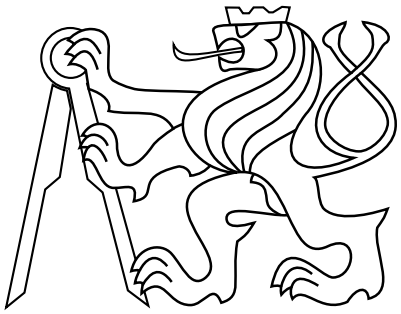
Czech Technical University in Prague

Faculty of Electrical Engineering



Explainable Classification of Financial Transactions

Semester Project

Alina Daroshka

Prague, January 2025

Study programme: Open Informatics
Branch of study: Artificial Intelligence and Computer Science

Supervisor: prof. Ing. Václav Šmídl, Ph.D.
Specialist Supervisor: Ing. Michaela Mašková

Chapter 1

Introduction

Fraud detection is a critical issue in financial transactions, impacting both consumers and financial institutions. These activities not only lead to substantial financial losses but also undermine the trust in financial systems. Fraud can manifest in various forms, such as identity theft, account takeovers and payment fraud. As technology evolves, so do the tactics employed by fraudsters. Detecting and preventing fraud on a large scale has become an increasingly complex challenge due to the sheer volume and diversity of financial transactions that must be monitored in real time.

Traditionally, fraud detection has relied on rule-based systems, manual intervention, and expert knowledge. However, these approaches often fail to adapt quickly to new fraud patterns, whilst also demanding substantial resources. In contrast, recent advances in machine learning (ML) have opened up new opportunities for automating and improving fraud detection systems. Machine learning models can learn from historical transaction data to identify patterns and anomalies that may indicate fraudulent behavior. These developments hold the potential to significantly enhance the efficiency and effectiveness of fraud detection.

1.1 Problem definition

Despite the effectiveness of machine learning in fraud detection, one of the key challenges is the lack of interpretability in many modern models. Highly accurate models often function as black boxes, making it difficult for stakeholders to understand why certain transactions are classified as fraudulent. This lack of transparency can hinder the adoption of machine learning models in the financial sector, where explainability is essential for regulatory compliance, stakeholder trust, and operational decision-making.

To address this issue, techniques for explaining machine learning models, such as SHAP (SHapley Additive exPlanations)[6], have gained significant attention. SHAP offers insights into model predictions by attributing the contribution of individual features, providing an understanding of the factors driving decisions. This study investigates the application of SHAP to explain the classifications of financial transactions made by two machine learning models: XGBoost[3] and Multilayer Perceptrons. Financial transactions are represented as feature vectors, and the models aim to effectively predict fraudulent activities.

The primary objective of this thesis is to evaluate the effectiveness of explainable machine

learning approaches in the context of financial transaction classification. By analyzing the explanations generated by SHAP for XGBoost and MLP models across two financial datasets, the study assesses their performance, interpretability, and practical applicability. This approach not only enhances understanding of the decision-making process but also promotes explainable artificial intelligence in financial applications.

1.2 State of the Art

Fraud detection in financial transactions has progressed significantly with the adoption of machine learning (ML) and deep neural networks (DNNs). While traditional methods such as logistic regression and decision trees offered interpretability, they often struggled with the complexity of modern fraud patterns. Ensemble models like CatBoost and XGBoost have emerged as top performers, achieving high accuracy, sometimes exceeding 99%, by effectively handling class imbalance and capturing feature interactions. Deep learning approaches, such as convolutional neural networks (CNNs), have also demonstrated success in detecting fraud by leveraging their ability to model complex relationships in transactional data.

The integration of explainable AI (XAI) methods has addressed the growing demand for interpretability in fraud detection models. Techniques such as SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations) have been applied to improve transparency, offering insights into how specific features contribute to a model's decisions. These methods enable compliance with regulatory requirements and support trust in automated systems by making black-box models more accessible to non-technical stakeholders.

Recent research has increasingly focused on combining high-performance ML and DNN methods with XAI techniques to balance predictive accuracy and interpretability[7]. Approaches incorporating feature importance explanations or blending gradient-boosting models with interpretability frameworks exemplify this trend. Despite these advancements, challenges remain in scaling XAI methods to large datasets and addressing the computational demands of deep learning-based fraud detection systems.

1.3 Aim of the Report

The aim of this report is to explore the application of SHAP (SHapley Additive exPlanations) in providing interpretability for machine learning models, specifically focusing on financial transaction fraud detection. The report investigates the use of SHAP to explain the predictions made by two different classifiers, XGBoost and Multilayer Perceptrons (MLP), on two distinct financial datasets. By evaluating the explanations generated for these models, the report seeks to assess their utility in providing insights into the factors influencing fraud detection decisions. This analysis aims to highlight the practical implications of explainable artificial intelligence (XAI) in the financial sector, where transparency and trust are crucial for regulatory compliance and operational decision-making.

Chapter 2

Theory

2.1 Classification

Classification is a fundamental task in machine learning where the objective is to assign a label $y \in \{1, 2, \ldots, K\}$ to an input feature vector $\mathbf{x} \in R^n$. The classifier learns a mapping function $f : R^n \to \{1, 2, \ldots, K\}$ from a labeled dataset, consisting of input-output pairs $(\mathbf{x}_i, y_i)$. This process involves training the model to recognize patterns in the feature space that differentiate between classes.

In many classification problems, the model outputs a probability distribution over the possible classes, $p(y = c \mid \mathbf{x})$, for each input. Probabilities are often computed using a softmax function:

$$p(y = c \mid \mathbf{x}) = \frac{\exp(f_c(\mathbf{x}))}{\sum_{k=1}^{K} \exp(f_k(\mathbf{x}))},$$

where $f_c(\mathbf{x})$ represents the model's score for class c, and K is the total number of classes.

The model's performance during training is optimized using a loss function. For classification, the most commonly used loss is the cross-entropy loss, which measures the divergence between the predicted probability distribution and the true label distribution. The cross-entropy loss for a single data point is defined as:

$$\mathcal{L} = -\sum_{c=1}^{K} \mathbf{1}[y = c] \log p(y = c \mid \mathbf{x}),$$

where $\mathbf{1}[y = c]$ is an indicator function that equals 1 if $y = c$, and 0 otherwise. The overall loss for a dataset is the average of these individual losses.

Classification models are evaluated using metrics such as accuracy, precision, recall, and the F1 score. These metrics quantify the model's ability to assign correct labels and are crucial for assessing performance, especially in cases of imbalanced datasets. In the context of financial transactions, classification tasks often involve distinguishing between normal and fraudulent activities, requiring robust and interpretable models to ensure reliability and trustworthiness.

2.1.1 Imbalanced Classification

Imbalanced data occurs when the distribution of classes within a dataset is unequal. In other words, one class has significantly more samples than the others. The disproportion may often arise due to the inherent nature of the problem. Some problems have a naturally skewed class distribution, such as fraud detection or rare disease diagnosis.

Training models on imbalanced datasets poses significant challenges that can undermine the reliability of predictions, particularly for the minority class. Two primary issues are:

1. **Misleading Metrics**
   Traditional evaluation metrics, such as accuracy, are often deceptive for imbalanced classification problems. A model may achieve high accuracy by predicting all instances as belonging to the majority class, effectively ignoring the minority class. This is especially problematic because the minority class typically represents the critical cases (e.g., fraudulent transactions) where correct classification is crucial. Relying solely on accuracy can lead to false confidence in a model's performance, masking its inability to detect the minority class.

2. **Bias in Learning**
   Models like neural networks and decision trees may prioritize the majority class during training, leading to poor generalization on the minority class. These models often optimize for overall error minimization, which skews the learning process toward the majority class (e.g. decision trees may split early based on majority class features).

Preprocessing Methods

Random Under-Sampler (RU)
Random Undersampling is the most straightforward method for undersampling. Instances from the majority class are randomly selected and removed to achieve a balance between the minority and majority classes. While it indeed makes models less biased and toward the majority class and simplifies the training process by reducing the dataset size, it may increase the variance of the classifier and is very likely to discard potentially useful or important samples. It leads to a great loss in available training data, and ultimately leads to reduced model performance.

Oversampling
Oversampling increases the size of the minority class by replicating its samples or generating synthetic data. The most popular technique is SMOTE (Synthetic Minority Oversampling Technique). It generates new samples by interpolating between existing minority class instances. The algorithm works as follows:

1. For each sample in the minority class:
   (a) Find its k nearest neighbors (other samples in the minority class).
   (b) Randomly pick one of these neighbors.
   (c) Create a synthetic sample by adding a small random difference between the sample and the chosen neighbor. This difference is calculated as:

$$\text{Synthetic Sample} = \text{Original Sample} + \lambda \cdot (\text{Neighbor} - \text{Original Sample}),$$

   where λ is a random number between 0 and 1.

2. Repeat the process for as many new samples as needed to balance the dataset.

There is a drawback to using SMOTE, as it does not consider the majority class while creating synthetic examples. This can cause issues where there is a strong overlap between the classes.

Class weights
Class weights assign higher weights to the minority class, allowing the model to pay more attention to its patterns and reducing bias towards the majority class. Most machine learning libraries allow you to specify class weights. May lead to overcompensation, causing false positives in the minority class.

Cost-Sensitive Learning

The idea is to assign a higher "cost" (or weight) to misclassified samples from the minority class, encouraging the model to focus on those samples during optimization.

Weighted BCE Loss:
Weighted BCE modifies the standard BCE by applying a weight to the positive class:

$$\text{Weighted BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[\alpha y_i \log(p_i) + (1-y_i)\log(1-p_i)\right].$$

Here: α (weight for positive class) is typically calculated as:

$$\alpha = \frac{\text{Number of Negative Samples}}{\text{Total Number of Samples}},$$

or set empirically.

Focal Loss[5]:
Focal loss is an extension of binary cross-entropy loss designed to address class imbalance by dynamically down-weighting the loss of well-classified samples, focusing more on hard-to-classify ones. It was originally proposed for object detection tasks but is widely used in imbalanced classification.

$$\text{Focal Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[\alpha(1-p_i)^{\gamma} y_i \log(p_i) + (1-y_i)\log(1-p_i)\right].$$

Key components:
- α: Balancing weight for the positive class (similar to weighted BCE).
- $(1-p_i)^{\gamma}$: A modulating factor that reduces the loss for well-classified samples (p_i close to 1 when $y_i = 1$).
- $\gamma \geq 0$: Focusing parameter. Larger values of γ increase the focus on hard-to-classify samples.

2.2 Classifiers

2.2.1 XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable implementation of gradient boosting algorithms, designed for performance and speed. It builds an ensemble of decision trees by iteratively training new trees to correct the errors (or residuals) made by previous ones. At each step, the algorithm minimizes a loss function using gradient descent, leading to an improved model that combines the predictions of individual trees. XGBoost includes regularization techniques to avoid overfitting, making it particularly suitable for tasks involving large datasets. Its flexibility allows for both binary and multi-class classification problems and has gained widespread popularity in various machine learning competitions due to its high performance and accuracy.

2.2.2 MLP (Multilayer Perceptron)

A Multilayer Perceptron (MLP) is a type of neural network composed of multiple layers of nodes (neurons) that are fully connected. It is a feedforward network where information flows from the input layer through one or more hidden layers to the output layer. Each connection between neurons is associated with a weight that is adjusted during training to minimize a loss function, typically using backpropagation and gradient descent. MLPs are capable of modeling complex, non-linear relationships between inputs and outputs. In classification tasks, the final layer of the MLP usually consists of a softmax function that outputs a probability distribution over the possible classes. MLPs are particularly effective for problems where feature interactions are complex and hard to model with simpler algorithms, though they often require larger amounts of data and computational power compared to tree-based methods.

2.3 Evaluation

The model will be evaluated using several metrics to measure its performance. These include commonly used metrics such as the confusion matrix, precision, recall, F1-score, ROC Curve and the Precision-Recall Curve (PRC). Choosing the right evaluation metrics can be confusing when it comes to imbalanced data, and it is relatively essential to choose the right one for model evaluation.

Accuracy

Accuracy is defined as the ratio of correctly identified occurrences comparative to the overall number of instances. The formula for quantifying binary accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = True Positive, FP = False Positive, TN = False Negative, FN = False Negative.

Despite being the most widely used metric, it is extremely misleading when it comes to highly imbalanced dataset like fraud detection in transactional data. The ratio of fraudulent transactions is usually less than 1 percent of the total data. Therefore, even if the model misclassifies all fraudulent cases, it will still achieve high accuracy.

Precision

Precision is the proportion of all the model's positive classifications that are actually positive.

$$\text{Precision} = \frac{\text{Correctly Classified Actual Positives}}{\text{Everything Classified as Positive}} = \frac{TP}{TP + FP}$$

In the fraud detection example, precision measures the fraction of transactions classified as fraud that were actually fraud.

Recall

Recall measures the model's ability to accurately predict the positive class. Intuitively, it reflects how effectively the model identifies all positive instances in the dataset.

$$\text{Recall} = \frac{\text{Correctly Classified Actual Positives}}{\text{Total Actual Positives}} = \frac{TP}{TP + FN}$$

A high recall value means that the model successfully identifies most of the fraudulent transactions, which is crucial in fraud detection to avoid missing potential fraudulent activity.

F1-score

The F1 score is the harmonic mean of the precision and recall. It is used to evaluate the balance between these two metrics in a classification model.

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

Confusion Matrix

The confusion matrix is a table used to evaluate the performance of a classification algorithm. It compares the predicted labels with the actual labels to show the number of correct and incorrect predictions made by the model. It is particularly useful for understanding the types of errors a classifier is making.

A confusion matrix for a binary classification problem is a 2x2 table that looks like this:

$$\begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix}$$

ROC Curve and AUC (Area Under Curve)

The ROC curve plots the True Positive Rate against the False Positive Rate across varying probability thresholds. The TPR is also known as recall, while the FPR indicates the proportion of actual negatives that are incorrectly classified as positives. AUC, or Area Under the Curve, is a single scalar value ranging from 0 to 1, that condenses this curve into a single value, indicating the model's ability to distinguish between classes. AUC close to 1 indicates the perfect model. AUC around 0.5 indicates that the model is making random predictions. In terms of highly imbalanced data, the ROC curve can be misleading because even with a large number of false positives, the ROC might still look good due to the overwhelming number of negatives.

Precision-recall curve (PRC)

Precision-recall curve (an alternative for the ROC curve) is a plot of the precision (positive predictive value, y-axis) against the recall (sensitivity, x-axis) for different thresholds. It significantly impacts the value of the machine learning model when false positives and false negatives have different costs. Unlike AUC-ROC, which can give a misleadingly optimistic view in imbalanced datasets, PRC focuses more on performance concerning the positive class, which is essential for fraud detection.

Chapter 3

Explanation

3.1 Need for Explanation and Interpretation

AI technology suffers from what is called a black box problem. In other words, while the input data is known, there is no visibility into the steps or processes that provide you with the output. This is especially problematic in deep learning and artificial neural network approaches, which contain many hidden layers of nodes.
Stakeholders often hesitate to trust machine learning projects because they lack an understanding of how these models function. It is especially crucial for industries where an AI prediction makes a profound impact, like healthcare or driverless cars, since the cost of the mistake is exceptionally high.

3.2 Explainability versus Interpretability

Inherently interpretable (white box) models are built using algorithms that, by design, are interpretable and allow the explanation of results at both the global and local levels. They provide their own explanations, faithful to what the model computes, whereas Explainable ML requires a second (post hoc) model to explain the first black box model. In other words, explainable ML describes why the model made a prediction, while interpretable model describes how it makes the prediction.

3.3 SHapley Additive exPlanations (SHAP)

SHAP is a unified framework for interpreting individual predictions of machine learning models. It belongs to Additive Feature Attribution Methods, meaning it breaks down the prediction into the sum of contributions from each feature, exactly matching the model's output. Since many modern machine learning models, such as neural networks or ensemble methods, are inherently opaque, SHAP provides explanations by approximating these complex models with simpler, explanation models.
SHAP methods utilize Shapley values, a concept from game theory, to assign credit for a model's prediction to each feature or feature value. The Shapley value is the average marginal contribution of a feature value across all possible coalitions. The formula is defined as:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} \left[f(S \cup \{i\}) - f(S) \right]$$

The formula calculates the contribution of feature i (ϕ_i) to the model's prediction by considering all possible subsets S of the remaining features (excluding i). For each subset S, the difference between the model's output with and without feature i ($f(S \cup \{i\}) - f(S)$) is weighted and summed over all possible feature value combinations.

SHAP values are the solutions to the above equation under the assumptions: $f(x_s) = E[f(x|x_s)]$. i.e. the prediction for any subset S of feature values is the expected value of the prediction for f(x) given the subset x_s. For more than a few features, finding the exact solution to this problem becomes challenging and computationally expensive as the number of possible coalitions grows exponentially with the addition of more features. To address this challenge, SHAP introduces approximation techniques, one of which is Kernel SHAP.
Kernel SHAP is a model-agnostic method to approximate SHAP values using ideas from Local Interpretable Model-agnostic Explanations (LIME) and Shapley values. Instead of evaluating all 2^M subsets(for a model with M features), Kernel SHAP samples a manageable number of subsets and uses their contributions to estimate the Shapley values. In the SHAP paper[6] the authors demonstrate that with a weighted linear regression model as the local surrogate model and an appropriate weighting kernel, the regression coefficients of the LIME surrogate model estimate the SHAP values. The Shapley kernel that recovers SHAP values is given by:

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}\,|z'|\,(M-|z'|)},$$

Where M is the number of features and $|z'|$ is the number of non-zero features in the simplified input z'.

3.4 Other approaches

In addition to model-agnostic explanation methods, there are inherently interpretable models that are designed to provide transparency in their predictions without requiring external explanation techniques. These models are designed in such a way that their decision-making process is easily understandable, offering insights into how individual features influence the model's output.

One such approach is the use of prototypes in machine learning. In this method, the model learns to classify new instances by comparing them to representative examples, or prototypes, of each class. These prototypes serve as central points in the feature space that best summarize the characteristics of the class they represent. By comparing a given input to these prototypes, the model assigns the input to the class of the closest prototype, providing a clear, interpretable decision-making process.

Rudin et al. [2] emphasize the advantages of inherently interpretable models and underscore the importance of designing machine learning systems that prioritize interpretability while maintaining robust performance.

Chapter 4

Experiments

4.1 Finantial datasets

4.1.1 IBM Dataset[1]

The IBM credit card fraud dataset is a synthetic dataset designed to simulate realistic credit card transaction activity, enabling research into fraud detection and modeling. The dataset consists of 24 million transactions generated for 20,000 users, with each transaction represented by a row comprising 12 distinct fields. These fields include both continuous and categorical features, such as merchant name, merchant address, transaction amount, transaction type, and timestamp, among others.
The dataset is generated using a rule-based stochastic sampling process, which ensures a diverse range of realistic transaction patterns while preserving anonymity and eliminating privacy concerns. To account for the sequential nature of transaction data, transactions for each user are chronologically ordered, capturing temporal dependencies that are crucial for fraud detection tasks

4.1.2 Amaretto Dataset[4]

The Amaretto dataset was designed as a synthetic simulation of transaction activities typical in the global capital market. It contains 29,704,090 transactions conducted by 400 end-customers over 60 days (12 weeks), with each week comprising five trading days. Transactions involve purchasing and selling various securities, such as equities and fixed income products. Key fields include transaction amounts, product class, type, time, currency, and market. Less than 1% of the data consists of anomalies, with five distinct types of suspicious patterns based on guidelines from the Financial Action Task Force (FATF). The dataset aims to replicate real-world scenarios for financial anomaly detection and analysis.

The five types of anomalies included in the dataset are:

1. **Small but highly frequent transactions within a short time frame**: These transactions occur just below reporting thresholds, often representing multiple small trades executed rapidly.

2. **Transactions with rounded normalized amounts**: These unusual trades involve amounts that are precisely rounded, which typically do not occur unless foreign exchange conversions are involved.

3. **Securities bought or sold at unusual times**: This anomaly is characterized by trades that occur outside of regular trading hours, such as before or after the opening and closing times of stock exchanges.

4. **Large asset withdrawal**: A sudden, significant withdrawal or transfer of assets from an account, typically without a commercial rationale or corporate event to justify the action.

5. **Unusually large amounts of collateral transferred in and out of an account in a short period**: This behavior is rare, as it would be uncommon for a client to engage in such transfers without a clear investment strategy.

4.1.3 Data Preprocessing

Various preprocessing techniques were explored to optimize the dataset for training, including feature engineering and handling missing data. Certain columns, such as personal information of customers and merchants, were dropped as they were considered irrelevant for fraud prediction.
Categorical features were transformed into one-hot encodings to enable compatibility with machine learning models. Continuous features were further scaled based on their distributions (e.g., standardization or normalization) to ensure numerical stability and facilitate effective training of MLP models. For the IBM dataset, features were manually selected based on domain knowledge to retain those most relevant for detecting fraudulent behavior. For the Amaretto dataset, additional transformations were applied to continuous features, including the computation of rolling statistics over temporal windows, to capture sequential trends and patterns in user behaviors.

4.2 Classifer Accuracy

4.2.1 IBM Dataset

Hyperparameter tuning

Hyperparameter tuning for XGBoost model was performed using RandomizedSearchCV, optimizing the number of estimators, learning rate, maximum depth, and subsampling rate. The best combination of parameters was as follows: **subsample**: 1.0, **n_estimators**: 300, **min_child_weight**: 3, **max_depth**: 7, **learning_rate**: 0.2, **gamma**: 0.1, **colsample_bytree**: 0.6.

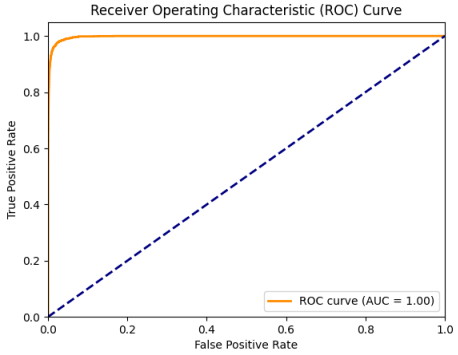
In addition to hyperparameter tuning, threshold optimization was applied to models to fine-tune the decision threshold for fraud detection.

XGBoost Model Results

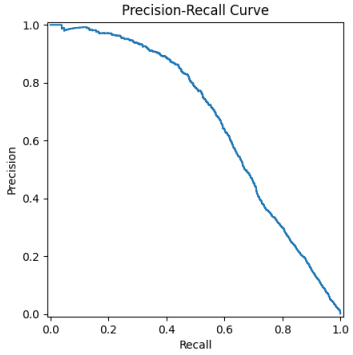
The entire dataset was used for testing, with a timesplit applied to ensure chronological consistency between training, validation, and testing sets. The results of the models trained for fraud detection are presented in terms of accuracy, precision, recall and F1-score.

Model	Accuracy			Precision			Recall			F1-Score		
	Train	Val	Test	Train	Val	Test	Train	Val	Test	Train	Val	Test
BM	0.9995	0.9992	0.9991	0.8236	0.7707	0.5543	0.7889	0.6491	0.6430	0.8059	0.7047	0.5954
Class Weights	0.9994	0.9988	0.9987	0.6994	0.6035	0.3924	0.8289	0.5022	0.5117	0.7586	0.5482	0.4442
SMOTE	0.9975	0.9990	0.9990	0.9989	0.6997	0.4991	0.9902	0.5275	0.4991	0.9945	0.6015	0.5231
Undersampling	0.9020	0.9988	0.9988	1.0000	0.5840	0.4283	0.6732	0.6391	0.6200	0.8047	0.6103	0.5067

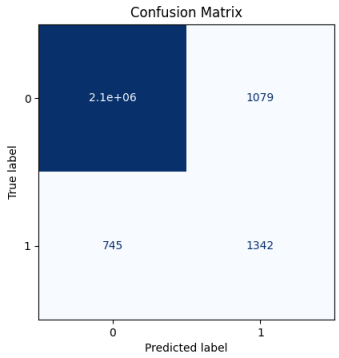
Table 4.1: Comparison of Different XGBoost Models Using Evaluation Metrics



(a) ROC Curve



(b) Precision-Recall Curve



(c) Confusion Matrix

Figure 4.1: Model Evaluation: ROC Curve, Precision-Recall Curve, and Confusion Matrix

The results presented in Table 4.1 demonstrate the performance of different techniques for addressing the class imbalance in the fraud detection problem. Despite applying methods such as class weighting, SMOTE, and undersampling, these approaches did not yield significant improvements across the evaluation metrics, particularly on the test set. In some cases, these techniques even resulted in reduced performance compared to the baseline model (BM).

Given the lack of substantial benefit observed with these imbalance-handling techniques, it was decided to proceed with the evaluation of additional datasets and models without employing such strategies. This decision ensures a focus on the primary modeling task and avoids unnecessary complexity that does not provide measurable advantages.

Multi-Layer Perceptron Model

The Multi-Layer Perceptron (MLP) model was designed with the following characteristics:

Characteristic	Value
Loss Function	Focal Loss
Hidden Layers	128, 128
Epochs	5
Learning Rate	0.0001
Optimizer	AdamW
Weight Decay	0.01
Activation Function	ReLU

Table 4.2: Characteristics of the Multi-Layer Perceptron Model for IBM Dataset

Initial results revealed that the fully connected neural network performed poorly when evaluated using a time-based split of the data. This was likely due to the issue of concept drift, where the data distribution changes over time. To address this, the data was split based on customers instead of time. Specifically, 1200 customers were used for training, 400 for evaluation, and 400 for testing. This adjustment improved the model's performance, and the following results were obtained for the best threshold of **0.4704**.

Metric	Train	Validation	Test
Accuracy	0.9996	0.9992	0.9993
Precision	0.6187	0.8232	0.7776
Recall	0.5236	0.5199	0.5149
F1 Score	0.5672	0.6373	0.6195

Table 4.3: Model performance metrics for different data splits.

4.2.2 Amaretto Dataset

Hyperparameter tuning

For the Amaretto dataset, hyperparameter tuning was not performed since the model's performance with the default XGBoost hyperparameters was found to be satisfactory.

XGBoost Model Results

The Amaretto dataset was evaluated using a multi-class classification model. The performance of the model was assessed on test data, with the results presented below.

Test Results:

Dataset	Accuracy	Precision (Weighted)	Recall (Weighted)	F1 Score (Weighted)
Test	0.9991	0.8731	0.8498	0.8508

Table 4.4: Overall Performance Metrics for Test Data

Class-Specific Metrics (Test Data):

- Class 0 - Precision: 0.9996, Recall: 0.9998, F1 Score: 0.9997
- Class 1 - Precision: 0.9036, Recall: 0.9271, F1 Score: 0.9152
- Class 2 - Precision: 1.0000, Recall: 0.9805, F1 Score: 0.9902
- Class 3 - Precision: 0.9983, Recall: 0.9864, F1 Score: 0.9923
- Class 4 - Precision: 0.5933, Recall: 0.7962, F1 Score: 0.6800
- Class 5 - Precision: 0.7439, Recall: 0.4086, F1 Score: 0.5275

Confusion Matrix for Test Data:

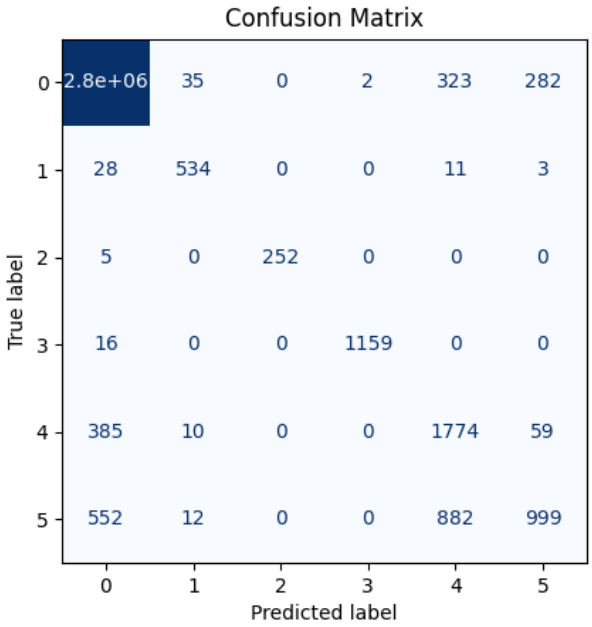


Figure 4.2: Confusion Matrix for Test Data

Multi-Layer Perceptron Model

The Multi-Layer Perceptron (MLP) model was designed with the following characteristics:

Characteristic	**Value**
Loss Function	Cross-Entropy Loss with class weights
Hidden Layers	64, 64
Epochs	3
Learning Rate	0.0001
Optimizer	AdamW
Weight Decay	0.01
Activation Function	ELU

Table 4.5: Characteristics of the Multi-Layer Perceptron Model for Amaretto Dataset

The performance metrics for the MLP model are presented below. The results include overall performance metrics, class-specific metrics, and a confusion matrix for the test data.

Dataset	Accuracy	Precision (Weighted)	Recall (Weighted)	F1 Score (Weighted)
Test	0.9985	0.9988	0.9985	0.9985

Table 4.6: Overall Performance Metrics for Test Data

Class-specific metrics for the test data are detailed below:

- Class 0 - Precision: 0.9997, Recall: 0.9993, F1 Score: 0.9995
- Class 1 - Precision: 0.6828, Recall: 0.3438, F1 Score: 0.4573
- Class 2 - Precision: 0.6970, Recall: 0.8949, F1 Score: 0.7836
- Class 3 - Precision: 0.8635, Recall: 0.9209, F1 Score: 0.8913
- Class 4 - Precision: 0.4498, Recall: 0.9408, F1 Score: 0.6086
- Class 5 - Precision: 0.6228, Recall: 0.3599, F1 Score: 0.4562

The confusion matrix for the test data is shown in Figure 4.3.

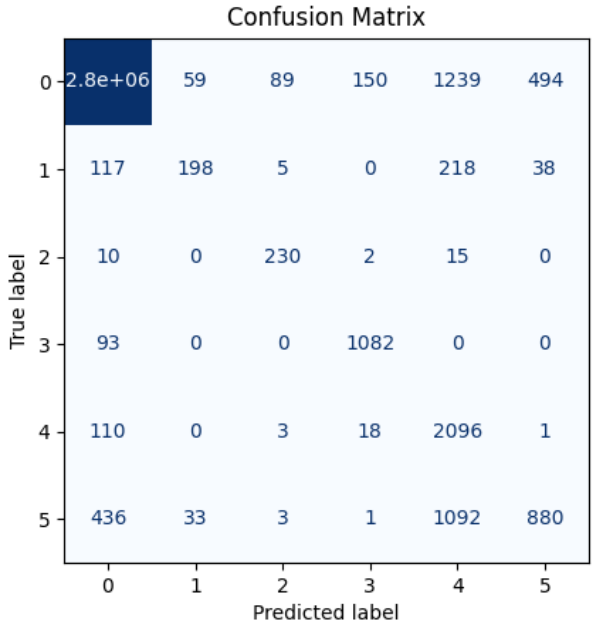


Figure 4.3: Confusion Matrix for Test Data for MLP Model

4.3 Classifier Explanation

To explain the predictions of the XGBoost model, the `shap.TreeExplainer` method from the SHAP library was employed. This method is specifically designed for tree-based models and efficiently computes SHAP values, which quantify the contribution of each feature

to the model's predictions. For the MLP model, the shap.KernelExplainer method was used. KernelExplainer is a model-agnostic method, making it suitable for explaining predictions of any machine learning model.

Multiple beeswarm plots were generated to visualize the SHAP values, showcasing the global importance of features. The beeswarm plot combines feature importance with feature effects. Each point on the summary plot is a SHAP value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the SHAP value. The color represents the value of the feature from low to high. Overlapping points are jittered in y-axis direction, so we get a sense of the distribution of the SHAP values per feature. The features are ordered according to their importance.

The global feature importance was visualized using a bar plot, where the importance of each feature was calculated as the mean absolute value of the SHAP values for that feature across all samples.

4.3.1 IBM Dataset

To analyze feature importance for the IBM dataset, SHAP was utilized to generate explanations for the XGBoost and MLP models. A representative subset of 40 random customers from testing data was selected, where each customer had 10 randomly selected transactions with a fraud ratio of 30%. The SHAP summary plots for both models, showcasing feature importance for the entire test dataset, are presented in Figure 4.6. These plots provide insights into the contribution and influence of each feature on the predictions of the respective models.

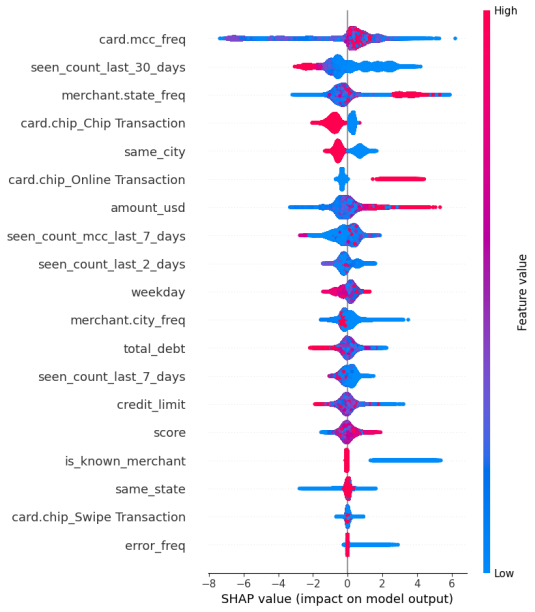


Figure 4.4: XGBoost model.

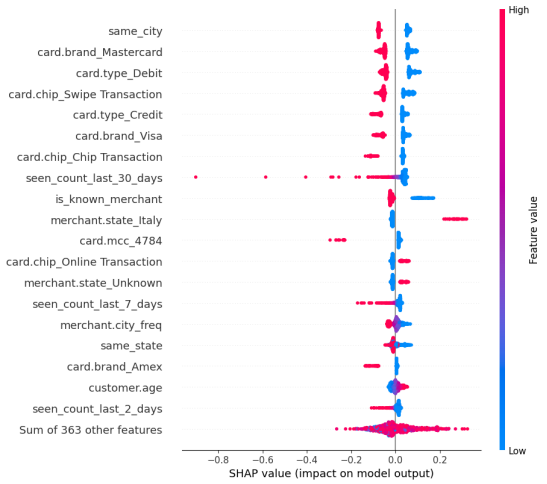


Figure 4.5: MLP model.

Figure 4.6: SHAP summary plots for XGBoost and MLP models

Additional plots were generated to provide further insights into feature importance for MLP model. These include a bar plot of the mean SHAP values and a beeswarm

plot of the maximum SHAP values, which are shown in Figure 4.9. The bar plot (Figure 4.7) orders features by the mean absolute value of their SHAP values. This ordering emphasizes the broad average impact of features across the dataset. The beeswarm plot (Figure 4.8) orders features by the maximum absolute value of their SHAP values. This ordering highlights rare but high-magnitude impacts of features.

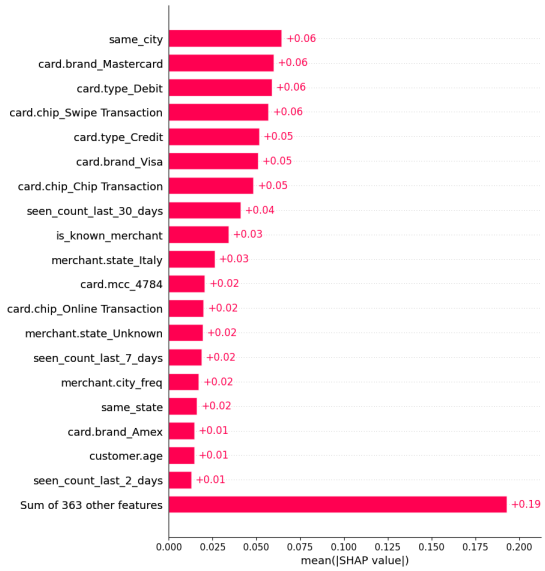


Figure 4.7: Bar plot of mean SHAP values.

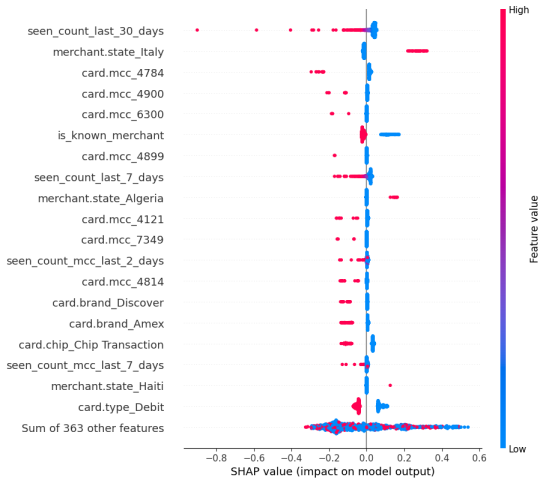


Figure 4.8: Beeswarm plot of maximum SHAP values.

Figure 4.9: Additional SHAP visualizations for the IBM dataset.

Comparison and Observations:

1. **Geographical Information:**
   Both models rely heavily on merchant information to detect anomalies related to geographic locations. XGBoost emphasizes *merchant.state* and *merchant.city*, while the MLP model focuses more on specific states like *Italy* and *Unknown*. These geographic patterns are crucial in identifying anomalous transactions in certain regions.

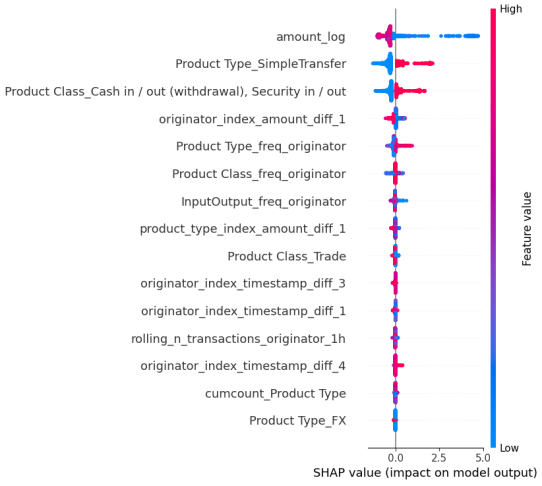
2. **Transaction History:**
   Both models emphasize transaction history with features like *seen_count_last_30_days*, *seen_count_last_7_days*, and *seen_count_last_2_days*, showing that transaction volume over time is a key indicator of fraudulent activity.

4.3.2 Amaretto Dataset

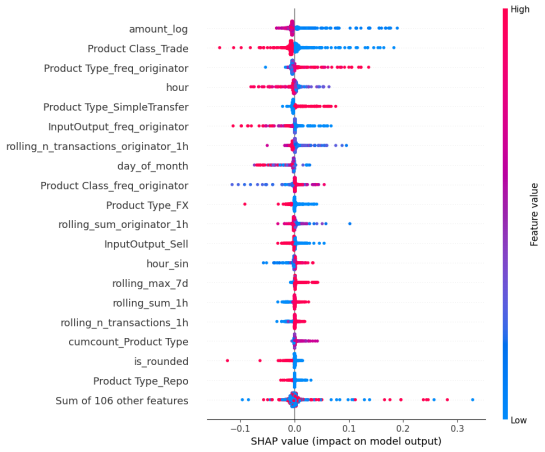
For each anomaly class in the Amaretto dataset, SHAP summary plots were created to analyze the contributions of features to the model predictions. The following figure presents a comparative view of SHAP summary plots for the XGBoost and MLP models. Each row corresponds to one anomaly class, with the left plot illustrating the XGBoost model's explanations and the right plot showing the MLP model's explanations.

To generate explanations, a subset of the testing data was selected. This subset was selected to include 30 percent of anomalies. Within this subset, the representation of each

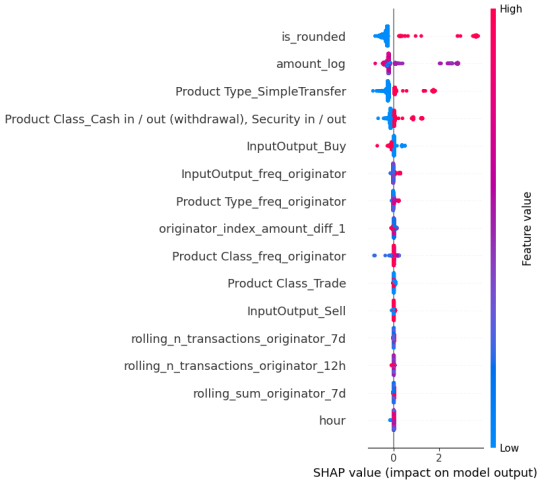
anomaly class was balanced, ensuring that each anomaly type contributed equally to the analysis.



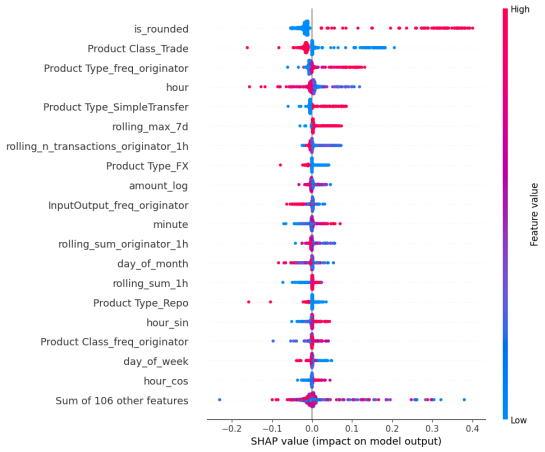
(a) Anomaly 1, XGBoost



(b) Anomaly 1, MLP

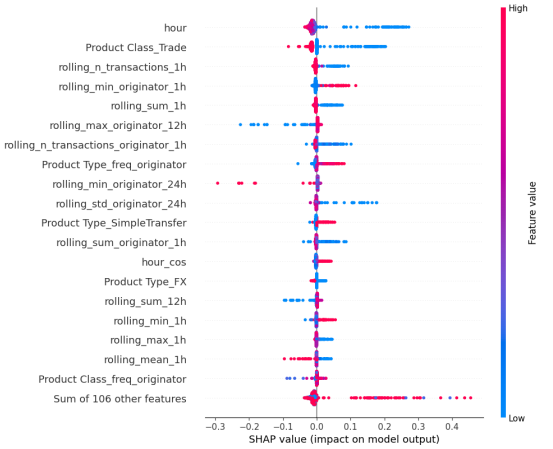


(c) Anomaly 2, XGBoost

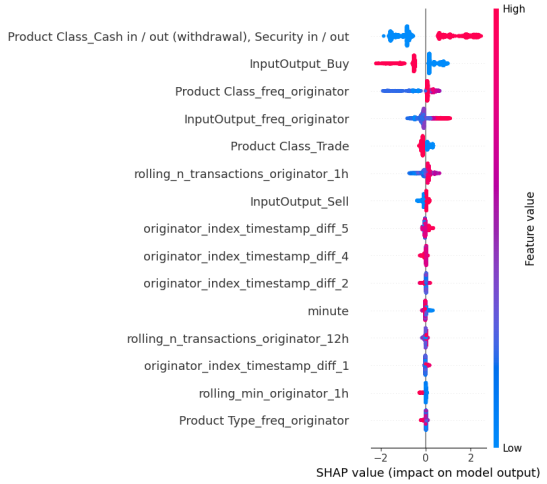


(d) Anomaly 2, MLP

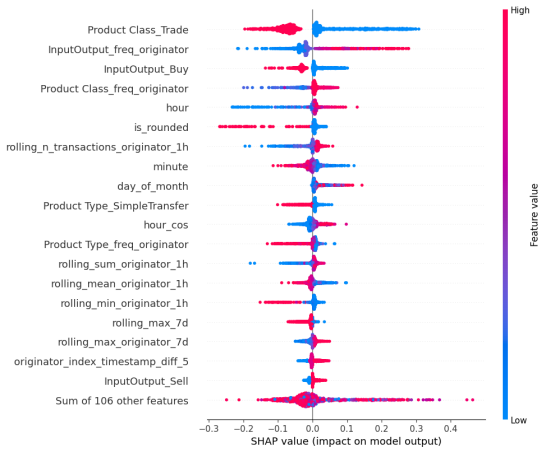
(e) Anomaly 3, XGBoost



(f) Anomaly 3, MLP

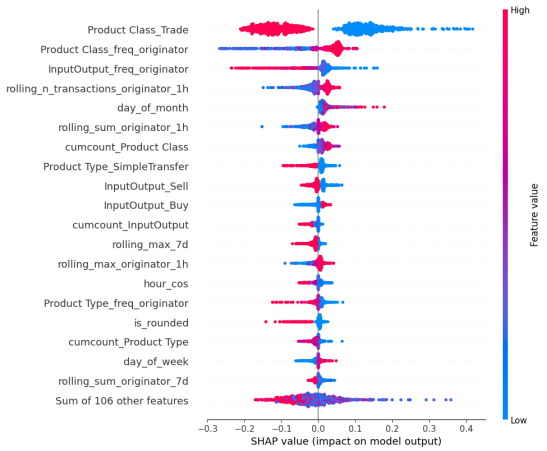


(a) Anomaly 4, XGBoost



(b) Anomaly 4, MLP

(c) Anomaly 5, XGBoost



(d) Anomaly 5, MLP

Figure 4.11: Beeswarm plots for XGBoost and MLP shap values for Amaretto Dataset.

Analysis of Model Explanations for Amaretto Dataset

- **Class 1: Small but Highly Frequent Transactions within a Short Time Frame**
  Both models identify *amount_log* as a key feature, indicating that small transaction amounts play a significant role in detecting this anomaly. XGBoost highlights transaction types like $Product\_Type\_SimpleTransfer$, while MLP emphasizes the frequency of transactions over time, such as $rolling\_n\_transactions\_originator\_1h$. These explanations are useful for detecting fraud related to rapid, small transactions, with MLP providing additional insights into transaction patterns over time.

- **Class 2: Transactions with Rounded Normalized Amounts**
  For this class, both models prioritize $is\_rounded$, directly identifying unusual amounts. XGBoost and MLP both focus on $Product\_Type\_SimpleTransfer$ and other product-related features, with MLP additionally highlighting foreign exchange-related features like $Product\_Type\_FX$. These explanations effectively pinpoint anomalies related to rounded transaction amounts, which are common in foreign exchange.

- **Class 3: Securities Bought or Sold at Unusual Times**
  Both models identify *hour* as the most important feature, helping detect transactions outside regular trading hours. XGBoost further highlights transaction types ($Product\_Type\_SimpleTransfer$) and the time of day with $hour\_cos$ and $hour\_sin$. MLP also emphasizes the volume of transactions in specific time windows with features like $rolling\_n\_transactions\_1h$. These explanations are valuable for detecting trades occurring outside typical market hours.

- **Class 4: Large Asset Withdrawal**
  XGBoost focuses on features like $Product\_Class\_Cash\_in/out\_()$, $\_Security\_in/out$ and $InputOutput\_Buy$, suggesting that large withdrawals are a key indicator. MLP shares similar features, with additional emphasis on transaction frequency, such as $rolling\_n\_transactions\_originator\_1h$. These explanations are useful for detecting large, sudden withdrawals, which are often indicative of fraudulent activity.

- **Class 5: Unusually Large Amounts of Collateral Transferred**
  For this class, both models identify $Product\_Class\_Cash\_in/out\_()$, $\_Security\_in/out$ and $Product\_Class\_freq\_originator$ as important features. XGBoost highlights timing-related features like $rolling\_max\_originator\_7d$, while MLP focuses more on transaction frequency with features like $rolling\_n\_transactions\_originator\_1h$. These explanations are helpful for detecting unusual collateral movements, which may signal potential fraud.

Chapter 5

Conclusion

Interpretability is a crucial aspect of machine learning, particularly in sensitive areas like fraud detection, where regulatory compliance and trust are essential. Although complex models such as XGBoost and Multilayer Perceptrons (MLPs) are widely used for detecting fraudulent transactions, their "black-box" nature limits understanding of how predictions are made. This lack of transparency can undermine confidence in model outputs. To address this challenge, SHAP was applied to explain the predictions of these models.

The results obtained from the application of SHAP to the IBM and Amaretto datasets showed that both XGBoost and MLP models generated meaningful explanations. In the Amaretto dataset, SHAP provided clear insights into the characteristics of different anomaly classes, helping to identify key features for detecting specific fraudulent behaviors.

While SHAP explanations significantly enhance interpretability, further research could focus on inherently interpretable models, such as prototype neural networks. These models are designed to provide more transparent decision-making from the outset, reducing the need for post-hoc explanations and fostering a deeper understanding of how fraud detection models make predictions.

Bibliography

[1] Erik R. Altman. Synthesizing credit card transactions, 2019.

[2] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: Deep learning for interpretable image recognition, 2019.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 11 of *KDD '16*, page 785–794. ACM, August 2016.

[4] Danilo Labanca, Luca Primerano, Marcus Markland-Montgomery, Mario Polino, Michele Carminati, and Stefano Zanero. Amaretto: An active learning framework for money laundering detection. *IEEE Access*, pages 1–1, 2022.

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[6] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[7] Chaithanya Vamshi Sai, Debashish Das, Nouh Elmitwally, Ogerta Elezaj, and Md Baharul Islam. Explainable ai-driven financial transaction fraud detection using machine learning and deep neural networks, 2023. 39 Pages, Posted: 18 May 2023, Birmingham City University, Available at: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4439980`.