credit risk management primarily relies on tabular data, including customer-submitted application information, transaction histories, credit records, and financial metrics. While this data format is ubiquitous in finance, it lags significantly behind unstructured data in terms of available scale and model performance[3]. In practical credit risk scenarios, tabular data faces the following challenges:

- **Data Sparsity:** Limited historical information for new customers often results in substantial missing values for key variables.
- **Data Noise:** Human errors during application entry or system processing introduce measurement uncertainties.
- **Limited Data:** Determining sufficient data volume for robust model generalization remains challenging across varying business conditions.

These factors lead to unstable model performance, poor transferability, and a heavy reliance on extensive feature engineering and manual intervention to achieve usable levels.

Early credit risk prediction methods primarily relied on traditional machine learning models such as logistic regression and decision trees. These approaches achieved risk stratification through feature selection and scored cards, for example by constructing credit scoring models using WOE (Weight of Evidence) and Information Value-based filtering[4]. Such methods struggle to capture complex nonlinear relationships between features and exhibit limited performance on high-dimensional sparse data.

In recent years, researchers have shifted their focus toward more powerful ensemble learning methods, particularly the practical application of gradient-boosted decision trees (GBDT) approaches. Algorithms like XGBoost[5], combined with feature engineering, have become mainstream methods for credit risk modeling. Furthermore, by incorporating regularization, automated missing value handling, and categorical feature encoding,

these methods have significantly enhanced model predictive capabilities and robustness against data noise. Simultaneously, deep learning approaches like MLP[6] and TabNet[7] architectures have been explored for tabular data modeling. By incorporating attention mechanisms and representation learning, these methods aim to automatically learn feature interactions, reducing reliance on manual feature engineering. However, on tabular datasets, deep learning methods still struggle to consistently outperform well-tuned GBDT models[8, 9].

Despite significant progress made with the aforementioned methods, two core challenges remain when dealing with real-world business data:

Challenge 1: Performance degradation under highly sparse data. Most existing models assume relatively complete data, underestimating the impact of extreme sparsity on model performance. In such extreme conditions, the issue is not that models cannot handle missing values, but rather that they struggle to extract sufficient discriminative features from the limited available information when the missing rate is high[10].

More specifically, if three out of the five most critical fields are missing in most samples, a large number of samples become similar in feature space. The model then relies more heavily on the few remaining fields or the "missingness itself" as proxy signals. Consequently, predictions tend to cluster closer to the overall average risk, leading to a significant drop in accuracy. This performance degradation with increasing missingness rate is evident across comparative experiments with different missingness proportions. In this context, the completeness, accuracy, and consistency of data directly impact model reliability. In this environment, the integrity, accuracy, and consistency of data will directly impact the reliability of the model[11].

Challenge 2: The gap between model predictions and operational deployment. Even when models demonstrate strong predictive performance during evaluation, translating these predictions into interpretable, auditable, and deployable business rules and procedures remains essential for real-world applications. Furthermore, AI applications in finance demand not only accuracy but also consideration of regulatory compliance and privacy, alongside transparency and auditability. Therefore, developing a unified framework that integrates predictive capability with interpretability remains a critical research direction in credit risk modeling[12].

These two challenges raise deeper scientific questions: How can models maintain stable performance under extreme data conditions? How can predictions from complex models be transformed into actionable business insights? More specifically, this leads to the research questions addressed in this study:

How do tabular foundation models (e.g., TabPFN) perform in real-world financial risk management tasks under varying data scales and structural conditions (sparsity, noise, data constraints)? What are the potential reasons for their strengths or weaknesses?

Moreover, this research question is broken down into three sub-problems:

- **RQ1:** What level of predictive performance and stability can the baseline model achieve on the current dataset?

  Specifically, evaluate the performance of traditional machine learning models such as

decision trees, KNN, random forests, and XGBoost using a unified data processing workflow and evaluation metrics.

- **RQ2:** How can TabPFN be effectively implemented and evaluated under this dataset and business constraints?

  Focus on the engineering implementation and evaluation methods of TabPFN, specifically including: training configurations (how to handle high-cardinality features, missing values, and data encoding), inference efficiency (computational overhead on datasets of this scale), evaluation metrics (e.g., Loss, Brier Score, AUC), and interpretable outputs (how to deploy model predictions into actual business operations).

- **RQ3:** How do TabPFN and baseline models differ in performance, robustness, and generalization under varying data scales and conditions?

  This question is addressed through systematic comparative experiments. Specifically, these include: variations in data scale (from 10% to 100% subsampling), sensitivity to label noise (introducing 0%-20% noise into training labels), and sensitivity to feature missingness (performance degradation as the missing rate increases further).

These three research questions form a progressive relationship: RQ1 establishes a baseline, RQ2 explores new methods, and RQ3 conducts systematic comparisons.

Based on the above analysis, this study proposes a credit risk assessment framework for sparse data, aiming to enhance predictive stability and business interpretability under extreme data conditions. This framework incorporates three core mechanisms to address the aforementioned limitations:

- **Unified Data Processing Workflow:** Establish a standardized preprocessing pipeline encompassing bucketized mapping, high-cardinality category dimensionality reduction, missing value indicator variable construction, and target encoding to ensure comparability across models under identical data conditions.

- **Multi-dimensional Robustness Testing:** Systematically evaluate model stability and generalization boundaries across four dimensions: random seed variation, subsampling of data scale, label noise injection, and feature dropout rate manipulation.

- **Explainability-Driven Deployment Strategy:** Construct diversion strategies based on model probability outputs, optimize manual review rates through threshold search, and deploy actionable business rules via model distillation.

This study is an applied research project conducted in collaboration with **De Lage Landen International B.V.** (Hereafter referred to as DLL), a wholly-owned subsidiary of Rabobank Group and an international supplier financing organization with over €47 billion in assets. As a global leader in asset financing services, DLL provides end-to-end financing solutions across the entire asset lifecycle in multiple verticals including agriculture, food, healthcare, clean technology, construction, transportation, industrial, office equipment, and technology. These solutions encompass commercial financing, retail financing, and used equipment financing. Within this framework, this study focuses

on leveraging artificial intelligence to optimize DLL's automated credit decision-making processes in North America. By utilizing real business data, this study explores the application potential of Tabular Foundation Models under challenging conditions such as high missing values and noisy data, aiming to reduce reliance on manual review while enhancing decision efficiency and consistency.

The remainder of this thesis is organized as follows: Chapter 2 reviews prior work on tabular data machine learning, foundational tabular models, credit risk prediction methods, and explainable AI; Chapter 3 describes the specific details and structural analysis of the applied dataset; Chapter 4 introduces the experimental framework, including data processing workflows, model configurations, evaluation metrics, and model training methods; Chapter 5 reports experimental results for baseline ML models and TabPFN, including robustness analysis and interpretability analysis; Chapter 6 discusses explainable AI and applies the model to real-world business decision-making; Chapter 7 discusses key findings, analyzes the framework's limitations, and explores potential extension directions; Chapter 8 concludes the paper.

2 Related Work

This chapter reviews the literature relevant to the sub-research, covering four directions: machine learning methods in risk learning, tabular data modeling, foundational models for tabular data, and explainable artificial intelligence in the financial domain.

2.1 Traditional Machine Learning Methods in Credit Risk Management

Credit risk prediction is one of the core tasks of financial institutions. Its primary function is to forecast the probability of default based on applicant information, which is used for credit decision-making[1]. This section reviews the evolution of credit risk modelling, progressing from classical statistical approaches to modern machine learning methods.

2.1.1 Starting from traditional credit scoring methods

Early credit scoring systems can be traced back to the 1950s, based on Altman's Z-score model and subsequent logistic regression methods[13, 14]. Using scorecards as the core framework, these methods construct an interpretable linear model through Weight of Evidence (WOE) coding and Information Value (IV)[4]. The core advantage of these methods lies in their simplicity, ease of deployment, and regulatory compliance, and remain widely used in banks and financial institutions to this day.

However, traditional scorecards have significant shortcomings. Their assumptions are built upon linear correlations, making it difficult to capture nonlinear mapping relationships. At the same time, reliance on manual feature engineering leads to lengthy model development cycles and high maintenance costs[15]. Therefore, researchers have gradually turned to machine learning.

2.1.2 Modern Approaches to GBDT

Gradient Boosting Decision Trees (GBDT) have become the mainstream approach for credit risk modeling. XGBoost significantly enhances efficiency by incorporating regularization and parallel computing[5]; Subsequently, LightGBM introduced a histogram-based algorithm and a leaf-first growth strategy, further reducing computational overhead[16]; CatBoost proposed an ordered target encoding approach for categorical features, reducing the prediction offset issue[17]. Multiple studies have demonstrated that GBDT methods combined with appropriate feature engineering typically achieve optimal performance[18, 19]. Lessmann compared 41 credit scoring datasets and found that it outperformed traditional statistical methods on most datasets[20].Brown and Mues' research on imbalanced credit score data also confirms this conclusion[21].

2.1.3 Exploration and Limitations of Deep Learning

In recent years, researchers have begun exploring the application of deep learning in the field of credit risk. Clements proposed using sequential deep learning models to process longitudinal credit data[22]; Kvamme applied recurrent neural networks to credit default prediction. However, these credit data typically lack the spatial or temporal structure that deep learning excels at[23]; secondly, on small to medium-sized datasets, deep models are prone to overfitting[24].

2.1.4 The Unique Characteristics of Credit Data

Credit risk data possesses unique quality characteristics. Mogus analyzed data quality issues from four dimensions: completeness, accuracy, consistency, and timeliness, pointing out that missing values and noise are common characteristics of credit data[25]. In practical business scenarios, new customers often lack historical credit records, and the coverage of external data sources varies, resulting in a highly sparse feature matrix. Additionally, certain data privacy protection requirements (such as GDPR) necessitate anonymizing or binning sensitive fields, further complicating the modeling process.

2.2 Machine Learning for Tabular Data

Tabular data is the primary form of structured data and is widely used in the financial sector. Tabular data possesses the following characteristics[9]:

1. Features in tables differ: Typically, numerical, categorical, and ordinal features are mixed within tables, and there is a lack of inherent spatial or sequential relationships between features.

2. Sample size scale: Typical tabular datasets range from thousands to hundreds of thousands of records, demanding higher sample efficiency from the model.

3. The objective function in tabular data exhibits irregularity: the model learns a mapping relationship between inputs and outputs, yet the decision boundary in tabular data presents a non-smooth, non-monotonic shape. This inductive bias conflicts with the structure of tabular data.

Based on the above characteristics, the choice of modeling methods is influenced; however, researchers have primarily explore the application of deep learning to tabular data, with key directions including: Embedding maps categorical features into a continuous space[6]; Subsequently, the introduction of TabNet enabled sparse attention to achieve interpretable feature selection[7]; Subsequently, the emergence of the Transformer architecture introduced novel approaches to such tasks. TabTransformer applies the Transformer encoder to categorical features, leveraging self-attention mechanisms to learn contextual relationships among features[26]; The FT-Transformer further incorporates numerical features to achieve unified modeling of heterogeneous features[27].

Research in another direction focuses on incorporating the inductive bias of tree models into neural networks. NODE (Neural Oblivious Decision Ensembles) employs differentiable soft decision trees, enabling the tree structure to be optimized via gradient descent[28]. DeepGBM distills knowledge from trained GBDTs into neural networks[29].

However, multiple benchmark studies indicate that while these methods may outperform GBDT in specific scenarios, none consistently outperform it across most datasets[30]. This dilemma raised the question of whether models could learn tabular data through large-scale pre-training, thereby motivating the development of foundation models for tabular data.

2.3 Foundation Tabular Models

Foundation Models The concept originates from the field of natural language processing, referring to a general-purpose model that is pre-trained on large-scale data and adaptable to various downstream tasks[31]. As large language models like GPT demonstrate powerful contextual learning capabilities, researchers have begun exploring the application of this paradigm to the field of tabular data.

2.3.1 From In-Context Learning to TabPFN

In-Context Learning is one of the core capabilities of large language models, enabling the model to accomplish new tasks by providing only a small number of examples in the input, without requiring parameter updates[32]. The core challenge of applying ICL to tabular data lies in enabling the model to learn a complete prediction algorithm from limited context samples. Müller et al. proposed Prior-Data Fitted Networks (PFN) as a solution[33]: during training, the model learns a mapping from training to test predictions by learning on a large number of synthetic tasks sampled from the prior distribution; during inference, training data from new tasks is directly input as context into the model, enabling predictions without requiring model updates.

2.3.2 TabPFN Architecture and Capabilities

TabPFN applies the PFN framework to table classification, forming a foundational model for tables[34]. The core content encompasses three aspects:

1. Synthetic Data Pre-training: TabPFN is pre-trained on millions of synthetic classification tasks sampled from structured prior knowledge.

2. Transformer architecture: The model employs a 12-layer Transformer encoder to uniformly encode training samples (features + labels) and testing samples (features only) into sequences. It leverages self-attention mechanisms to capture relationships between samples, directly outputting the posterior prediction distribution for testing samples.

3. Zero-shot reasoning: When encountering new tasks, TabPFN requires no fine-tuning or hyperparameter search. It processes training data as contextual input and completes predictions in approximately one second.

TabPFN achieves state-of-the-art performance across 95 benchmark datasets, demonstrating particularly strong advantages when the sample size is less than 10,000 and the number of features is below 100.

The success of TabPFN has garnered widespread attention, with recent extensions in-

cluding TabPFN v2 scaling to larger datasets through subsampling strategies[35]; Drift-Resilient TabPFN introduces an adaptive mechanism to handle distribution drift[36];The paper by van Breugel and van der Schaar directly states that foundational models for tables should be the focus of research[3].

However, TabPFN still faces unresolved issues: the original model imposes rigid constraints on sample size and feature count; moreover, its performance under extremely sparse conditions has yet to be systematically investigated. How does it perform in real-world financial settings? This is precisely the core question this study seeks to address.

2.4 Explainable Artificial Intelligence in Finance

In real-world financial environments, regulatory compliance and privacy requirements must also be considered, making model interpretability crucial. To explain predictions from black-box models, researchers have proposed various post-hoc explanation methods:

For example, SHAP, based on the Shapley value from game theory, decomposes predictions into the marginal contributions of each feature[37].LIME provides intuitive feature weight interpretations for individual predictions by fitting local linear models near the prediction point[38]. Integrated Gradients assign feature importance by integrating gradients along the input path for neural networks[39].

However, in real-world scenarios, deploying such models as operational rules remains challenging. Decision-makers require not merely the features contributing most to rejection, but explicit rules such as "automatic approval if condition A is met." Additionally, research indicates that SHAP explanations are sensitive to input perturbations, undermining their credibility in high-stakes decision-making[40].Therefore, this study seeks to bridge this gap through threshold optimization and shallow decision tree distillation, converting the final model predictions into directly deployable rules that balance predictive capability with auditability.

3 Preliminaries

This chapter introduces the theoretical foundation for subsequent experiments, formalizing credit risk assessment as a machine learning problem. It then details the architecture and mechanisms of TabPFN, the core model of this study. Following this, it conducts a systematic analysis of the experimental dataset and finally establishes a unified evaluation framework.

3.1 Problem Formulation

3.1.1 Machine Learning Modeling for Credit Risk Management

The core task of credit risk assessment is to predict an applicant's probability of default based on their multidimensional characteristics. This process involves screening applicants into three categories: Accept, Refer, and Decline. The Refer category is further converted into Accept and Decline, ultimately formalizing the assessment as a supervised learning binary classification problem. As shown in the figure below:

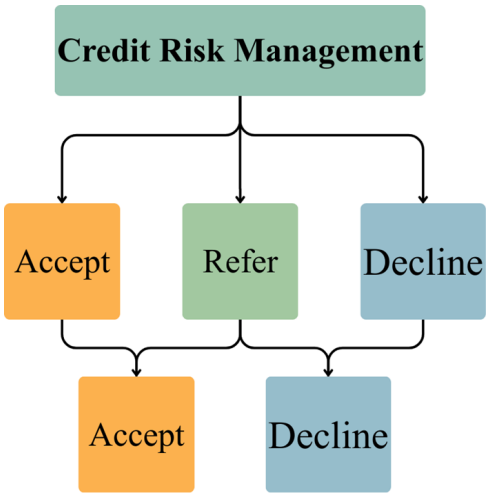


Figure 3.1: Risk Credit Management Process.

Definition 3.1 (Credit risk classification). Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ denotes the d-dimensional feature vector of the i-th applicant and $y_i \in \{0, 1\}$ denotes the final decision (0=Accept, 1=Decline), the goal is to learn a mapping

$$f : \mathcal{X} \to [0, 1]$$

such that $f(\mathbf{x})$ outputs the posterior probability $P(Y = 1 \mid \mathbf{x})$.

In practical operations, decisions are not made directly based on the binary output of f(x), but rather through a ternary routing strategy implemented using a probability threshold.

Definition 3.2 (Ternary triage policy). Given a threshold parameter $\tau \in (0.5, 1)$,

define the decision function

$$g_\tau(\mathbf{x}) = \begin{cases} \text{Auto-Accept} & \text{if } f(\mathbf{x}) \leq 1 - \tau \\ \text{Auto-Decline} & \text{if } f(\mathbf{x}) \geq \tau \\ \text{Manual-Review} & \text{otherwise.} \end{cases}$$

This strategy categorizes applications into three groups: high-confidence automatic approvals, high-confidence automatic rejections, and an uncertain range requiring manual review.

3.1.2 Data Quality Assessment Metrics

Before conducting the experiment, a systematic evaluation of the data is required. Therefore, data quality checks are performed in the following three aspects: Feature sparsity, pseudo-noise generated by preprocessing, and feature heterogeneity

1.Feature sparsity:

Feature sparsity refers to the phenomenon where a significant proportion of values are missing in the feature matrix. In tabular data, it can be evaluated separately at the row level (from the sample perspective), column level (feature dimension), and overall. **Definition 3.3 (Row, Column, Overall Sparsity).** Given a sample $\mathbf{x}_i \in \mathbb{R}^d$, let its missingness indicator be $\mathbf{m}_i \in \{0,1\}^d$, where $m_{ij} = 1$ indicates that the j-th feature is missing for sample i (and $m_{ij} = 0$ otherwise). For a dataset with N samples and d features, the row sparsity, column sparsity, and overall sparsity are defined as:

$$s_i^{\text{row}} = \frac{1}{d}\sum_{j=1}^{d} m_{ij}, \qquad s_j^{\text{col}} = \frac{1}{N}\sum_{i=1}^{N} m_{ij}, \qquad \rho = \frac{1}{Nd}\sum_{i=1}^{N}\sum_{j=1}^{d} m_{ij}.$$

When ρ is high, the effective information available for model training is extremely limited.

2.Pseudo-noise in data:

Compared to data noise, random errors, corruption, or irrelevant variations may occur in datasets. Due to policies and privacy protections, datasets may experience systematic information loss. Therefore, the concept of pseudo-noise is introduced. The following are the specific definitions:

Definition 3.4 (Measurement Noise). Measurement noise refers to random disturbances introduced during data collection, such as a customer's age being recorded as "280 years old" instead of "28 years old." Such noise typically follows a random distribution.

Definition 3.5 (Pseudo-noise). Pseudo-noise refers to systematic information loss introduced by deliberate data preprocessing, often unavoidable due to operational or regulatory constraints. In this study, pseudo-noise primarily originates from two sources:

(a) **Binning of continuous variables.** Due to data privacy protection requirements, all financial domains involving customer financial information are independently

classified according to predetermined intervals:

$$\tilde{x}_j = B_k, \text{ if } x_j \in [b_{k-1}, b_k),$$

where $\{B_1, B_2, \ldots, B_K\}$ denotes the set of bin labels and $\{b_0, b_1, \ldots, b_K\}$ denotes the bin boundaries.

The specific binning scheme applied is: 100K–250K, 250K–500K, 500K–750K, 750K–1M, and $>$ 1M (unit: EUR). For example, in two applications, the requested amounts are €267,500 and €490,000 respectively. After clustering, both are mapped to the same category "250K–500K." From a risk perspective, these two applications may exhibit significantly different risk characteristics and belong to distinct risk levels. However, after partitioning, this distinction is lost, and the model cannot differentiate between them.

Meanwhile, information loss can be quantified through entropy reduction:

$$\Delta H = H(X) - H(\tilde{X}),$$

where $H(X)$ is the entropy of the original continuous variable and $H(\tilde{X})$ is the entropy of the binned variable. Coarser binning leads to larger ΔH.

(b) **Conflation of zero values and missing values.** On the other hand, the binning process combines true zeros and missing values into a single "zero or missing" category. This conflation obscures a potentially important distinction:

- **True Zero:** the applicant genuinely has no value for this attribute (e.g., a new customer with zero existing exposure).
- **Missing:** the information was not collected or is unavailable (e.g., exposure data was not queried due to cost constraints).

These two scenarios may carry fundamentally different risk implications: new users represent unknown risks, while existing users with missing values require supplementation through data quality reviews and external data sources.

3.Feature Heterogeneity: A feature set $\{x_1, x_2, \ldots, x_d\}$ exhibits *heterogeneity* when it contains multiple data types, including:

- **Numerical features** $x_j^{(\text{num})} \in \mathbb{R}$: continuous or discrete numeric values (e.g., employee count, years in business).
- **Categorical features** $x_j^{(\text{cat})} \in \mathcal{C}_j$: discrete labels from a finite set (e.g., industry code, product type).
- **Ordinal features** $x_j^{(\text{ord})} \in \{1, 2, \ldots, K\}$: ordered categories (e.g., credit rating grades).

The cardinality $|\mathcal{C}_j|$ of categorical features may vary significantly. High-cardinality categorical features pose particular challenges for encoding and generalization.

3.1.3 Optimization Objective

The optimization objectives of this study consist of two layers:

Predictive performance objective. To minimize the cross-entropy loss between predicted probabilities and true labels:

$$\mathcal{L}_{\text{pred}} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log f(x_i) + (1-y_i)\log(1-f(x_i))\right].$$

Business deployment objective. Under the constraint of maintaining automated decision purity, Coverage measures the proportion of samples routed to the automated decision buckets. For the auto-approval bucket, more specifically:

$$\text{Coverage}_A(\tau) = \frac{|\{i:\ f(\mathbf{x}_i) \le 1-\tau\}|}{N},$$

and for the auto-rejection bucket,

$$\text{Coverage}_D(\tau) = \frac{|\{i:\ f(\mathbf{x}_i) \ge \tau\}|}{N}.$$

A higher coverage indicates that more cases can be processed automatically, thereby reducing the workload of manual review.

Purity measures the proportion of correct decisions within each automated decision bucket. Specifically,

$$\text{Purity}_A(\tau) = \frac{|\{i:\ f(\mathbf{x}_i) \le 1-\tau\ \wedge\ y_i = 0\}|}{|\{i:\ f(\mathbf{x}_i) \le 1-\tau\}|}, \qquad \text{Purity}_D(\tau) = \frac{|\{i:\ f(\mathbf{x}_i) \ge \tau\ \wedge\ y_i = 1\}|}{|\{i:\ f(\mathbf{x}_i) \ge \tau\}|}.$$

$\text{Purity}_A(\tau)$ denotes the fraction of truly approvable cases among those automatically approved, while $\text{Purity}_D(\tau)$ denotes the fraction of truly rejectable cases among those automatically rejected. Purity reflects the reliability of automated decisions.

The aim is to maximize the automated coverage rate:

$$\max_{\tau}\left[\text{Coverage}_A(\tau) + \text{Coverage}_D(\tau)\right]$$

$$\text{s.t.}\quad \text{Purity}_A(\tau) \ge \theta, \qquad \text{Purity}_D(\tau) \ge \theta.$$

Here, θ denotes the minimum purity threshold required for business deployment. In this study, θ is set to 0.90. This will be discussed in greater detail in Chapter 6.

3.2 TabPFN Architecture and Mechanism

TabPFN (Tabular Prior-Data Fitted Network) is the core model evaluated in this study. This section details its theoretical foundation, architectural design, and operational mechanism.

3.2.1 From Bayesian Inference to Contextual Learning

The theoretical foundation of TabPFN lies in the connection between Bayesian inference and in-context learning (ICL). Given a training set $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and a test input $\mathbf{x}_{test}$, the Bayesian posterior predictive distribution is:

$$p(y_{test}|\mathbf{x}_{test}, \mathcal{D}_{train}) = \int p(y_{test}|\mathbf{x}_{test}, \theta)\, p(\theta|\mathcal{D}_{train})\, d\theta$$

where θ represents model parameters and $p(\theta|\mathcal{D}_{train})$ is the posterior distribution over parameters given the training data.

Computing this integral exactly is generally intractable, as it requires marginalizing over all possible parameter configurations weighted by their posterior probability.

TabPFN circumvents this computational challenge by training a neural network q_ϕ to directly approximate the posterior predictive distribution. Rather than explicitly inferring parameters θ and then making predictions, the network learns to map directly from $(\mathbf{x}_{test}, \mathcal{D}_{train})$ to $p(y_{test}|\mathbf{x}_{test}, \mathcal{D}_{train})$. This approach, termed in-context learning, was first observed in large language models[41], and has since been shown to enable transformers to implicitly implement learning algorithms such as gradient descent and Bayesian inference[42].

3.2.2 Prior-Data Fitted Networks

The Prior-data Fitted Network (PFN) framework formalizes this approach. The key insight is that instead of training on a single dataset, the model can be trained across a distribution of datasets sampled from a prior $p(\mathcal{T})$, where each task $\mathcal{T}$ consists of a training set and associated test samples.

The PFN training objective minimizes the expected cross-entropy loss across all synthetic tasks:

$$\mathcal{L}_{PFN} = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\, \mathbb{E}_{(\mathcal{D}_{train}, \mathbf{x}_{test}, y_{test})\sim\mathcal{T}} \left[-\log q_\phi(y_{test}|\mathbf{x}_{test}, \mathcal{D}_{train})\right]$$

where q_ϕ denotes the neural network parameterized by ϕ. Crucially, the parameters ϕ are shared across all tasks, because neural networks learn general learning algorithms, not task-specific parameters. After training, when presented with a new real-world dataset, the network performs learning and prediction simultaneously in a single forward pass, with ϕ held fixed.

This framework can be interpreted as approximating Bayesian model averaging: the trained PFN implicitly integrates over a weighted space of possible learning algorithms, with the weights determined by the prior distribution used during training[33]. The quality of predictions on real-world data therefore depends critically on the alignment between the synthetic training prior and the characteristics of real-world datasets.

3.2.3 Synthetic Data Generation via Structural Causal Models

The performance of TabPFN relies on generating synthetic training datasets that capture the diversity and challenges of real-world tabular data. TabPFN employs a generative process based on Structural Causal Models (SCMs)[43].

The generation flow proceeds as follows: First, high-order hyperparameters determining dataset attributes (such as sample size, feature count, and task difficulty) are sampled from predefined distributions. Based on these hyperparameters, a directed acyclic graph (DAG) $\mathcal{G}$ is constructed. Each node in the graph stores a vector representation, while edges implement computational transformation mechanisms, including: small neural networks employing multiple activation functions, decision tree structures encoding rule dependencies, and discretization mechanisms for generating categorical features.

During sample generation, random noise propagates from the root node of the causal graph, undergoing transformations along edges. Feature values and target values are extracted from randomly selected intermediate nodes, yielding samples with complex, realistic dependencies. Post-processing introduces additional challenges: Kumaraswamy distribution deformations introduce nonlinear distortions, quantization simulates discrete features, and random masking simulates missing values.

Through this generative process, TabPFN was trained on approximately 100 million synthetic datasets, each with unique causal structures, feature types, and functional characteristics. This phase of work also constitutes the offline pre-training component, with the process illustrated in the figure below:

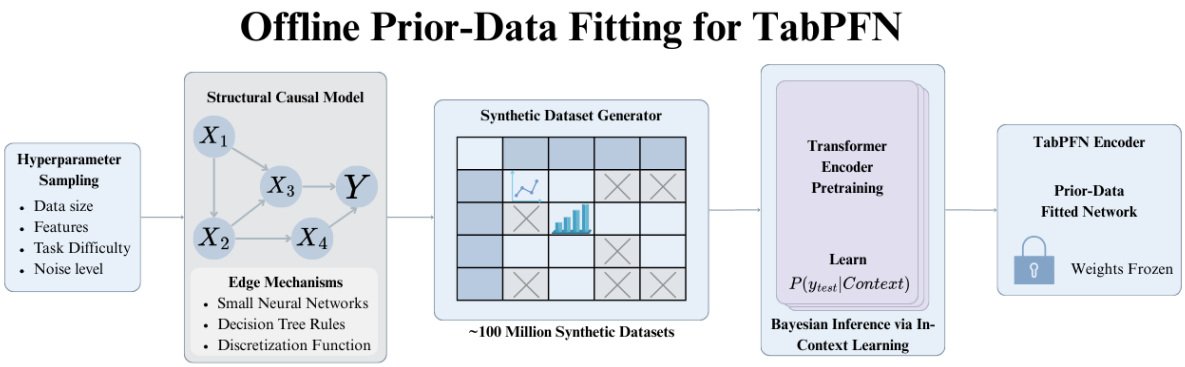


Figure 3.2: TabPFN Offline Pre-training Workflow.

3.2.4 Transformer Architecture for Tabular Data

TabPFN employs a modified Transformer encoder architecture specifically designed for the two-dimensional structure of tabular data[44]. Standard transformers treat input as a one-dimensional sequence. TabPFN assigns a separate representation to each cell in the table and employing a Bidirectional attention mechanism.

The architecture processes an input table with n training samples, m test samples, and d features as follows. Each cell (i, j)representing the j-th feature of the i-th sample, and is first encoded into an embedding vector through a linear projection. The model then

applies alternating attention operations: feature attention allows each cell to attend to other features within the same sample (row), while sample attention allows each cell to attend to the same feature across all samples (column).

During sample attention, test samples are permitted to attend to training samples but not to each other. This ensures that predictions for different test samples remain independent and that the training set representations are not influenced by the test set. Formally, for test sample embeddings $\mathbf{h}_{test}$, the attention is computed as:

$$\text{Attention}(\mathbf{Q}_{test}, \mathbf{K}_{train}, \mathbf{V}_{train}) = \text{softmax}\left(\frac{\mathbf{Q}_{test}\mathbf{K}_{train}^{T}}{\sqrt{d_k}}\right)\mathbf{V}_{train}$$

Therefore, the representations of training samples can be computed once and cached, allowing efficient prediction on multiple test samples without redundant computation.

The final layer transforms the test sample embeddings into output predictions. For classification tasks, a softmax layer produces class probability distributions. For regression tasks, TabPFN outputs a piece-wise constant distribution over target values.

This stage is also referred to as the Online Inference phase, with the specific workflow illustrated in the diagram below:

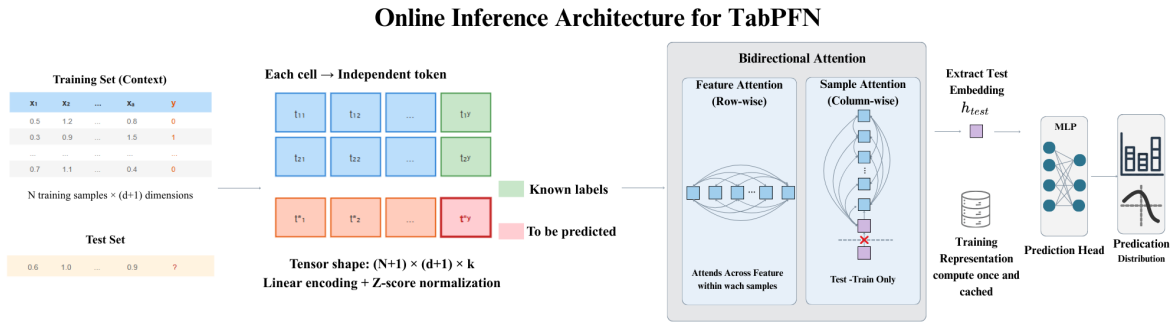


Figure 3.3: Online Inference Architecture for TabPFN Workflow.

Based on the probability outputs from TabPFN, this study further introduces Isotonic Regression for post-processing probability calibration. This section will be detailed in Chapter 4.

3.3 Dataset Description and Analysis

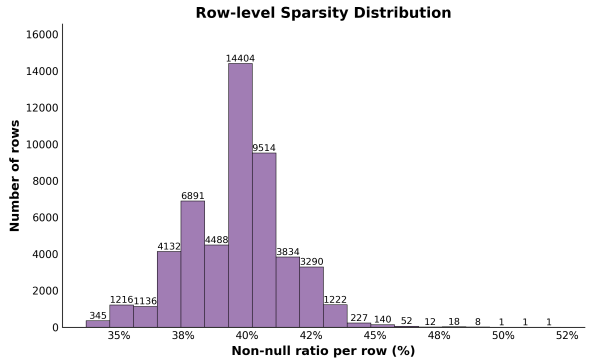
The dataset used in this study originates from DLL's real-world user data in North America. Within the company's credit decision-making process, the system receives multidimensional information about applicants, automatically generates preliminary decision recommendations, and subsequently undergoes manual review by specialists to produce a final binary decision outcome. The target variable in this study is precisely this final decision result.

3.3.1 Dataset Quality Analysis

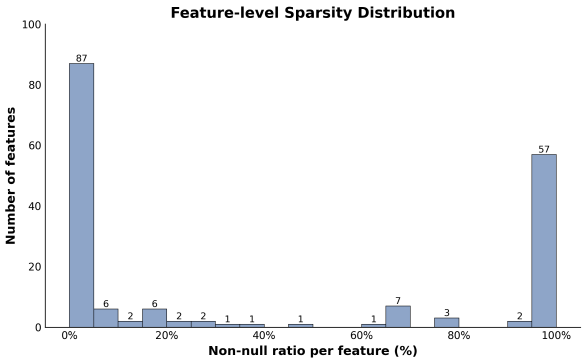
This dataset contains 50932 rows, 178 columns. The target variable distribution is Accept: 50.97%, Decline: 49.03%. The target variable distribution is nearly balanced. All monetary fields are expressed in thousands of euros (K EUR) and have undergone binning.

In terms of sparsity, the overall missing rate is $\rho = 60.46\%$. From the row dimension perspective, As shown in Figure 3.4a, each application record has values for only 39.5% of its features on average. This proportion is highly consistent across samples, with a standard deviation of just 0.023, indicating that nearly all samples exhibit a "uniformly sparse" pattern. There are no subgroups with particularly rich or poor information content.

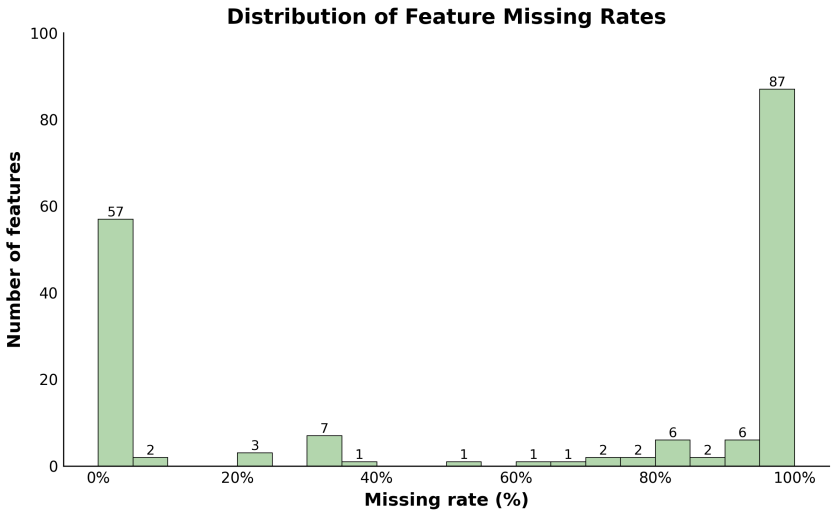
From the column dimension, As shown in Figure 3.4b, feature coverage exhibits extreme polarization. Moreover, in terms of the rate of missing data, As shown in Figure 3.4c, among 178 features, 57 are entirely empty, 87 have coverage below 5%, and only about 14 features have values across all samples. This extreme distribution implies that traditional mean-filling strategies may introduce severe bias.



(a) Rows-Sparsity Distribution.



(b) Feature-Level Distribution.



(c) Missing Rate Distribution.

Figure 3.4: Distribution of Sparsity.

After grouping by business dimension, as shown in Table 3.1, significant variations exist in the missing rate across categories: Application Information and Exposure and Quota

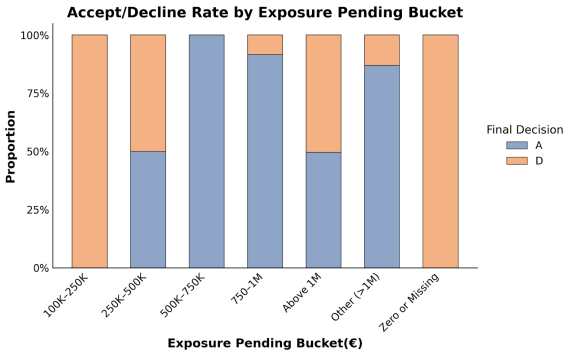
Indicators exhibit virtually no missing values; Customer and Business Attributes show partial missingness; while Trade/Credit Behavior features are almost entirely empty. This indicates that missingness is a systemic phenomenon driven by data collection processes and the cost of querying external data sources.

Table 3.1: Missing-rate summary by feature group.

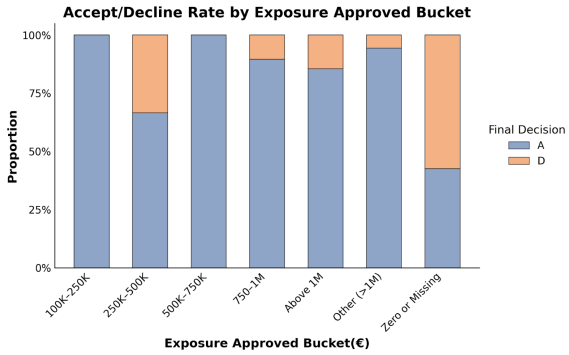
Feature Group	**Features**	**Mean**	**Median**	**Range**
Application/Transaction Basic Info	12	0.003	0.000	[0.00, 0.04]
Customer and Business Attributes	28	0.194	0.109	[0.00, 0.81]
Financial Product Attributes	15	0.271	0.000	[0.00, 0.81]
Exposure and Quota Indicators	20	0.002	0.000	[0.00, 0.03]
Trade/Credit Behavior	45	0.967	1.000	[0.82, 1.00]
Financial Bucket Field	58	0.012	0.000	[0.00, 0.19]

Additionally, from the perspective of pseudo-noise caused by information loss, the information degradation introduced by binning far exceeds the impact of typical measurement noise. Monetary variables such as loan application amount, registered risk exposure, and down payment amount are mapped to coarse-grained intervals. Among these three variables, the "zero or missing" category accounts for 0.6%, 62.8%, and 97.9% of observations respectively, generating spurious noise and distorting the modeling signal.

The following three typical features further illustrate this issue. First, for the risk exposure bucket feature (Figures 3.5a and 3.5b), a significant distribution bias exists between bucket label assignments and final decisions. The medium-to-high exposure ranges (e.g., 500K–750K) are almost entirely approved, while the "zero or missing" bucket was the only category predominantly rejected. Only a few ranges (e.g., 250K–500K) demonstrated some risk differentiation capability, indicating the model largely learned the bucket labels themselves. Second, regarding bill size, medium-to-large bills exhibit significantly higher approval rates, while small bills are more likely to be rejected, reflecting a strong correlation between approval decisions and transaction scale. Third, external credit scores (FICO) exhibit extreme sparsity due to query cost constraints: among the five FICO fields, only P1 has values in approximately 11.7% of samples, P2 is below 1%, and P3 to P5 are close to zero. This renders such features informationally worthless for the vast majority of records.



(a) Exposure data distribution (Set 1).



(b) Exposure data distribution (Set 2).

Figure 3.5: **Exposure data**: Two views of the exposure data distribution.

3.4 Performance Metrics

This section establishes a unified model evaluation framework, with the following specific metrics: **Definition 3.6 ROC-AUC.** Let $\mathcal{D}^+ = \{i : y_i = 1\}$ and $\mathcal{D}^- = \{i : y_i = 0\}$ denote the sets of positive and negative samples, respectively. The area under the ROC curve (AUC) is defined as the probability that a randomly selected positive sample receives a higher score than a randomly selected negative sample:

$$\text{AUC} = \frac{1}{|\mathcal{D}^+|\,|\mathcal{D}^-|} \sum_{i \in \mathcal{D}^+} \sum_{j \in \mathcal{D}^-} \mathbf{1}[f(\mathbf{x}_i) > f(\mathbf{x}_j)]\,.$$

The AUC takes values in $[0, 1]$, where larger values indicate stronger discriminative capability[45].

Definition 3.7 Log Loss. Log loss measures the cross-entropy between predicted probabilities and true labels:

$$\text{LogLoss} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log f(\mathbf{x}_i) + (1 - y_i) \log\big(1 - f(\mathbf{x}_i)\big) \right].$$

The log loss takes values in $[0, +\infty)$, where smaller values indicate more accurate probabilistic estimates.

Definition 3.8 Brier Score. The Brier score measures the mean squared error between predicted probabilities and true labels:

$$\text{Brier} = \frac{1}{N} \sum_{i=1}^{N} \big(f(\mathbf{x}_i) - y_i\big)^2.$$

The Brier score takes values in $[0, 1]$, where smaller values are better. Moreover, the Brier score can be decomposed into a calibration component and a refinement component.

Definition 3.9 Confusion metrics. Given a decision threshold t, the predicted label is defined as

$$\hat{y}_i = \mathbf{1}[f(\mathbf{x}_i) \geq t]\,.$$

Based on the confusion matrix, the following metrics are defined:

$$\text{Accuracy} = \frac{TP + TN}{N}, \qquad \text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN},$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Here, TP, TN, FP, and FN denote the numbers of true positives, true negatives, false positives, and false negatives, respectively. Another point to note is that in subsequent interpretability analyses of the model, a trade-off exists between manual review rates and automated decision error rates. To address this, the concept of the Pareto frontier is introduced for analysis, which will be explained in detail in Chapter 6[46].

The definition is as follows: **Definition 3.10 Pareto Frontier.** Let the threshold parameter space be defined as

$$\mathcal{T} = \{\tau : 0.5 < \tau < 1\}.$$

Consider the following objective functions:

- $f_1(\tau) = \text{ReviewRate}(\tau)$, representing manual review cost;
- $f_2(\tau) = \text{ErrorRate}(\tau) = \text{FN}_A(\tau) + \text{FP}_D(\tau)$, representing error exposure.

The Pareto frontier is defined as the set of all Pareto-optimal solutions:

$$\mathcal{P} = \{\tau^* \in \mathcal{T} : \nexists\, \tau \in \mathcal{T} \text{ such that } f_1(\tau) < f_1(\tau^*) \ \wedge \ f_2(\tau) < f_2(\tau^*)\}\,.$$

A threshold τ^* is Pareto-optimal if no other threshold simultaneously achieves lower manual review cost and lower error exposure.

4 Method

This chapter introduces the overall framework, improvement methods, and specific methods for explainable artificial intelligence.

4.1 Overview of the Framework

This study constructs a five-stage experimental framework spanning from data comprehension to operational deployment.As shown in Figure 4.1,

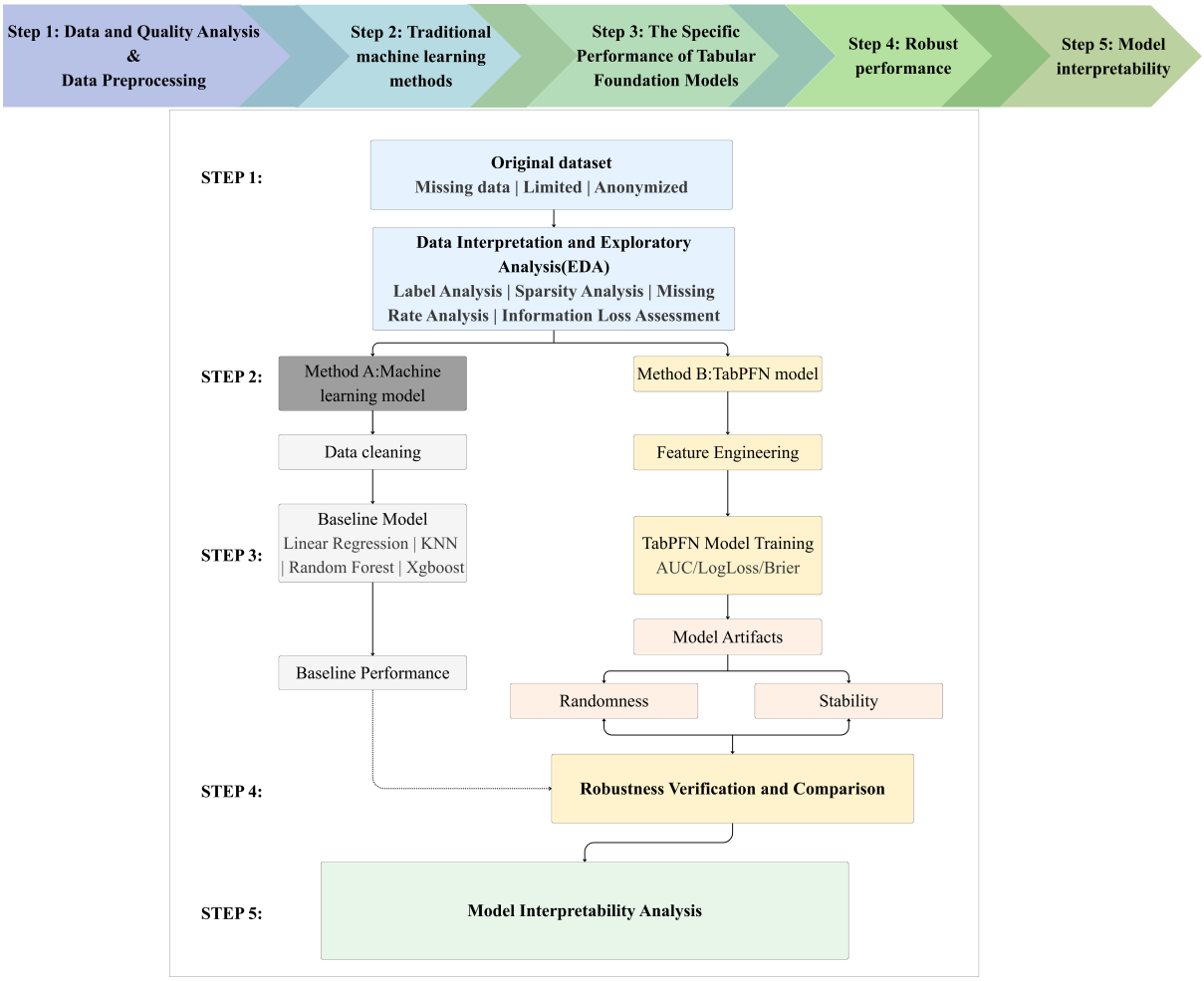


Figure 4.1: Overall Framework Process Diagram.

this framework first conducts a systematic quality assessment of the dataset to identify key data characteristics such as sparsity, pseudo-noise, and feature heterogeneity. Based on this foundation, it implements four classic baseline models: decision trees, KNN, random forests, and XGBoost, under a unified data processing workflow and evaluation metrics, establishing reproducible performance benchmarks corresponding to RQ1. Subsequently, feature engineering, training, inference, and performance evaluation for

TabPFN are completed tailored to the dataset's characteristics, addressing RQ2. Since TabPFN's Transformer architecture requires pure numerical matrices as input, this stage necessitates additional encoding and transformation steps; The fourth phase systematically evaluates model robustness under varying data degradation conditions through perturbation experiments, including stratified subsampling, label noise injection, and feature dropout simulation, addressing RQ3. The final phase converts model probability outputs into auditable business rules, achieving the dual optimization goal of prediction performance and operational deployment outlined in Section 3.1.3.

4.1.1 Data Preprocessing

This study establishes a unified pre-modeling standardization process. In terms of field standardization and feature engineering, all continuous monetary fields were uniformly mapped to corresponding bucketed fields. Features were organized into three categories based on business context: application dimension, enterprise information dimension, and CRA credit rating dimension. For high-cardinality categorical features, such as project names, a frequency truncation strategy is employed: the top 10 most frequent categories in the training set are retained, while the remainder are merged into a "rest" category to reduce dimensionality and prevent overfitting.

For missing value handling, a binary missing indicator variable $m_{ij} \in \{0, 1\}$ is generated for each numerical feature. The missingness itself is retained as a learnable signal, and the generated indicator column is incorporated into the categorical feature set. Subsequently, after distinguishing between identified numerical and categorical feature columns, the target variable is converted to "Final Decision" field and placed in the final column of the data frame, completing the structured data preprocessing.

4.1.2 Feature Engineering

The Transformer architecture of TabPFN requires pure numerical matrices as input. Therefore, beyond general preprocessing, categorical features and bucketed fields undergo additional numerical conversion.

For bucketed fields, numerical mapping is performed using the median of the interval (e.g., "250K–500K" maps to 375). For categorical features, missing values are uniformly labeled as "Missing". Long-tail categories appearing fewer than 50 times in the training set are merged into "Rare". Categories not present in the training set within the validation and test sets are labeled as "Unseen".

Building upon this foundation, to mitigate the common issue of dimensionality explosion following encoding, 5-fold hierarchical target encoding is performed on each category column and incorporate Bayesian smoothing. The core approach involves: Partitioning the training set into 5 folds based on label stratification; For samples in the jth fold, their target encoding is computed solely using the target statistics of that category from the remaining 4/5 training folds (the out-of-fold portion), ensuring each training sample's encoding does not directly depend on its own label. For the validation and test sets, encoding is performed using category statistics from the entire training set.

To mitigate high variance caused by small sample sizes, Bayesian smoothing is applied to target mean encoding: means within categories contract toward a global prior mean. The

contraction strengthens as the number of samples in a category decreases and weakens as the number increases, approaching the true mean of that category. Formally, this can be expressed as:

$$\mathrm{TE}(c) = \frac{n_c \mu_c + \alpha \mu}{n_c + \alpha},$$

where n_c is the number of samples in category c, μ_c is the target mean of that category, μ is the global target mean, and α is the smoothing hyperparameter.

Additionally, to supplement category information and further stabilize model training, frequency encoding for each category is incorporated as an auxiliary feature. Finally, the numerical columns are concatenated with the target encoding and frequency encoding for each category to form the feature vector, with missing values filled with -1.

4.1.3 Probability Calibration

As defined in Section 3.2, the three-way diversion strategy partitions the predicted probability using the threshold τ to partition prediction probabilities. If the original probabilities exhibit systematic bias, where predicted values diverge from empirical label frequencies, the threshold's operational significance diminishes. For instance, when a model outputs p=0.9 but the actual rejection rate is only 0.75, the purity of the automatic rejection bucket falls below the expected reliability threshold. Therefore, Isotonic Regression is introduced for posterior probability calibration[47].

Isotonic regression is a rank-preserving regression method. Given the predicted probability $\hat{p}_i = f(\mathbf{x}_i)$ and the true label y_i on the validation set, rank-preserving regression fits a non-decreasing function. The problem is formulated as follows:

$$\min \sum_i w_i (y_i - \hat{y}_i)^2 \quad \text{subject to} \quad \hat{y}_i \leq \hat{y}_j \text{ for } X_i \leq X_j,\ w_i > 0 \tag{4.1}$$

For the current element y_i, if y_{i+1} is out of order with y_i, the average of the two elements is taken to replace their original values. If the next element y_{i+2} still does not maintain the same order as their average, the average of these three elements is used to replace their original values. This process continues recursively until the next element y_{i+m}shares the same order as them. y_{i+m} is then designated as the new local starting element, and the preceding process is repeated. During model calibration, the original model predicts estimated values $P(x_i)$ on the dataset. $P(x_i)$ is sorted by magnitude and partitioned into n bins. For each bin, the true label ratio of the samples within it is calculated. The resulting curve is the reliability curve. Ordered regression is performed on the points of the reliability curve to obtain the final calibrated probability.

4.2 XAI Techniques

4.2.1 Global Feature Importance

The primary task is to analyze which features contribute the most. Given the computational overhead during the model training phase, directly employing exhaustive methods like permutation importance is prohibitively costly. Therefore, the Spearman rank corre-

lation coefficient is adopted as the importance metric to calculate the relationship between each feature x_j and the model prediction probability $p(x)$[48]:

$$I_j = |\rho_S(x_j,\ p(x))|$$

This method requires no access to the model's internal structure, relying solely on the monotonic relationship between inputs and outputs. To enhance robustness, experiments incorporate 100-time Bootstrap resampling, with the final mean serving as the ultimate score.

4.2.2 Local Contribution Attribution

Spearman's rank correlation measures the overall monotonic association between features and predicted probabilities. To further examine how each feature contributes to prediction outcomes under the influence of other features, Shapley Values for Prediction are employed in conjunction with a proxy model for local attribution analysis[40]. Specifically, the Gradient Boosting Regressor is fitted to TabPFN's probability output $p(x)$, then compute the Shapley values for each feature using TreeSHAP. TreeSHAP efficiently performs weighted summation over all possible feature subsets within the tree model structure, calculating the expected marginal contribution of each feature to the prediction value across different feature sets to derive Shapley attribution.

To synthesize the conclusions of both methods, this study computes the rank sum of each feature across the Spearman and SHAP rankings:

$$R_j = \text{rank}_j^{\text{Spearman}} + \text{rank}_j^{\text{SHAP}}.$$

The final composite importance ordering is obtained by sorting R_j in ascending order, which provides a more robust ranking.

4.2.3 High-Confidence Binning and Rule Distillation

Based on the probability-calibrated $\tilde{p}_i$, this section further implements the three-way diversion strategy defined in Section 3.2 and the optimization objective from Section 3.1.3.

For threshold selection, a grid search is conducted over $\tau \in [0.50, 0.99]$ with step size 0.01 to identify the threshold that minimizes the manual review rate while satisfying the purity constraints.

To convert the model's continuous probability decision boundary into deployable business rules, this study trains shallow decision trees to approximate the ternary binning labels $\{A, \text{Review}, D\}$. Notably, the fitting target is not the original class labels, but the ternary binning labels generated by the selected threshold τ^*. The decision tree is trained using features that rank highly in the comprehensive evaluation in Section 4.2.2, and is regularized by limiting the maximum depth and enforcing a minimum number of samples per leaf. Depths d $d \in \{2, 3, \ldots, 10\}$ are scanned and export the resulting tree as a deployable `IF-ELSE` decision function.

Furthermore, to quantify the trade-off between the manual review rate and the automated decision error rate, the Pareto frontier is computed for different values of τ according to

Definition 3.10. This analysis provides stakeholders with a set of actionable operating points: along the frontier, any reduction in the review rate necessarily incurs increased error exposure.

5 Experiments and Results

This chapter systematically presents the experimental design and results of this study. Section 5.1 describes the experimental design, dataset partitioning, baseline model configuration, TabPFN training, and robustness testing setup. Section 5.2 presents the experimental results, discussing in turn the performance of baseline models, the performance of TabPFN, and the specific manifestations of robustness.

5.1 Experimental Design

5.1.1 Experimental Setup

This system is implemented in PyTorch and all experiments are conducted on Databricks compute clusters. The training and evaluation jobs are executed on GPU-enabled instances equipped with an NVIDIA GeForce RTX 4060 Ti. The CPU resources are provided and managed by Databricks as part of the cluster configuration. The version of TabPFN is V2.5.

5.1.2 Dataset Partitioning Strategy

The dataset was divided into three subsets using stratified random sampling. A fixed random seed of 42 was used to allocate 20% of the full dataset as the test set. The remaining 80% was further split into training and validation sets at an 8:2 ratio using the same seed. The approval-to-rejection ratio remained balanced across all three subsets during the stratification process.

Table 5.1: Summary of dataset splits used in the experiments. n denotes the number of samples and d denotes the feature dimension.

Split	n (samples)	d (features)
Train	32,596	64
Validation	8,149	64
Test	10,187	64
Full dataset	50,932	72

Another point is that model inputs consist of two types of features: numerical features and categorical features. Numerical features are directly used for modeling. Categorical features undergo unified missing value handling and rare category merging before entering the model, and are encoded to convert them into continuous numerical representations. During categorical feature preprocessing, high-frequency categories retain their distinct labels, while low-frequency categories are merged into a unified "other" category. Additionally, unknown categories are ignored during encoding. Furthermore, interval-scaled variables are mapped to numerical values.

5.1.3 Baseline Model Configuration and Parameter Tuning

This study trained four traditional supervised classifiers as baseline models. All baseline models utilized a unified 71-dimensional numerical feature set supplemented by category features processed through One-Hot Encoding, which expands a categorical variable into multiple 0/1 binary features. The specific configurations are as follows:

1. **Decision Tree**: A decision tree recursively partitions the feature space by selecting, at each node, the split threshold that best separates the data, yielding a root-to-leaf *if–else* rule structure for prediction. Hyperparameters are tuned via grid search with stratified 5-fold cross-validation, exploring the maximum tree depth, minimum number of samples required to split an internal node, minimum number of samples in a leaf node, and split criterion. Sample weights are incorporated during fitting, and the selected configuration is evaluated on the validation and test sets.

2. **K-Nearest Neighbors**: For a query instance, KNN retrieves the K closest training samples in the feature space and predicts via distance-weighted voting, using either Euclidean or Manhattan distance as the similarity metric. Hyperparameters are optimized using randomized search with stratified 5-fold cross-validation, sampling the number of neighbors, distance metric, and voting weight strategy. The best configuration is then assessed for generalization on the test set.

3. **Random Forest**: Random Forest trains an ensemble of decision trees on bootstrap-resampled instances with feature subsampling, and aggregates predictions via majority voting (classification) to reduce variance and mitigate overfitting. Key hyperparameters, including the number of trees, maximum depth, minimum split size, minimum leaf size, and the number of features considered per split—are tuned using randomized search with stratified 5-fold cross-validation. Performance is reported on the held-out test set.

4. **XGBoost**: XGBoost builds an additive ensemble of weak trees in a stage-wise manner, where each successive tree fits the gradients of the loss from the previous iteration. The baseline configuration specifies the number of trees, maximum depth, learning rate, and subsampling ratios, and employs histogram-based training for computational efficiency, with log loss as the internal evaluation objective to promote stable probabilistic outputs. Hyperparameter tuning is conducted via randomized search with stratified 5-fold cross-validation over the number of trees, tree depth, learning rate, instance subsampling ratio, and feature subsampling ratio. The best cross-validated model is selected and evaluated on the test set.

5.1.4 Training TabPFN

In this experiment, the training phase of TabPFN utilizes the training set for model fitting, the validation set for model selection and subsequent calibration, and the test set for final reporting. Considering that TabPFN requires conditional inference on test samples relative to training subsets when predicting probabilities, this approach introduces increased GPU memory consumption and inference time during training. Therefore, a batch inference strategy is adopted: dividing the prediction samples into fixed-size batches, computing positive class probabilities batch by batch, and concatenating them to reconstruct the complete probability vector. The model performance report covers

both discrete classification quality and probability quality dimensions: - At the classification level, category labels are output based on a fixed threshold, and the confusion matrix and precision calibration effects are calculated. - The calibration effect is quantified by comparing changes in probability quality metrics on the test set before and after calibration, focusing on the reduction in log loss, recall, and F1 score. For ranking and discrimination, ROC-AUC measures the model's ability to distinguish between two sample classes. At the probability level, log loss and Brier score evaluate the alignment between predicted probabilities and true labels.

Additionally, for probability calibration, the raw prediction probabilities on the validation set serve as calibration inputs. A monotonic non-decreasing calibration mapping function is fitted using the validation set's true labels as the supervision signal. This mapping is then applied to the raw prediction probabilities on the test set to obtain calibrated probability outputs. Concurrently, this study further conducts error pattern analysis to understand typical scenarios where the model fails: By calculating single-sample log loss, "extreme error" samples are identified as samples exhibiting significant conflicts between true labels and high-confidence model predictions. These are contrasted with normally performing samples within the same true category to observe statistical differences and distribution shifts in the input feature space. This approach helps pinpoint potential sources of issues such as label noise, long-tail patterns, or insufficient feature coverage.

5.1.5 Robustness Testing

To evaluate the impact of data quality fluctuations encountered in real deployment scenarios on predictive performance, this study designs four sets of perturbation experiments. The evaluation metrics are consistently reported as ROC-AUC, LogLoss, and Brier Score. Each experiment is conducted on both TabPFN and XGBoost. The experimental designs are described as follows:

1. **Random seed stability.** The random seed for data splitting is set to five distinct values: 42, 123, 456, 789, and 2024. For each seed, the training, validation, and test sets are independently partitioned from the full dataset, and the models are trained accordingly. The mean and standard deviation of each metric are then computed across the five runs. This design mitigates the risk that a particular split may disproportionately allocate "easier" samples to the test set.

2. **Data-scale sensitivity.** With fixed splits and a fixed seed, training is performed on progressively larger subsets of the training set (10%, 20%, 40%, 60%, 80%, and 100%) via stratified sampling, while the test set remains unchanged. This experiment investigates the minimum labeled data volume required to achieve near-saturated performance, and whether TabPFN's zero-shot paradigm exhibits higher data efficiency than XGBoost's gradient-based training when labeled resources are constrained.

3. **Label noise sensitivity.** Symmetric label noise is injected into the training labels at rates of 0%, 5%, 10%, 15%, and 20%. At each noise level, the corresponding proportion of training labels (A and D) is randomly flipped. The target encoding is recomputed using the noisy labels, and the models are trained on the noisy data while evaluated against clean test labels. This experiment assesses robustness under subjective judgement differences and potential data-entry errors.

4. **Feature missing sensitivity.** Missing values are randomly introduced into the feature matrices of both the training and test sets at rates of 0%, 10%, 20%, 30%, and 40%. This experiment simulates increasingly incomplete data conditions and evaluates whether model performance degrades gracefully or collapses under severe missingness.

5.2 Results

This section presents the comparison results between the baseline model and TabPFN, and further evaluates the model's robustness under various data perturbation conditions.

5.2.1 Baseline Model Performance

Table 5.2 and Table 5.3 present the overall performance of the four baseline models on the validation and test sets. Overall, the ensemble tree model significantly outperformed both single decision trees and distance-based KNN: while decision trees achieved high Recall for the Accept class (validation set Recall=0.87, and Recall=0.88 on the test set), it exhibits low Recall for the Decline class (Recall=0.42 on the validation set and Recall=0.41 on the test set). This results in a high number of false negatives (FN errors) and reflects poor probability quality. KNN significantly outperformed decision trees in Accuracy, but its probability quality metrics were suboptimal, indicating noticeable distortion in this task.

Among all baseline models, XGBoost achieved optimal performance on both validation and test sets. On the validation set, XGBoost attained the highest accuracy while demonstrating superior precision and F1 score for the Decline class, alongside lower LogLoss and Brier scores. On the test set, XGBoost also achieved the highest accuracy, balancing FP and FN trade-offs with stable probability quality. In comparison, Random Forest's discriminative performance approached XGBoost's, but exhibited higher FN rates and slightly weaker probability quality on the test set. Consequently, XGBoost serves as the strongest baseline representative for subsequent evaluations.

This demonstrates that for this binary classification task, models require not only high classification accuracy but also stable and reliable probability outputs to support subsequent decision-making.

Table 5.2: Baseline model performance on the validation set ($n = 8149$).

Model	Acc	P(A)	R(A)	F1(A)	P(D)	R(D)	F1(D)	FP	FN	AUC	LogLoss	Brier
Decision Tree	0.65	0.61	**0.87**	0.72	0.76	0.42	0.54	**525**	2325	0.875	0.544	0.183
KNN	0.79	0.79	0.80	0.79	0.79	0.77	0.78	817	908	0.863	0.702	0.137
Random Forest	**0.82**	**0.81**	0.85	**0.83**	0.84	**0.79**	**0.81**	610	**857**	0.893	**0.411**	0.128
XGBoost	**0.83**	**0.81**	0.86	**0.84**	**0.85**	**0.79**	**0.82**	568	842	**0.894**	0.405	**0.127**

A = Accept; D = Decline. FP = false declines (A→D); FN = false approvals (D→A).
P = Precision; R = Recall. Best results are highlighted in bold.

Table 5.3: Baseline model performance on the test set ($n = 10187$).

Model	Acc	P(A)	R(A)	F1(A)	P(D)	R(D)	F1(D)	FP	FN	AUC	LogLoss	Brier
Decision Tree	0.65	0.61	**0.88**	0.72	0.77	0.41	0.54	**614**	2932	0.877	0.538	0.181
KNN	0.78	0.78	0.80	0.79	0.79	0.77	0.78	1048	1149	0.864	0.638	0.134
Random Forest	0.81	0.80	0.85	0.82	0.83	0.77	0.80	762	1139	**0.895**	**0.412**	0.130
XGBoost	**0.82**	**0.80**	0.87	**0.83**	**0.85**	**0.77**	**0.81**	688	**1128**	**0.895**	0.406	**0.128**

A = Accept; D = Decline. FP = false declines (A→D); FN = false approvals (D→A).
P = Precision; R = Recall. Best results are highlighted in bold.

5.2.2 TabPFN Performance

Table 5.4 shows the performance of TabPFN on the validation set and test set. Overall, TabPFN performs very similarly to the strongest baseline XGBoost under fixed data splits: TabPFN achieves an Accuracy of 0.82 on the test set, matching XGBoost. The error structures of the two models are also highly consistent, indicating that TabPFN does not significantly alter the overall trade-off between misdeclines and misaccepts. However, it slightly reduces misdeclines while maintaining the scale of misaccepts. Regarding probabilistic quality, TabPFN achieves LogLoss=0.405 and Brier=0.128 on the test set, performing essentially on par with XGBoost.

Furthermore, TabPFN achieves an AUC of 0.896 on the test set, maintaining discriminative capabilities comparable to strong baselines while demonstrating greater stability in distinguishing marginal samples. More importantly, TabPFN's outputs exhibit consistency at both the classification and probability levels: it delivers reliable probability predictions while maintaining accuracy and F1 scores comparable to optimal baselines.

Regarding probability calibration, Isotonic Regression performs post-processing on test set probabilities, maintaining a calibrated LogLoss of 0.405. This indicates the model's raw probabilities already exhibit good usability, with calibration primarily enhancing threshold interpretation consistency. Additionally, on the test set, 0.91% of extreme error samples that should be declined were predicted with rejection probabilities p<0.1. This proportion reflects the worst-case risk magnitude potentially exposed when applying low-risk thresholds for automated approval, providing an intuitive reference for threshold setting and risk control.

Table 5.4: TabPFN performance on the validation and test sets.

Split	Acc	P(A)	R(A)	F1(A)	P(D)	R(D)	F1(D)	FP	FN	AUC	LogLoss	Brier
Val	0.83	0.81	0.86	0.83	0.85	0.79	0.82	567	851	0.893	0.409	0.128
Test	0.82	0.80	0.87	0.84	0.85	0.77	0.81	664	1124	0.896	0.405	0.128

Split	Calibrated LogLoss (Isotonic)	Extreme errors ($y = 1$, $p < 0.1$)
Test	0.405	93 (0.91%)

Notes. A = Accept; D = Decline; FP = false declines; FN = false approvals; P = Precision; R = Recall.

5.2.3 Robustness Performance

To evaluate the model's robustness, this section constructs experiments across four dimensions: random seed variation, training data scale reduction, label noise augmenta-

tion, and feature dropout intensification. These experiments compare TabPFN against the strongest baseline, XGBoost.

1. **Random seed stability**.

   As shown in Table 5.5 and 5.6, both models exhibit minimal metric fluctuations across five random seed sets, indicating strong stability for both approaches. Taking the test set as an example, TabPFN achieves an average AUC of 0.8950, while XGBoost yields 0.8957, a negligible difference. Regarding probabilistic quality, XGBoost's average LogLoss and Brier scores slightly outperform TabPFN, though the gap remains extremely small.

Table 5.5: Robustness of TabPFN across random seeds (test set). Lower is better for LogLoss and Brier; higher is better for AUC.

Seed	AUC ↑	LogLoss ↓	Brier ↓
42	0.8953	0.4054	0.1282
123	0.8953	0.4033	0.1272
456	0.8933	0.4071	0.1282
789	0.8914	0.4112	0.1299
2024	0.8997	0.3961	0.1244
Mean ± Std	0.8950 ± 0.0031	0.4046 ± 0.0056	0.1276 ± 0.0020

Table 5.6: Robustness of XGBoost across random seeds (test set). Lower is better for LogLoss and Brier; higher is better for AUC.

Seed	AUC ↑	LogLoss ↓	Brier ↓
42	0.8940	0.4058	0.1283
123	0.8978	0.3992	0.1258
456	0.8944	0.4056	0.1280
789	0.8931	0.4084	0.1289
2024	0.8992	0.3964	0.1243
Mean ± Std	0.8957 ± 0.0026	0.4031 ± 0.0051	0.1271 ± 0.0019

2. **Data-scale sensitivity**.

   As shown in Table 5.7 and Table 5.8, TabPFN shows a clearer advantage when the available training data is limited. With only 10% of the training data, TabPFN achieves an average AUC of 0.8829 ± 0.0029, compared with 0.8755 ± 0.0030 for XGBoost. The difference is also visible in the probability-based metrics, where TabPFN obtains a lower LogLoss of 0.4246 ± 0.0050 and a lower Brier score of 0.1347 ± 0.0018, while XGBoost obtains 0.4572 ± 0.0097 and 0.1424 ± 0.0030, respectively. As the fraction of training data increases, the performance gap gradually narrows, and the two models become more similar on the full training set.

Table 5.7: TabPFN performance vs. training data fraction on the test set. Results are reported as mean ± standard deviation. Higher is better for AUC, while lower is better for LogLoss and Brier.

Fraction	AUC ↑	LogLoss ↓	Brier ↓
0.1	0.8829 ± 0.0029	0.4246 ± 0.0050	0.1347 ± 0.0018
0.2	0.8886 ± 0.0023	0.4151 ± 0.0048	0.1314 ± 0.0019
0.4	0.8921 ± 0.0028	0.4091 ± 0.0051	0.1293 ± 0.0018
0.6	0.8946 ± 0.0027	0.4053 ± 0.0050	0.1279 ± 0.0020
0.8	0.8958 ± 0.0028	0.4031 ± 0.0050	0.1272 ± 0.0019
1.0	0.8971 ± 0.0032	0.4010 ± 0.0055	0.1264 ± 0.0021

Table 5.8: XGBoost performance vs. training data fraction on the test set. Results are reported as mean ± standard deviation. Higher is better for AUC, while lower is better for LogLoss and Brier.

Fraction	AUC ↑	LogLoss ↓	Brier ↓
0.1	0.8755 ± 0.0030	0.4572 ± 0.0097	0.1424 ± 0.0030
0.2	0.8836 ± 0.0031	0.4298 ± 0.0055	0.1358 ± 0.0021
0.4	0.8906 ± 0.0030	0.4138 ± 0.0058	0.1304 ± 0.0021
0.6	0.8930 ± 0.0032	0.4083 ± 0.0062	0.1287 ± 0.0023
0.8	0.8947 ± 0.0023	0.4052 ± 0.0044	0.1279 ± 0.0018
1.0	0.8957 ± 0.0026	0.4031 ± 0.0050	0.1271 ± 0.0019

3. **Label noise sensitivity**.

   As shown in Table 5.9 and Table 5.10, both models show a gradual performance decline as the label noise level increases from 0% to 20%. Their AUC values decrease, while LogLoss and Brier scores increase, indicating that label noise affects both ranking performance and probability quality. Under the same noise level, TabPFN shows slightly better robustness than XGBoost. For example, at the 20% noise level, TabPFN achieves an average AUC of 0.8889 ± 0.0037, LogLoss of 0.4703 ± 0.0060, and Brier score of 0.1470 ± 0.0018, while XGBoost obtains 0.8837 ± 0.0027, 0.4767 ± 0.0050, and 0.1500 ± 0.0016, respectively. This suggests that TabPFN is slightly more resilient to label perturbations and maintains better probability quality under noisy supervision.

Table 5.9: Sensitivity to label noise for TabPFN on the test set. AUC is reported as mean ± standard deviation across five random seeds. Higher is better for AUC, while lower is better for LogLoss and Brier.

Noise level	AUC ↓	LogLoss ↑	Brier ↑
0.00	0.8971 ± 0.0032	0.4054 ± 0.0063	0.1282 ± 0.0030
0.05	0.8949 ± 0.0032	0.4124 ± 0.0051	0.1295 ± 0.0029
0.10	0.8935 ± 0.0033	0.4270 ± 0.0043	0.1331 ± 0.0021
0.15	0.8909 ± 0.0036	0.4463 ± 0.0060	0.1387 ± 0.0022
0.20	0.8889 ± 0.0037	0.4703 ± 0.0060	0.1470 ± 0.0018

Table 5.10: Sensitivity to label noise for XGBoost on the test set. AUC is reported as mean ± standard deviation across five random seeds. Higher is better for AUC, while lower is better for LogLoss and Brier.

Noise level	AUC ↓	LogLoss ↑	Brier ↑
0.00	0.8957 ± 0.0026	0.4058 ± 0.0088	0.1283 ± 0.0010
0.05	0.8930 ± 0.0034	0.4170 ± 0.0042	0.1308 ± 0.0015
0.10	0.8906 ± 0.0027	0.4327 ± 0.0052	0.1351 ± 0.0011
0.15	0.8874 ± 0.0038	0.4521 ± 0.0031	0.1411 ± 0.0012
0.20	0.8837 ± 0.0027	0.4767 ± 0.0050	0.1500 ± 0.0016

4. **Feature-missingness sensitivity**.

   As shown in Table 5.11 and Table 5.12, both models exhibit performance degradation as the feature missing rate increases, indicating that incomplete feature information reduces both predictive discrimination and probability quality. TabPFN retains a small absolute advantage at several missing-rate levels. At a 40% missing rate, it achieves an average AUC of 0.8796 ± 0.0021, LogLoss of 0.4299 ± 0.0039, and Brier Score of 0.1369 ± 0.0014, while XGBoost obtains 0.8739 ± 0.0015, 0.4404 ± 0.0033, and 0.1407 ± 0.0013, respectively. However, the margin remains limited and should not be interpreted as strong evidence that TabPFN is more robust to structural missingness. Rather, the results suggest that TabPFN remains competitive under additional missingness, while XGBoost continues to provide a stable reference due to its local split-based handling of missing-value patterns.

Table 5.11: Sensitivity to missing values for TabPFN on the test set. Results are reported as mean ± standard deviation. Higher is better for AUC, while lower is better for LogLoss and Brier.

Missing rate	AUC ↑	LogLoss ↓	Brier ↓
0.0	0.8971 ± 0.0032	0.4010 ± 0.0055	0.1264 ± 0.0021
0.1	0.8904 ± 0.0029	0.4121 ± 0.0052	0.1304 ± 0.0020
0.2	0.8870 ± 0.0031	0.4175 ± 0.0057	0.1323 ± 0.0020
0.3	0.8839 ± 0.0028	0.4226 ± 0.0052	0.1342 ± 0.0018
0.4	0.8796 ± 0.0021	0.4299 ± 0.0039	0.1369 ± 0.0014

Table 5.12: Sensitivity to missing values for XGBoost on the test set. Results are reported as mean ± standard deviation. Higher is better for AUC, while lower is better for LogLoss and Brier.

Missing rate	AUC ↑	LogLoss ↓	Brier ↓
0.0	0.8957 ± 0.0026	0.4031 ± 0.0050	0.1271 ± 0.0019
0.1	0.8877 ± 0.0023	0.4168 ± 0.0043	0.1321 ± 0.0017
0.2	0.8834 ± 0.0015	0.4243 ± 0.0034	0.1349 ± 0.0011
0.3	0.8789 ± 0.0022	0.4320 ± 0.0044	0.1377 ± 0.0016
0.4	0.8739 ± 0.0015	0.4404 ± 0.0033	0.1407 ± 0.0013

Based on the synthesis of four experiments, it can be observed that under conventional data conditions, the overall performance of TabPFN is comparable to that of XGBoost.

However, under perturbations such as insufficient data and increased label noise, TabPFN demonstrates greater advantages; while under perturbations involving exacerbated feature missingness, XGBoost is more stable. Therefore, in real-world scenarios, where risks are more likely to stem from limited samples and label uncertainty, TabPFN exhibits smaller performance degradation and more stable probability mass under these two types of perturbations. Conversely, the issue of feature missingness is better addressed through engineered solutions involving data pipelines and missing value handling strategies, thereby making TabPFN's advantages more aligned with the deployment constraints and risk priorities emphasized in this study.

6 Explainable AI

In the actual credit risk control process, the value of a model extends beyond improving offline metrics to its ability to translate predictive scores into stable actionable strategies. The primary concern in this task is determining how many applications can be automatically approved, how many should be automatically rejected, and how many must undergo manual review. Therefore, this section applies the interpretability framework defined in Section 4.2 to the calibrated TabPFN model. Specifically, this section introduces global feature importance in Section 6.1, high-confidence binning based on Pareto-optimal threshold selection in Section 6.2, and rule distillation via shallow decision trees in Section 6.3.

6.1 Global Feature Importance

TabPFN's inference mechanism relies on attention weights and does not provide a native feature attribution interface. To address this, two complementary methods are employed to independently evaluate feature importance, with rankings subsequently determined through rank-sum fusion.

6.1.1 Spearman Screening

The primary objective of global interpretation is to identify which information the model relies on to make decisions. To this end, a Bootstrap Spearman screening procedure is applied: for each feature, the Spearman rank correlation coefficient is computed with respect to the model-predicted probability on the validation set, while conducting 200 bootstrap resamples in parallel. This strategy captures the monotonic association between individual variables and the model output without training additional surrogate models, and runs in $O(n \log n)$ time due to ranking operations.

As shown in Fig. 6.1, the feature importance distribution exhibits a pronounced long-tail pattern, with the mean Spearman absolute correlation coefficients for the top 5 features ranging from 0.59 to 0.71, while the 20th feature drops to approximately 0.20. This indicates that a small number of core features dominate the predictive behavior of TabPFN. Bootstrap boxplots reveal that the rankings of the top 5 features remain highly stable across 200 resamples, with interquartile ranges narrower than 0.02, indicating reliable scoring.

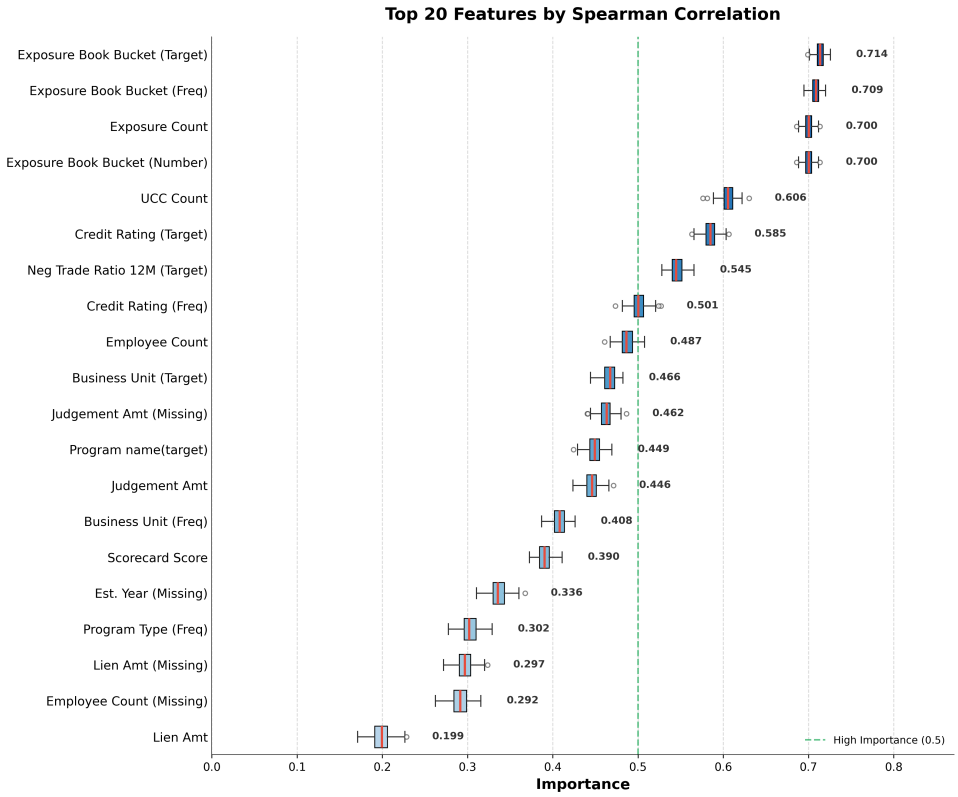


Figure 6.1: Spearman importance.

6.1.2 SHAP

To further observe the contribution of each feature to the final outcome under the influence of other features, SHAP analysis is introduced. First, this study trained a gradient-boosted regressor as a proxy model using TabPFN's prediction probabilities on the validation set as the regression target, in order to validate the proxy model's fidelity. Subsequently, TreeShape was applied to the entire validation set to capture feature interactions and provide directional information. Positive and negative SHAP values indicate whether a feature increases or decreases the probability, respectively. This yielded local SHAP attribution values for each sample across every feature. Visually.

As shown in Fig 6.2, Higher risk grades increase the probability of decline; higher credit assessment scores reduce the probability of decline; higher values for characteristics such as exposure amount and legal records correspondingly increase the probability of decline, consistent with business intuition.

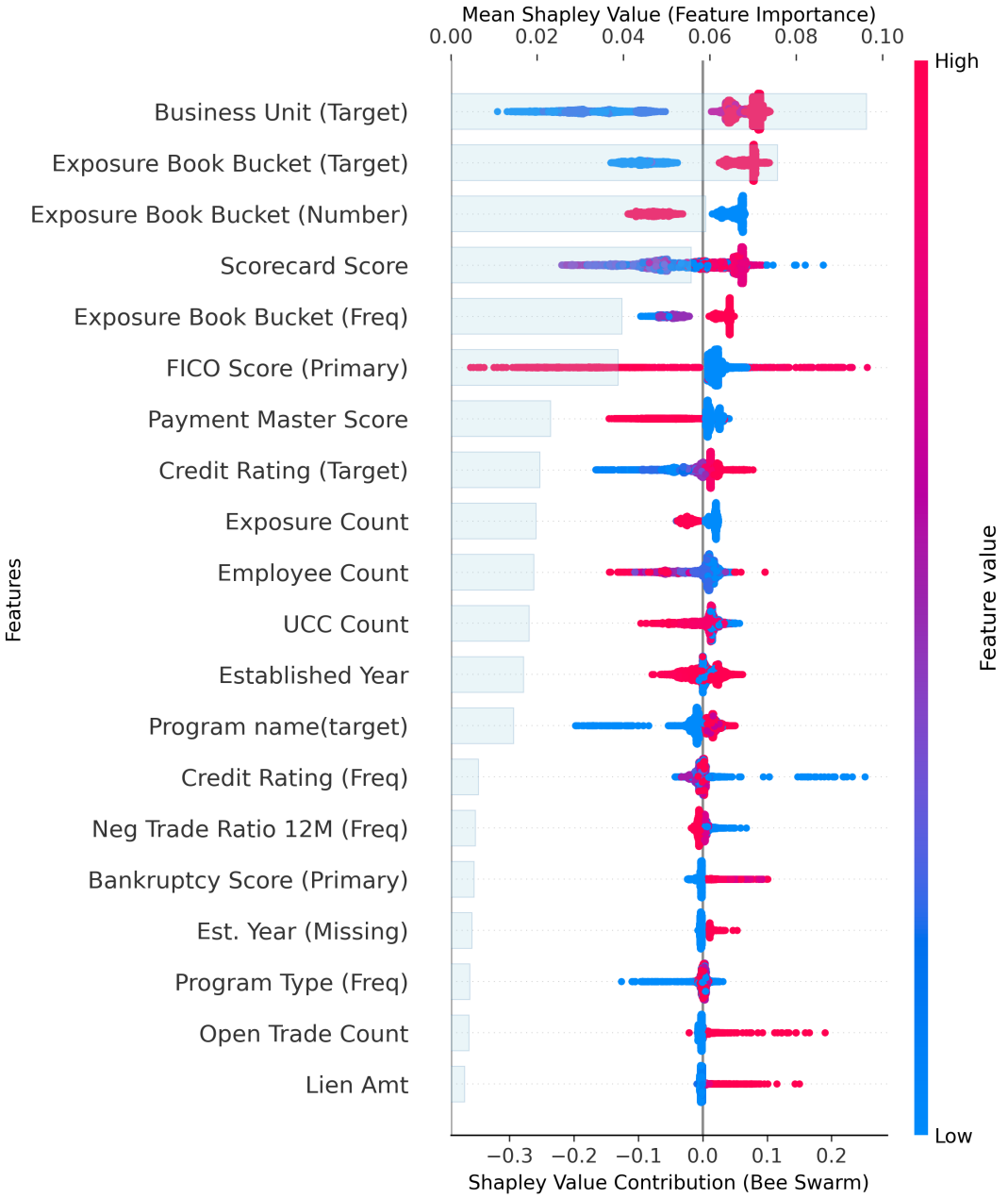


Figure 6.2: Shap Value Rank.

Combine the two ranking methods by summing the Spearman rank and SHAP rank positions for each feature, then sort the results in ascending order of rank sum. This approach mitigates the risk of results being dominated by a single method.

As shown in Fig 6.3 and combined results,

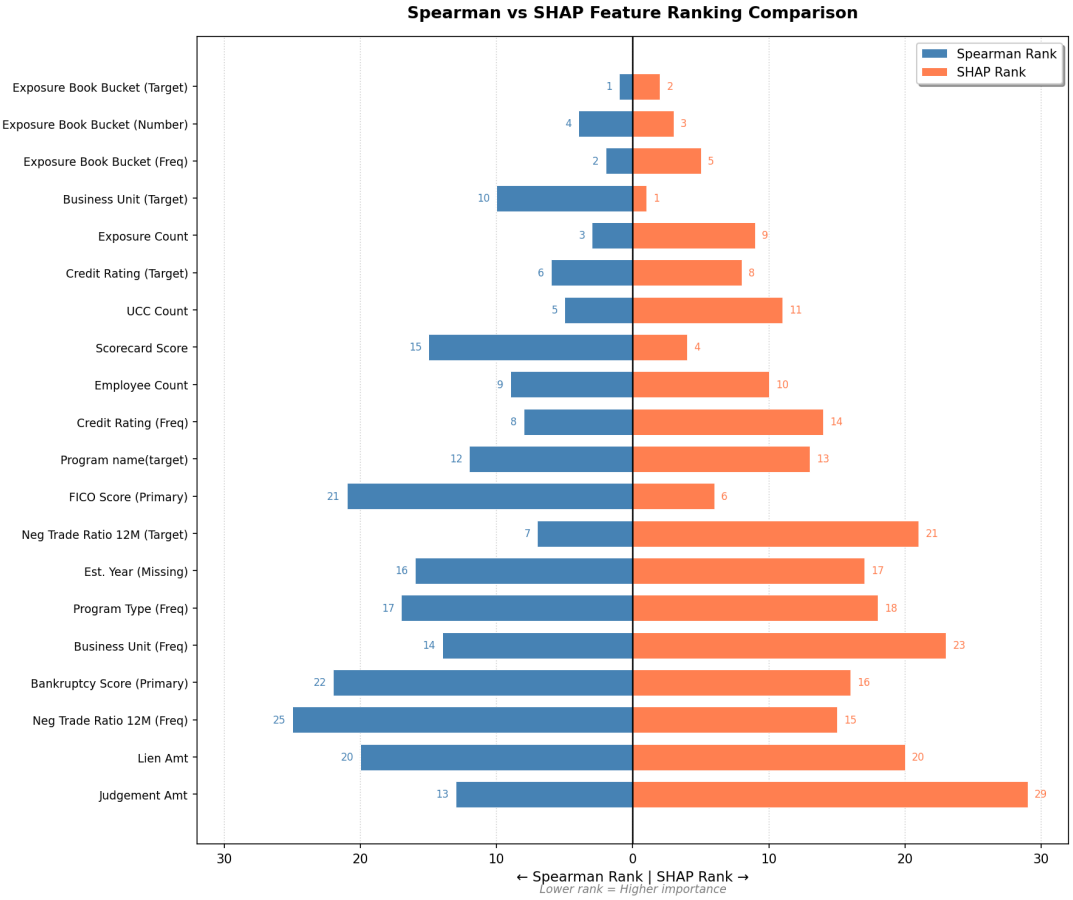


Figure 6.3: Rank Sum Result.

three patterns emerge from the ranking comparisons. First, features consistently rank highly in both rankings with extremely low rank sums. This indicates that such features not only exhibit stable monotonic marginal associations with model outputs but also retain significant irreplaceable contributions even after controlling for other variables. These features represent the primary direction of the prediction function;

Second, features that rank relatively low in Spearman but rise significantly in SHAP reflect effects more likely manifested through nonlinear thresholding or interactive structures with other signals. Since Spearman evaluates only univariate marginal monotonicity, the marginal relevance of such interaction-driven variables tends to be weaker, making them prone to underestimation in feature selection; In contrast, tree-based proxy models explicitly capture interactions through conditional splits. TreeSHAP attributes interaction gains back to relevant variables during sample-level attribution, elevating their prominence in SHAP rankings. This phenomenon suggests the model's discriminative mechanism relies not on linear superposition of independent marginal signals, but on conditional dependency structures and hierarchical decision logic.

The third category of features ranks highly in Spearman but declines in SHAP, indicating their strong marginal correlation likely stems primarily from shared information with other variables. When the model makes predictions in the joint feature space, the independent gain of such variables is shared by correlated variables, leading to reduced weighting in SHAP's contribution decomposition. This result suggests that relying solely

on marginal correlation amplifies the importance of redundant signals, thereby impairing the ability to identify true decision drivers.

In summary, relying solely on Spearman screening omits interaction-driven features, while SHAP alone may overemphasize the approximation quality of proxy models and remain sensitive to fitting biases. Rank-sum fusion mitigates the risk of systematic omissions inherent in either method by intersecting their respective bias directions, making it a more suitable foundation for subsequent feature selection and simplification.

6.2 Decision Bucket and Automation Analysis

To address the operational question of which applications can be automatically approved or declined, and which ones require manual review, this study converts TabPFN's predicted decline probability $\hat{p}$ into three decision buckets. Specifically, applications are assigned to **Auto-Accept** when $\hat{p} \leq 1 - \tau$, to **Auto-Decline** when $\hat{p} \geq \tau$, and to **Manual Review** otherwise, i.e., $1 - \tau < \hat{p} < \tau$. This bucketing mechanism maps continuous scores to actionable decisions, enabling direct quantification of automation coverage and error exposure.

The threshold τ is selected using a safety-constrained optimization criterion. On the validation set, τ is swept over $\tau \in [0.50, 0.99)$ with a step size of 0.01. The search enforces that the purity of the Auto-Accept bucket (fraction of true Accepts within the bucket) and the purity of the Auto-Decline bucket (fraction of true Declines within the bucket) are both at least 90%. Among all feasible thresholds satisfying this two-sided purity constraint, the optimal τ is chosen to minimize the proportion of samples assigned to Manual Review. This procedure is conducted exclusively on the validation set to avoid selection bias; the test set is used only to evaluate threshold transferability and operational performance.

The optimal threshold on the validation set is $\tau^* = 0.84$. When applied to the test set, the high-confidence acceptance bucket achieves 27.3% coverage with 90.5% purity, and the high-confidence rejection bucket achieves 29.4% coverage with 93.7% purity; the remaining 43.3% of cases are routed to manual review. Overall, 56.7% of applications (5,772 cases) can be decided without human intervention, substantially reducing manual workload and demonstrating the practical translation from probabilistic prediction to process automation.

Based on the results in Table 6.1, in terms of error exposure, this threshold yields 264 false negatives (FN; true Decline but automatically approved, 2.6% of the test set) and 188 false positives (FP; true Accept but automatically declined, 1.8% of the test set). These error types are asymmetric in business impact: FNs correspond to potential credit losses and risk spillover and are typically more costly, whereas FPs primarily represent opportunity costs and can be mitigated via review/appeal mechanisms or by adopting a more conservative threshold.

Table 6.1: High-confidence decision buckets on the test set at $\tau = 0.84$. Coverage denotes the proportion of samples assigned to each bucket; purity denotes the proportion of correct decisions within the bucket (Accept purity for High-conf A, Decline purity for High-conf D).

Bucket	n	Coverage (%)	Purity (%)
High-conf A (Auto-Accept)	2,778	27.27	90.50
High-conf D (Auto-Decline)	2,994	29.39	93.72
Mid/Review (Manual)	4,415	43.34	–

6.3 Efficiency and Pareto Trade-offs in Threshold

The choice of the threshold τ inherently reflects a Pareto trade-off between automation efficiency and decision safety. As τ increases, the system triggers auto-approval and auto-decline only under higher confidence, which reduces the overall error rate but increases the proportion of applications routed to manual review. Conversely, lowering τ expands automated coverage at the cost of higher error exposure. To characterize this trade-off, τ is swept over the candidate range $\tau \in [0.50, 0.98]$ on the test set. For each threshold, bucket purity, review rate, the numbers of false approvals (FN) and false declines (FP), and the overall error rate are recorded.

As shown in Figure 6.4 and Table 6.2, the efficiency–security trade-off exhibits clear diminishing marginal returns. For example, when τ increases from 0.70 to 0.84, the review rate rises by 25.5 percentage points, whereas false negatives (FN) decrease from 568 to 215, corresponding to a 62% reduction. At the same time, the overall error rate drops from 10.8% to 4.7%. This suggests that, within a moderate threshold range, a limited reduction in automated coverage can yield substantial risk reduction. However, when the threshold is further increased to $\tau = 0.90$, the review rate increases by an additional 16.1 percentage points, while FN decreases by only 127 cases, indicating that marginal safety gains begin to diminish.

Under the dual constraints on bucket purity and coverage described in the previous section, $\tau = 0.84$ achieves the lowest review rate among all feasible thresholds. It can therefore be regarded as a Pareto-optimal operating point, as it maximizes automation coverage while satisfying the minimum safety requirements.

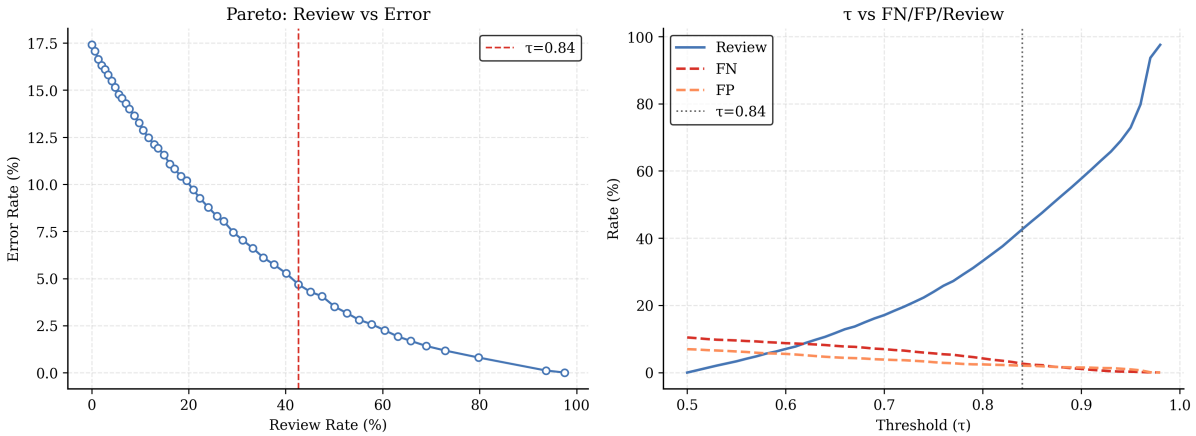


Figure 6.4: Efficiency trade-off across thresholds τ.

Table 6.2: Operational metrics of decision buckets under different thresholds τ on the test set. Purity is reported for the Auto-Accept and Auto-Decline buckets; Review denotes the proportion routed to manual review. FN indicates false approvals, where true Decline cases are auto-accepted, and FP indicates false declines, where true Accept cases are auto-declined.

τ	Accept purity (%)	Decline purity (%)	Review (%)	FN	FP	Error (%)
0.70	84.7	89.7	17.1	568	314	10.8
0.75	86.1	91.2	24.0	465	251	8.8
0.80	87.9	92.5	33.2	342	197	6.6
0.84	90.4	93.2	42.6	215	167	4.7
0.87	91.8	93.9	50.1	146	140	3.5
0.90	93.5	94.5	58.7	88	116	2.5
0.95	96.3	96.7	76.1	31	53	1.0

To further examine the operational implications of data quality degradation on the deployment strategy, the Pareto frontier is recomputed under two perturbation regimes: label noise and feature missingness.

As shown in Figure 6.5, label noise exerts a pronounced and asymmetric effect on the Pareto frontier. Under clean labels ($\eta = 0$), the optimal threshold $\tau^* = 0.83$ achieves a review rate of 40.8% with an error rate of 4.8%. At 5% label noise, the frontier shifts rightward and upward. As a result, the system must adopt a substantially more conservative threshold ($\tau^* = 0.91$) to preserve bilateral purity above 90%, raising the minimum feasible review rate to 60.7%. At 10% noise, the review rate reaches 80.4%, indicating that label uncertainty severely constrains automation coverage.

Beyond this point, the bilateral purity constraint ($\geq 90\%$) becomes infeasible under symmetric noise injection. The system therefore falls back to maximizing the minimum purity across both buckets, causing the frontier to exhibit non-monotonic behavior. Specifically, the optimal threshold recedes to $\tau^* = 0.94$ at 15% and 20% noise, partially reducing the review rate to 68.5%, while error exposure continues to rise from 5.5% to 7.0%. This non-monotonicity reflects a structural transition in the constraint landscape rather than an improvement in decision quality, and underscores the fragility of purity-constrained

automation under label uncertainty.

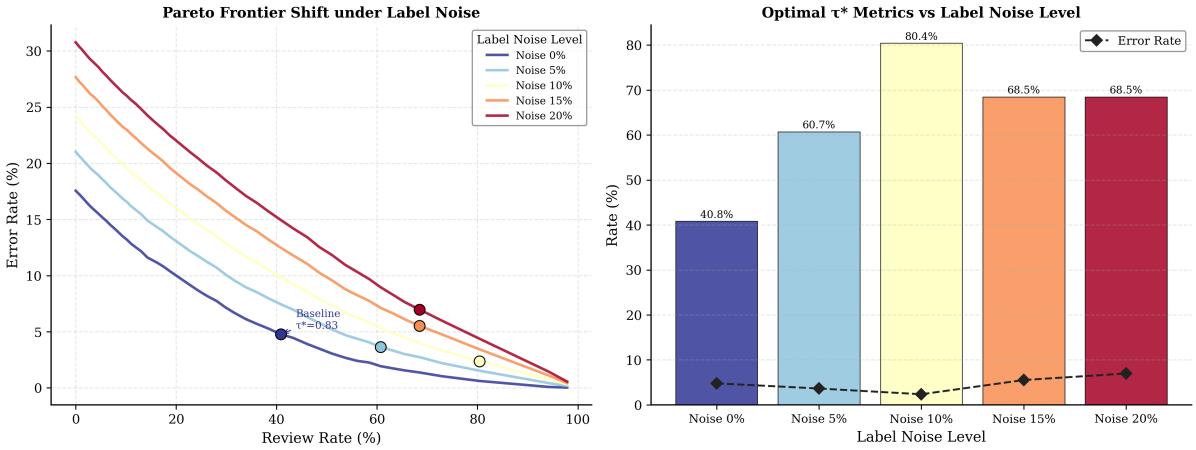


Figure 6.5: Pareto frontier shift under label noise. Left: multi-curve overlay showing how the frontier moves as label noise increases from 0% to 20%. Right: optimal review rate and error rate at τ^* under each noise level.

By contrast, as shown in Figure 6.6, the Pareto frontier demonstrates remarkable stability under increasing feature missingness. The optimal review rate varies only between 40.8% and 43.1% across all missing-rate conditions from 0% to 40%, and the error rate remains essentially unchanged at approximately 4.7%. The optimal threshold τ^* shifts only modestly from 0.83 to 0.80, a negligible change compared with the dramatic frontier displacement observed under label noise.

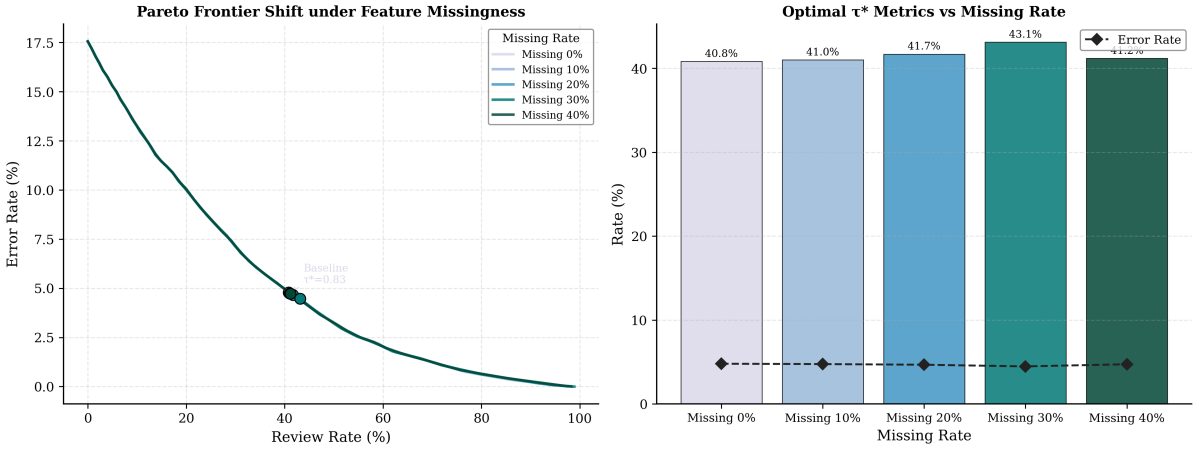


Figure 6.6: Pareto frontier shift under feature missingness. Left: frontier curves under missing rates from 0% to 40%. Right: optimal τ^* metrics remain stable across all missing-rate conditions.

Overall, under the predefined bilateral purity constraints, $\tau = 0.84$ provides the best operational compromise. Among all feasible thresholds, it achieves the lowest manual review rate while maintaining both high-confidence decision buckets above the required purity level. It can therefore be regarded as the preferred Pareto-optimal operating point.

Furthermore, label quality emerges as the primary factor affecting the stability of automated decision-making. Injecting 5% label noise increases the required manual review

rate by approximately 20 percentage points, whereas increasing the feature missing rate to 40% adds less than 3 percentage points to the review burden. In practical deployment, this suggests that practitioners should prioritize label quality assurance over further reductions in feature-level missingness.

Finally, the threshold selected on the validation set transfers well to the test set, where the purity of both the auto-accept and auto-decline buckets remains above 90%. This cross-split stability is consistent with the probability quality results reported earlier. Since the model produces relatively stable probability distributions, the same threshold can maintain similar operational behavior on unseen data, thereby reducing the risk of threshold overfitting during deployment.

6.4 Rules Distillation

To transform model outputs into deployable, auditable decision logic, this study further examines whether TabPFN's behavior can be compressed into a set of stable, easily compliant IF-ELSE rules with a simple structure. Compared to directly deploying black-box models, rule-based formats offer significant advantages in regulatory reporting, credit auditing, and system integration.

This study designs a two-stage distillation framework to approximate TabPFN's three-tier decision mechanism. First, based on the aforementioned high-confidence threshold strategy, prediction probabilities of validation set samples are mapped into three labels: Auto-Accept, Review, and Auto-Decline. This transforms the distillation objective from continuous probability regression to discrete decision boundary learning. Subsequently, shallow decision trees are trained on an expanded feature space to fit pseudo labels. This expanded feature space comprises two components: key foundational features selected through global importance fusion, and a small number of high-discriminative ratio interaction features that encode the joint relationship between two foundational signals using a single variable.

To enhance rule stability, cost complexity pruning is introduced during distillation tree training. Optimal pruning strength is selected via grid search, with a large minimum leaf node sample size set to prevent fragile rules covering only a tiny fraction of samples. Additionally, the first-layer decision tree distinguishes between Accept and Others, while the second-layer tree further divides Others into Reviews and Declines.

Distillation results indicate(table 6.3), the figure 6.7 provides a structural view of the distilled decision trees, showing that shallow trees can reproduce TabPFN's ternary decisions with high precision. The first-layer distillation achieves strong training-to-holdout consistency at limited depth, with a training accuracy of 92.2%, a holdout accuracy of 90.9%, and a Macro-F1 score of 0.899. Cost-complexity pruning effectively suppresses overfitting and improves cross-sample stability of the resulting rules. The second-layer distillation is more compact, yielding a holdout accuracy of 93.2% that does not fall below its training accuracy of 92.3%, indicating stronger separability and consistency within the non-approved subspace. After composing the two layers, the distilled rule set reaches an overall fidelity of 86.6% on the validation set and 85.8% on the test set with respect to TabPFN's original three-bucket decisions, i.e., approximately 86% of samples receive decisions consistent with TabPFN.

Table 6.3: Two-stage distilled decision tree results. Layer 1 separates Auto-Accept (A) from Others (Review + D); Layer 2 further separates Auto-Decline (D) from Manual Review within the non-A subset.

Layer	Task	Depth	Leaves	CCP-α	Train Acc.	Hold-out Acc.	Macro-F1
Layer 1	A vs. Other	6	20	0.0015	92.2%	90.9%	0.899
Layer 2	D vs. Review	3	7	0.0021	92.3%	93.2%	0.923

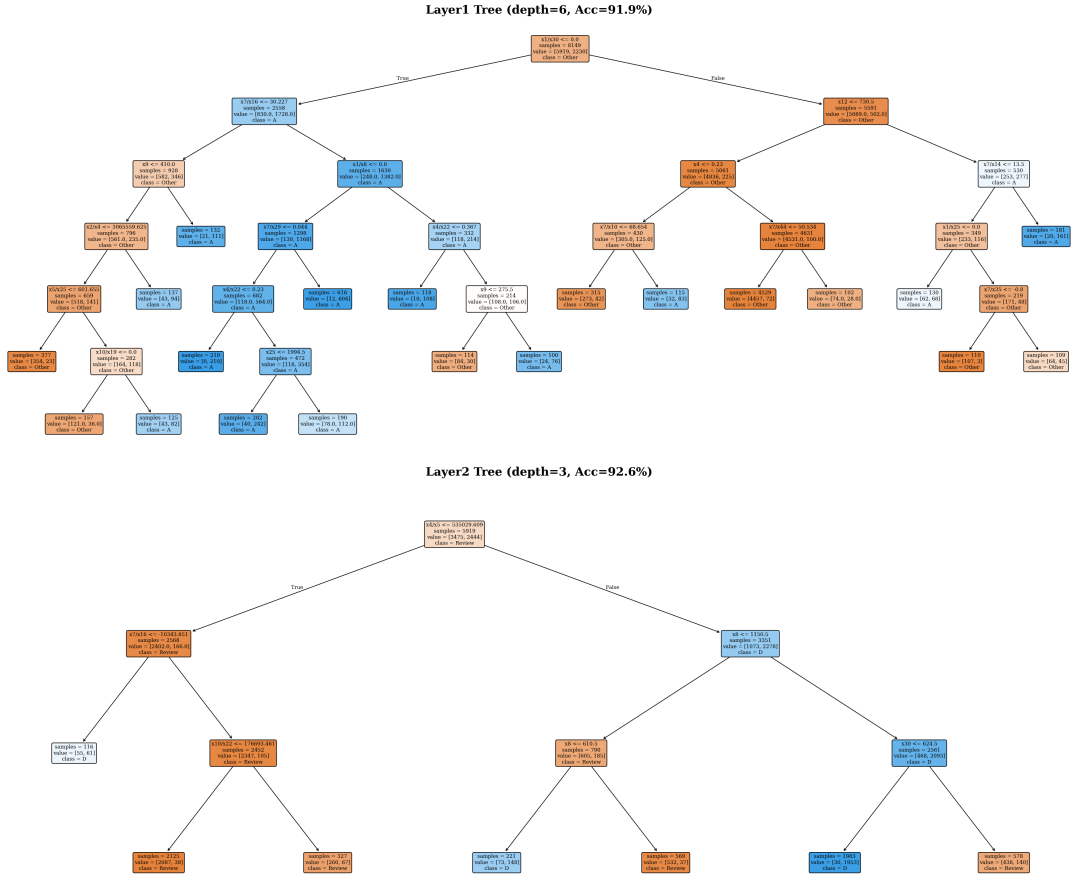


Figure 6.7: Top: First-level tree structure. Bottom: Second-level tree structure.

Further analysis shows that the fidelity loss is primarily concentrated in the *Review* bucket. This bucket corresponds to the model's uncertainty region, where samples are more dispersed and the decision boundary is more fragmented; consequently, shallow trees cannot finely approximate this region without substantially increasing model complexity.

Meanwhile, the distilled trees contain multiple high-purity leaf nodes that can be directly translated into deployable `IF--ELSE` rule fragments. Their purity remains consistently above the business-defined threshold, enabling an interpretable high-confidence automation subsystem for operational use.

Nevertheless, rule distillation has unavoidable limitations. First, approximately 14% of samples exhibit decision discrepancies between the distilled rules and TabPFN; without explicit monitoring of these disagreement cases, rule-based deployment may introduce

systematic bias. Second, although the current rule set is already far less complex than the original model, stricter regulatory requirements may demand further simplification, which would inevitably reduce fidelity. Accordingly, this study positions the distilled rules as an auditable, high-confidence subsystem—maximizing automatable coverage while preserving interpretability, and providing a structured foundation for rule monitoring and iterative refinement in deployment.

7 Discussion

Challenge One focuses on model performance degradation under highly sparse data, while Challenge Two addresses the gap between model predictions and real-world business deployment. Experimental results demonstrate that TabPFN exhibits distinct strengths and weaknesses compared to traditional machine learning methods in both challenges. It neither comprehensively outperforms existing machine learning approaches under all conditions nor falls short of them; rather, it demonstrates complementary value in specific data-driven business scenarios.

7.1 Discussion of Challenge 1: Performance degradation under highly sparse data

The core motivation of this study is to address non-random missingness and extreme sparsity in real-world business scenarios. Constraints arising from external data query costs, privacy isolation requirements, and business process limitations jointly lead to a structural imbalance in feature availability. This missingness pattern deviates from the random missingness assumption commonly adopted in traditional machine learning, thereby imposing higher requirements on model robustness.

The experimental results show that, under the current data conditions, TabPFN achieves performance on the test set comparable to that of the strongest baseline, XGBoost. From a mechanistic perspective, TabPFN is explicitly exposed to missingness patterns during its synthetic-data pretraining stage through random masking mechanisms. As a result, it develops prior robustness to sparse inputs and is able to aggregate effective signals from sparse features through bidirectional attention, without relying on explicit imputation.

However, there exists a subtle interaction between this prior advantage and the specific pseudo-noise structure identified in the financial dataset. On the one hand, the continuous monetary variables in the dataset are pre-discretized into coarse-grained intervals. This is structurally highly consistent with the quantization and discretization mechanisms used when structural causal models generate synthetic data during TabPFN pretraining. TabPFN's prior therefore already contains experience in handling discretized features, which partly explains the model's relative tolerance to the information loss introduced by binning.

On the other hand, the treatment of true zero values and missing values as a single merged category in the dataset constitutes a form of pseudo-noise that lies beyond the pretraining distribution of TabPFN. In the synthetic data used for pretraining, missing values appear as explicit random masks, allowing the model to exploit missingness itself as a predictive signal. In the financial dataset considered in this study, however, this semantic distinction has already been erased during data collection, preventing the model from distinguishing the fundamentally different credit-risk implications of the two cases. This semantic ambiguity has a relatively limited impact on XGBoost, because XGBoost handles missing values through learned default split directions and does not rely on a

precise distinction in the semantics of missingness.

The feature-missingness sensitivity experiments further reveal the performance boundary of TabPFN. When an additional 10% to 40% missingness is introduced, the degradation in AUC for TabPFN is significantly larger than that of XGBoost. This difference is not accidental, but rather a direct manifestation of the core architectures of the two models.

During inference, the bidirectional attention mechanism of TabPFN performs global similarity computation between test samples and training samples. Through a cross-attention mechanism, a test sample queries all training samples, determines which training samples are similar to itself, and refers to their labels for prediction. This process considers all feature dimensions simultaneously and is therefore global in nature.

When a large number of samples appear superficially similar in the feature space due to shared missingness in certain features, the attention weights may be dispersed across training samples that are not truly similar in terms of risk. Consequently, the quality of effective neighbors declines, weakening predictive performance. More importantly, the weights of TabPFN remain fixed after pretraining, and the model cannot locally adapt to the missingness structure of the current input during inference. Once the missingness distribution at inference time deviates from that observed during training, this implicit dependence on the global feature-availability structure is amplified and becomes a major source of performance degradation.

The operating mechanism of XGBoost is different. At each internal node of each tree, the splitting decision considers only the subset of samples reaching that node and selects a locally optimal split point based on the currently available features. For missing values, XGBoost learns a default direction for each split point during training; during inference, samples with missing values are routed along this pre-learned direction, without requiring any recomputation of global information.

When additional features are missing at inference time, the affected samples simply follow the pre-learned default paths at each node. The resulting degradation is local and node-level, and does not propagate through the global structure of the model. Although this local greedy splitting strategy is theoretically less precise than Bayesian inference from a statistical perspective, it provides precisely the local adaptivity that TabPFN lacks, enabling XGBoost to maintain more stable performance when missingness patterns drift.

In sharp contrast to its sensitivity to missingness, TabPFN exhibits substantially stronger robustness under label-noise perturbation. When the training label noise increases from 0% to 20%, the decline in AUC for TabPFN remains consistently smaller than that of XGBoost. Probability-quality metrics such as LogLoss and Brier Score also remain more stable.

From an architectural perspective, the inference process of TabPFN can be understood as an approximation of the Bayesian posterior predictive distribution. During inference, the model implicitly performs a weighted averaging over all possible underlying functions, where the weights are jointly determined by the pretrained prior and the training-set context.

When a small proportion of training labels is flipped by noise, these noisy samples constitute only local outliers within the overall training distribution. The Bayesian averaging

mechanism naturally dilutes their influence: the regions of the function space corresponding to noisy labels receive lower weights and are insufficient to dominate the final posterior prediction, thereby counteracting noise perturbations.

In contrast, XGBoost minimizes the loss function by sequentially fitting residuals, with each iteration actively correcting the discrepancy between the previous prediction and the observed label. When noisy labels are present in the training set, the model treats them as genuine signals that need to be explained and allocates additional model capacity to fit these errors. As a result, the influence of noise is gradually embedded into the tree structures, leading to relatively higher sensitivity to label noise.

Therefore, the Bayesian posterior approximation mechanism of TabPFN provides a natural advantage in scenarios where label quality is unstable but the feature structure remains relatively stable. Conversely, its global attention structure becomes a disadvantage when the distribution of feature availability shifts.

Another notable advantage of TabPFN appears under small-sample conditions. When only 10% of the training data is used, TabPFN achieves substantially better predictive performance than XGBoost, which is consistent with its theoretically expected few-shot generalization capability. This property stems from the rich prior knowledge accumulated by TabPFN during pretraining, allowing it to provide reliable predictions in new scenarios with limited labeled data. This is of practical significance for early-stage business deployment scenarios where historical data accumulation is still insufficient.

Taken together, in practical business deployment, TabPFN is a more suitable choice when feature coverage is relatively stable, label quality is uncertain, and the amount of available historical data is limited. When systematic shifts in feature-missingness patterns exist between training and inference, it is advisable to first reduce distributional drift through data-pipeline optimization and feature-engineering strategies, while considering XGBoost as an alternative model that is more robust to missingness drift.

7.2 Discussion of Challenge 2: The gap between model predictions and operational deployment

Another challenge lies in converting probabilistic model outputs into interpretable, auditable business rules. In financial services scenarios, even if a model achieves high metrics in offline evaluations, its practical business value remains limited if its outputs cannot be embedded into automated systems. To bridge this gap, this study constructs a comprehensive deployment pipeline across three layers—probability, strategy, and rules—encompassing probability calibration, threshold-driven binning, and rule distillation. At the probability calibration level, experimental results demonstrate that TabPFN's raw probabilities exhibit superior calibration properties compared to typical black-box models requiring significant recalibration for threshold-based decisions. Due to its training paradigm based on Bayesian posterior prediction approximation, TabPFN tends to produce distribution-level reliable probability estimates.

At the threshold optimization layer, results show that at the optimal threshold $\tau^* = 0.84$, the system can route 56.7% of applications to automated decisions, with only 43.3% entering manual review. Further Pareto frontier analysis indicates that increasing the

threshold from $\tau = 0.84$ to $\tau = 0.90$ raises the manual review rate by an additional 16.1 percentage points, yet reduces false negatives by only 127 cases. This demonstrates that the marginal safety gains from further tightening the strategy at high threshold intervals have significantly diminished. At the rule distillation level, after training shallow decision trees using TabPFN's three-bucket decision as a supervisory signal, the distilled rules achieved approximately 86% fidelity on the test set. Specifically, the first-layer tree achieved 90.9% test accuracy, while the second-layer tree reached 93.2%. The resulting IF-ELSE rule fragments can be directly embedded into the decision system. Fidelity loss primarily occurs in the manual review segment, where sample boundaries are more granular and distributions more dispersed. Shallow trees struggle to achieve precise fitting without significantly increasing complexity. Operationally, these samples should inherently undergo manual review, making this error relatively minor in practical impact.

Finally, in actual production deployment, τ^* is determined by the validation set and transferred to the test set, with the default data distribution being relatively stable. If the actual data changes over time, the threshold should be recalibrated periodically. Rule distillation is also a static approximation of model behavior at a specific point in time; recalibration is required when changes occur.

7.3 Limitations and Future Work

Although this study ultimately yielded results with practical value, certain limitations remain, which are categorized as follows:

1. **Insufficient dataset scope and domain diversity.** All experiments in this study were conducted using a proprietary dataset from DLL's regional operations. Its business context remains confined to asset financing, exhibiting significant data gaps compared to scenarios such as consumer credit and mortgage lending. Therefore, future work may consider incorporating datasets from diverse backgrounds for multi-faceted validation.

2. **Lack of robustness assessment against temporal distribution drift.** The study employed random stratified sampling for training without explicitly modeling distribution shifts over time. In real corporate operations, macroeconomic fluctuations, product strategy adjustments, and evolving customer demographics may trigger conceptual drift, compromising long-term reliability. Future work should incorporate time-sliced validation schemes and further explore model stability and maintainability in dynamic business environments.

3. **Approximation limitations of explainability methods.** The XAI analysis approximates TabPFN's probabilistic outputs via a proxy model, where fitting quality directly determines attribution credibility. Current validation relies solely on predictive fit metrics, leaving room for interpretive bias. Future research should explore native explainability frameworks for Transformer-based attribution methods to reduce proxy model dependency and enhance usability.

8 Conclusion

This study focuses on two core challenges arising from real financial transaction data: first, the systematic degradation of model performance under conditions of high sparsity and structural data missingness; second, the implementation gap between probabilistic prediction outputs and auditable, executable business decision-making processes.

To address these challenges, this study proposes an end-to-end evaluation and deployment framework for highly sparse credit data, comprising three complementary key components:

1. A unified preprocessing pipeline that evaluates and addresses structural sparsity, pseudo-noise, and feature heterogeneity through standardized methods;

2. A multidimensional robustness testing system that characterizes model generalization performance across four dimensions: random seed stability, data scale sensitivity, label noise injection, and feature missing perturbations;

3. An explainability-driven deployment strategy that identifies key factors through Spearman screening and SHAP rank fusion, constructs three-tier decision buckets using Pareto optimal threshold search, and further compresses model behavior into directly integrable IF-ELSE rules via two-stage shallow decision tree distillation.

Experimental results demonstrate that TabPFN achieves comparable performance to XGBoost under these data conditions, indicating that traditional machine learning models approach the upper bound of achievable performance given the current sample size and feature conditions. Further perturbation experiments reveal complementary robustness between the two models: TabPFN exhibits superior generalization on small samples, while XGBoost maintains greater stability and disturbance resilience under higher feature dropout rates.

For business implementation, this study translates probabilistic outputs into three-class decision buckets. Under safety constraints requiring bilateral purity $\geq 90\%$, the optimal threshold $\tau^* = 0.84$ is derived, enabling 56.7% of applications to be processed automatically without human intervention. Simultaneously, the two-stage rule distillation achieves approximately 86% fidelity on the test set, delivering an auditable, high-confidence rule subsystem that provides a viable pathway for embedding complex models into compliance and risk governance workflows.

Nevertheless, this study has several limitations: experiments are based solely on a single private dataset, requiring validation of cross-domain generalizability across more public or multi-institutional scenarios; long-term stability under temporal distribution drift remains unexplicitly evaluated; interpretability analysis relies on proxy models introducing approximation errors; further systematic research is needed on label selection bias and inference rejection issues. Overall, this study demonstrates the effectiveness of combining standardized preprocessing, multidimensional robustness evaluation, and explainability-driven deployment strategies. It also shows that tabular foundation models exhibit clear

advantages in scenarios with sparse annotations and unstable label quality. The proposed framework provides a reproducible, auditable, and operationally metric-oriented methodological reference for their responsible deployment in financial risk management.

References

[1] Alastair Graham and Brian Coyle. *Framework for: Credit risk management*. Global Professional Publishi, 2000.

[2] Alan N Fish. *Knowledge automation: how to implement decision management in business processes*. John Wiley & Sons, 2012.

[3] Boris Van Breugel and Mihaela Van Der Schaar. Why tabular foundation models should be a research priority. *arXiv preprint arXiv:2405.01147*, 2024.

[4] Naeem Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*, volume 3. John Wiley & Sons, 2012.

[5] Tianqi Chen. Xgboost: A scalable tree boosting system. *Cornell University*, 2016.

[6] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[7] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687, 2021.

[8] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 35(6):7499–7519, 2022.

[9] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520, 2022.

[10] Raquel Florez-Lopez. Effects of missing data in credit risk scoring. a comparative analysis of methods to achieve robustness in the absence of sufficient data. *Journal of the Operational Research Society*, 61(3):486–501, 2010.

[11] Edgar Acuna and Caroline Rodriguez. The treatment of missing values and its effect on classifier accuracy. In *Classification, Clustering, and Data Mining Applications: Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Illinois Institute of Technology, Chicago, 15–18 July 2004*, pages 639–647. Springer, 2004.

[12] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. Explainable machine learning in credit risk management. *Computational Economics*, 57(1):203–216, 2021.

[13] Edward I Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The journal of finance*, 23(4):589–609, 1968.

[14] James A Ohlson. Financial ratios and the probabilistic prediction of bankruptcy.

Journal of accounting research, pages 109–131, 1980.

[15] Bart Baesens, Tony Van Gestel, Stijn Viaene, Maria Stepanova, Johan Suykens, and Jan Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society*, 54(6):627–635, 2003.

[16] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[17] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[18] Yufei Xia, Chuanzhe Liu, YuYing Li, and Nana Liu. A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert systems with applications*, 78:225–241, 2017.

[19] Björn Rafn Gunnarsson, Seppe Vanden Broucke, Bart Baesens, María Óskarsdóttir, and Wilfried Lemahieu. Deep learning for credit scoring: Do or don't? *European Journal of Operational Research*, 295(1):292–305, 2021.

[20] Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, and Lyn C Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European journal of operational research*, 247(1):124–136, 2015.

[21] Iain Brown and Christophe Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert systems with applications*, 39(3):3446–3453, 2012.

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

[23] Håvard Kvamme, Nikolai Sellereite, Kjersti Aas, and Steffen Sjursen. Predicting mortgage default using convolutional neural networks. *Expert Systems with Applications*, 102:207–217, 2018.

[24] Ira Shavitt and Eran Segal. Regularization learning networks: deep learning for tabular datasets. *Advances in neural information processing systems*, 31, 2018.

[25] Helen-Tadesse Moges, Karel Dejaeger, Wilfried Lemahieu, and Bart Baesens. A multidimensional analysis of data quality for credit risk management: New insights and challenges. *Information & management*, 50(1):43–58, 2013.

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

[27] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.

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

[29] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 384–394, 2019.

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

[31] Walter F Wiggins and Ali S Tejani. On the opportunities and risks of foundation models for natural language processing in radiology. *Radiology: Artificial Intelligence*, 4(4):e220119, 2022.

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

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

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

[35] Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at tabpfn v2: Strength, limitation, and extension. *arXiv preprint arXiv:2502.17361*, 2025.

[36] Kai Helli, David Schnurr, Noah Hollmann, Samuel Müller, and Frank Hutter. Drift-resilient tabpfn: In-context learning temporal distribution shifts on tabular data. *Advances in Neural Information Processing Systems*, 37:98742–98781, 2024.

[37] SM Lundberg and SI Lee. A unified approach to interpreting model predictions. neurips, 2017.

[38] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[39] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[40] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[41] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento,

Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

[42] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

[43] Donald Gillies. Causality: Models, reasoning, and inference judea pearl. *The British Journal for the Philosophy of Science*, 52(3):613–622, 2001.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[45] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ROC Analysis in Pattern Recognition.

[46] Gerard Debreu. Valuation equilibrium and pareto optimum. *Proceedings of the national academy of sciences*, 40(7):588–592, 1954.

[47] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

[48] Douglas G Bonett and Thomas A Wright. Sample size requirements for estimating pearson, kendall and spearman correlations. *Psychometrika*, 65(1):23–28, 2000.