Mobile Phone Transactions Fraud Detection

Chujun Chen
Data Science Institute, Brown University
Github: https://github.com/Christina-Chen01/DATA1030-FinalProject-Fraud-Detection

1. Introduction

In the era of digital transactions, the rise in mobile phone-based financial activities has rendered the monitoring of fraudulent transactions both increasingly vital and challenging. The absence of an effective fraud detection system can lead to severe consequences, including significant financial losses for both individuals and businesses, erosion of customer trust, and escalated operational costs for financial institutions. Hence, the development of effective fraud detection systems is imperative to protect assets and maintain financial market integrity.

Previous research has extensively explored various fraud detection models. Studies by Varmedja et al. (2019) and Khatri et al., (2020) have particularly focused on the efficacy of models like k-nearest neighbor (KNN), logistic regression, decision tree, and random forest. Both studies identified the decision tree and random forest models as particularly effective, yielding the best precision score in fraud detection. Nevertheless, these models were applied post-oversampling of the minority class in the dataset prior to modelings.

Without implementing any oversampling techniques, this project delves into the realm of fraud detection within mobile phone transactions by utilizing a highly imbalanced dataset. It employs a range of supervised machine learning techniques, such as Logistic Regression, K Nearest Neighbor, Random Forest, and XGBoost, implemented within a Jupyter Notebook environment.

2. Exploratory Data Analysis

The dataset utilized in this project was obtained from Kaggle and is a synthetic dataset created by the PaySim simulator. This simulator mimics the usual patterns of mobile money transactions, drawing inspiration from genuine transaction data. The original dataset comprises 2,770,409 records across 10 columns, offering a substantial basis for analysis and model comparison.

Given the extensive volume of the dataset, we strategically chose to focus our initial training efforts on the data from the seven days.

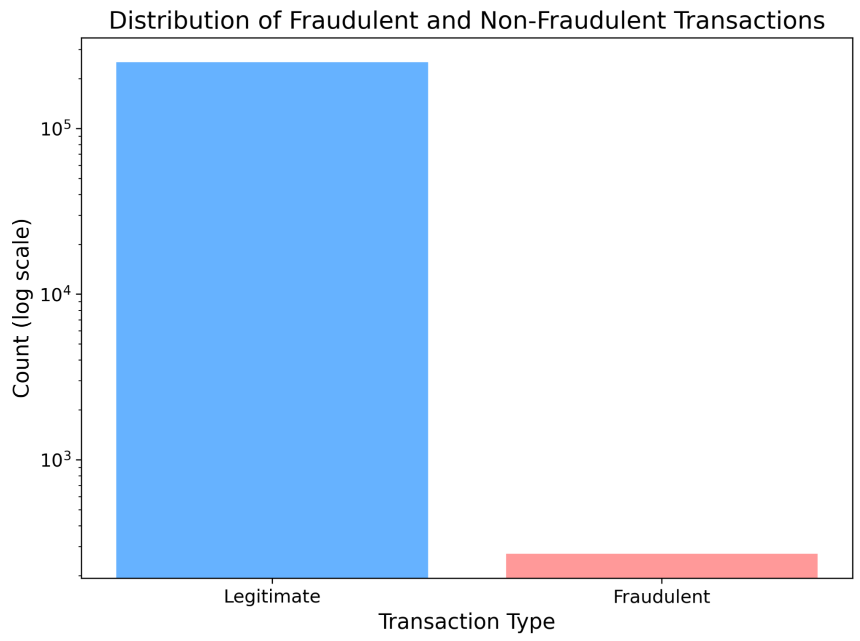


Figure 1. highlights the pronounced imbalance in the 'isFraud' target variable, where a mere 0.018% of transactions are labeled as fraudulent.

Figure 2 revealed that fraudulent activities were exclusively associated with two types of transactions: 'CASH_OUT' and 'TRANSFER'. In light of this, we filtered out other transaction types to address class imbalance, thus sharpening our focus on the most relevant data for fraud detection.

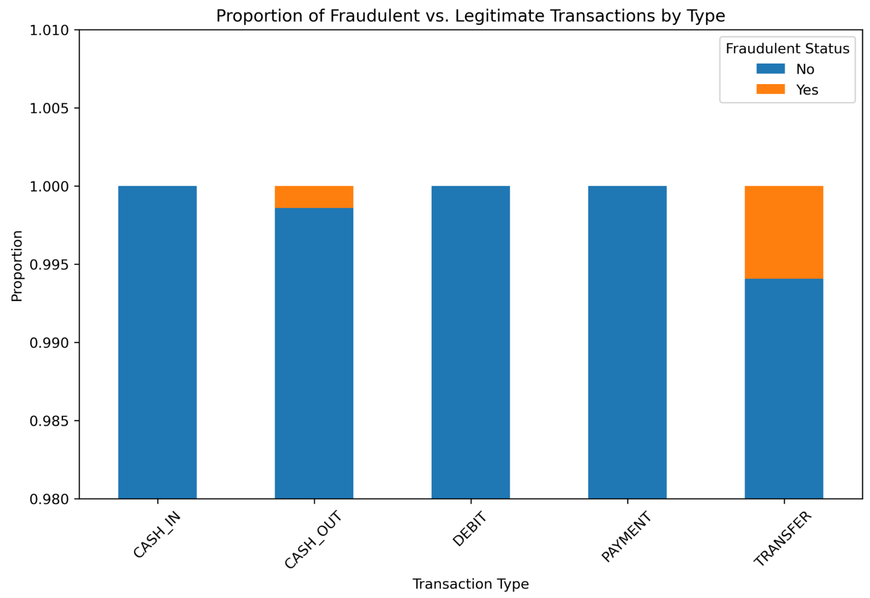


Figure 2. illustrates the proportion of fraudulent vs Legiticame transactions by type.

Transaction type and amount both impact fraud. Figure 3 shows higher median transaction amounts for fraud compared to legitimate transactions. Recognizing this distribution is vital for fraud detection and can enhance machine learning model accuracy by using larger sums as a potential feature.

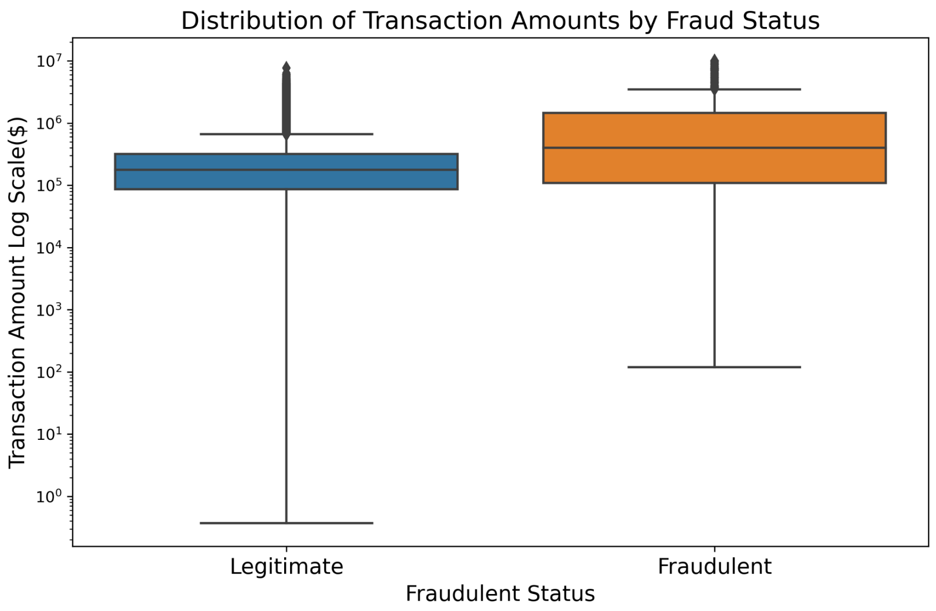
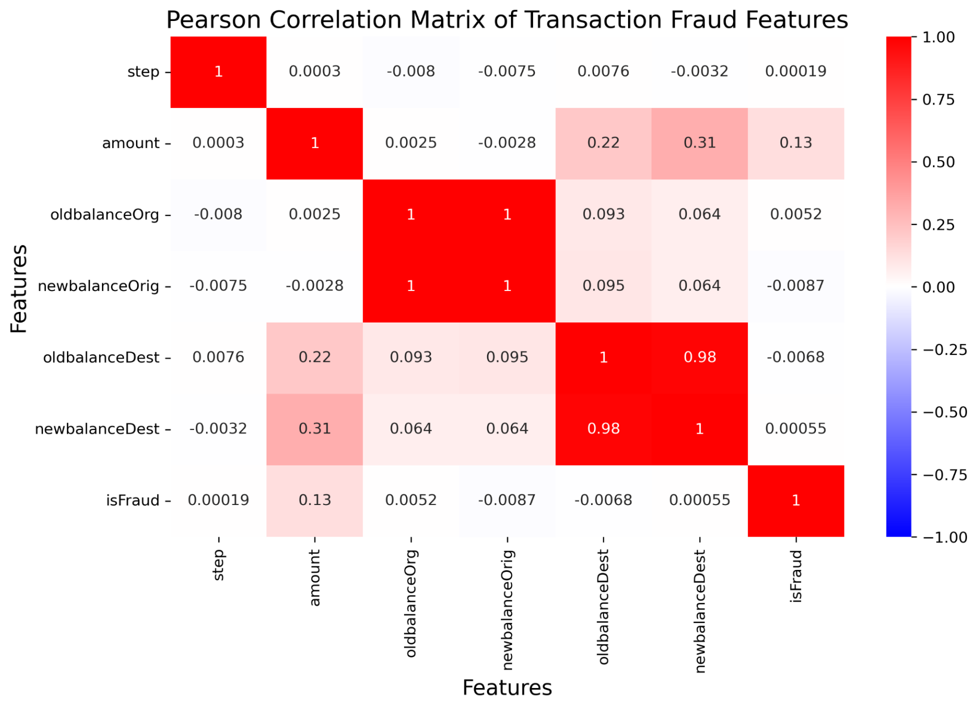


Figure 3. highlights a higher median of transaction amount for fraudulent transactions than its legitimate counterpart.



	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1	0.0003	-0.008	-0.0075	0.0076	-0.0032	0.00019
amount	0.0003	1	0.0025	-0.0028	0.22	0.31	0.13
oldbalanceOrg	-0.008	0.0025	1	1	0.093	0.064	0.0052
newbalanceOrig	-0.0075	-0.0028	1	1	0.095	0.064	-0.0087
oldbalanceDest	0.0076	0.22	0.093	0.095	1	0.98	-0.0068
newbalanceDest	-0.0032	0.31	0.064	0.064	0.98	1	0.00055
isFraud	0.00019	0.13	0.0052	-0.0087	-0.0068	0.00055	1

Figure 4. Pearson Correlation Coefficient matrix for features, highlighting potential multicollinearity that may necessitate features removal.

Regarding the assessment of linearity, Figure 4 reveals a strong correlation between the new and old balances for both the sender and the recipient in the feature matrix. This suggests the need to eliminate 'oldbalanceOrg' and 'oldbalanceOrg' due to their collinearity before data training and splitting.

After applying these implications, the dataset was truncated to 106,197 observations and 6 features and the target variable 'isFraud'.

3. Method

We assessed model performance using Average Precision Score, a metric ideal for imbalanced datasets focusing on the accurate prediction of the minority class (Khatri et al., 2020). This score simplifies model comparison by encapsulating performance in a single figure. We also used the Area Under the Precision-Recall Curve (AUPRC) to balance precision and recall, providing a detailed view of our models' performance.

3.1 Data Split

Given the time series nature of our dataset, we employed a *TimeSeriesSplit* strategy (*n_splits = 4*) to achieve 80 – 20 splits of our train_val to test datasets, while preserving the integrity and chronological order of transactions. This final 20% served as a holdout test set for evaluating unseen data.

3.2 Preprocessor

Data preprocessing involved handling categorical variables through *OneHotEncoder* and scaling numerical features to normalize their range through *StandardScaler*. The number of features increased by 1 since there are two different transaction types.

3.3 Hyperparameter Tuning and Cross Validation

Hyperparameter tuning in our models was targeted to optimize performance based on average precision scores. We implemented Lasso (l1) and Ridge (l2) regressions to control overfitting.

To counter the imbalanced nature of fraud detection, we adjusted the class weights in Logistic Regression and Random Forest, and prioritized closer neighbors in KNN. XGBoost adopted 'scale_pos_weight', based on the ratio of negative to positive instances, to give greater weight to the minority class.

ML Mode	**Hyperparameter**	**Values**
Logistic Regression	C penalty solver class_weight	0.001, 0.01, 0.1, 1, 10, 100, 1000 l1, l2 saga balanced

Random Forest Classifier	n_estimators max_depth max_features class_weight	25, 50, 100, 200 10, 20, 30, None None, sqrt balanced, balanced_subsample
KNeighbors Classifier	n_neighbors weights	5, 10, 15, 20 distance
XGBoost Classifier	n_estimator scale_pos_weight learning_rate reg_alpha reg_lambda colsample_bytree	500, 750 weight = (y == 0).sum() / (1.0 * (y == 1).sum()) 0.01, 0.03 0.01, 1, 100 0.01, 1, 100 0.5, 0.9

Table 1. Models trained and hyperparameters tuned.

3.4 GridSearchCV and ML Pipeline

We leveraged sklearn's GridSearchCV for systematic hyperparameter tuning described in Section 3.3, coupled with *TimeSeriesSplit* (*n_splits* = *4*) for maintaining temporal data integrity. After partitioning 80% of the dataset for model tuning, we tested the best-performing models on the remaining 20% holdout set to gauge their accuracy on unseen data.

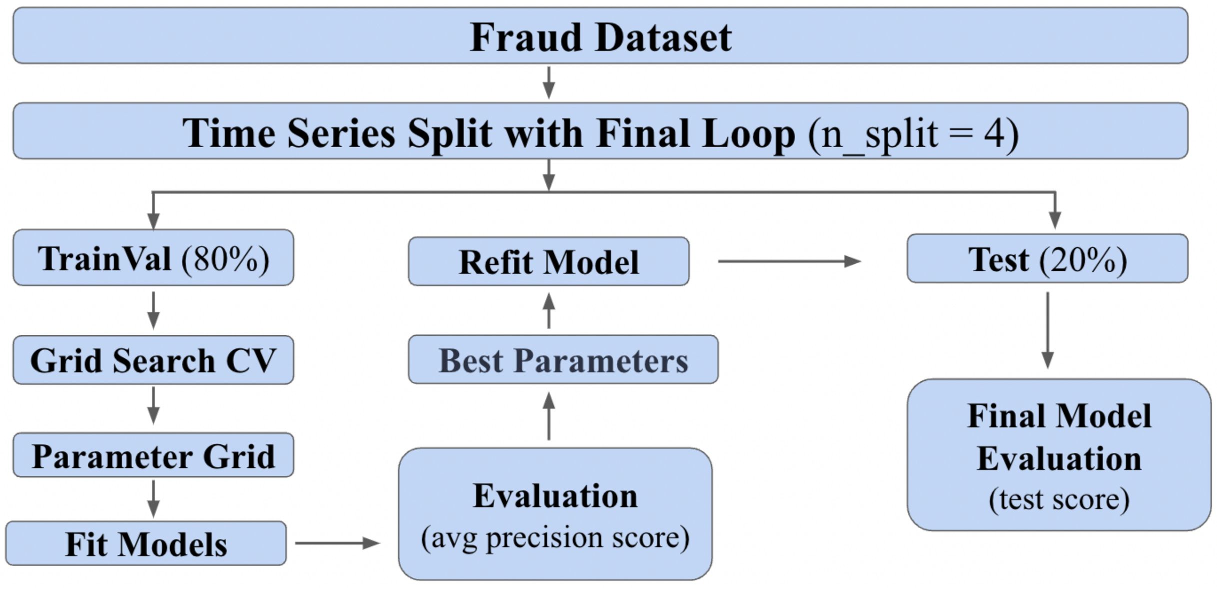

Fraud Dataset
Time Series Split with Final Loop (n_split = 4)
TrainVal (80%)
Grid Search CV
Parameter Grid
Fit Models
Evaluation (avg precision score)
Best Parameters
Refit Model
Test (20%)
Final Model Evaluation (test score)

Figure 6. Flow chart of the GridSearchCV pipeline.

4. Results

The baseline average precision score is calculated by the proportion of positive examples in the holdout test set, which is 0.0154. The best parameters for each model and runtime are shown in the following table.

ML Mode	Best Hyperparameter	Values	Average Precision Score	Run time (seconds)
Logistic Regression	C penalty solver class_weight	1.0 l1 saga balanced	0.088	76.477
Random Forest Classifier	n_estimators max_depth max_features class_weight	200 10 sqrt balanced	0.713	557.083
KNeighbors Classifier	n_neighbors weights	30 distance	0.063	1.492
XGBoost Classifier	n_estimator scale_pos_weight learning_rate reg_alpha reg_lambda colsample_bytree	750 54.766 0.03 1.0 1.0 0.5	0.736	1054.58

Table 2. Best parameter configuration obtained, test score, and run time for each model

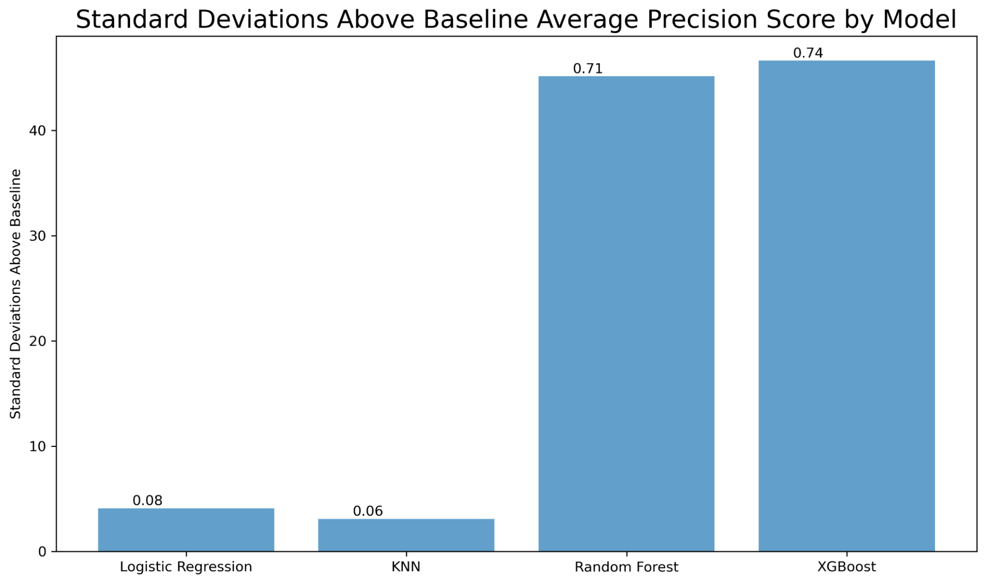


Figure 7. displays the standard deviations above the baseline performance and the actual test score labeled above each bar for four predictive models.

Analyzing Table 2 and Figure 7, it's evident that XGBoost achieves the highest average precision score, closely followed by Random Forest with a marginally lower score. In terms of baseline precision, both XGBoost and Random Forest significantly outperform the baseline, exceeding it by more than 40 standard deviations.

In contrast, Logistic Regression and KNN show only marginal improvements over the baseline, with their scores being merely 0.08 and 0.06 standard deviations higher, respectively. This suggests that the advanced ensemble techniques employed by XGBoost and Random Forest may be more adept at capturing the complexities of the dataset.

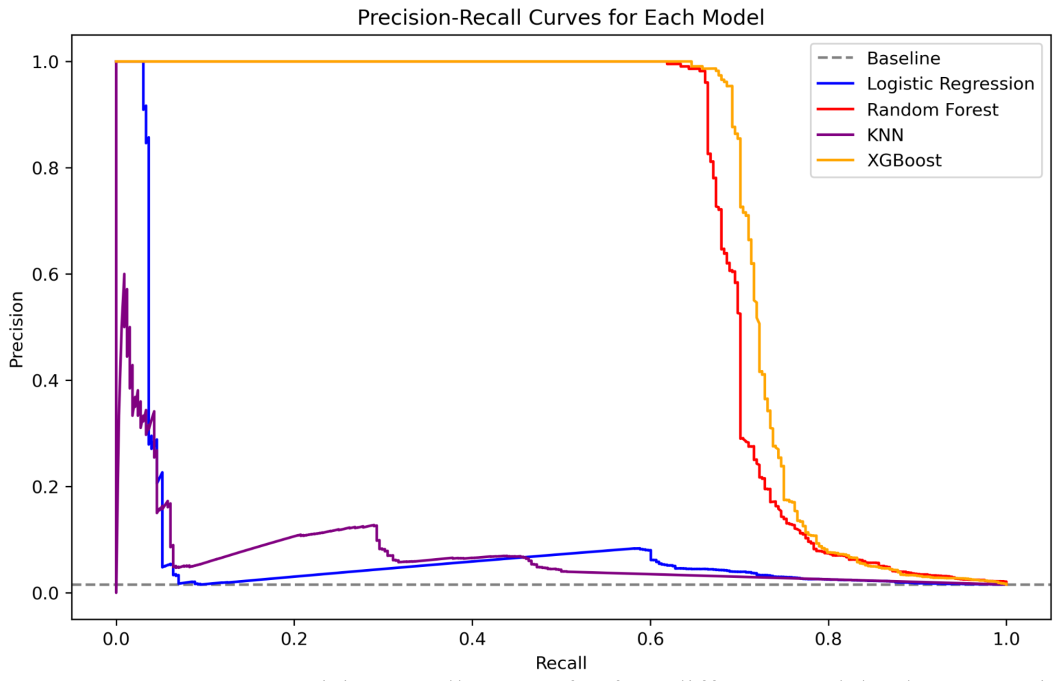


Figure 8. presents precision-recall curves for four different models, demonstrating their trade-offs between precision and recall.

The precision-recall curve reinforces these findings, particularly highlighting XGBoost's superior performance with the largest area under the curve, indicating a strong balance of high precision and recall.

4.1 Global Feature Importance

We computed three distinct global feature importance metrics for the best-performing XGBoost model: permutation feature importance, XGBoost Gain, and SHAP values. These metrics provided comprehensive insights into feature relevance, with each method offering a unique perspective on how different features influence the model's predictions.

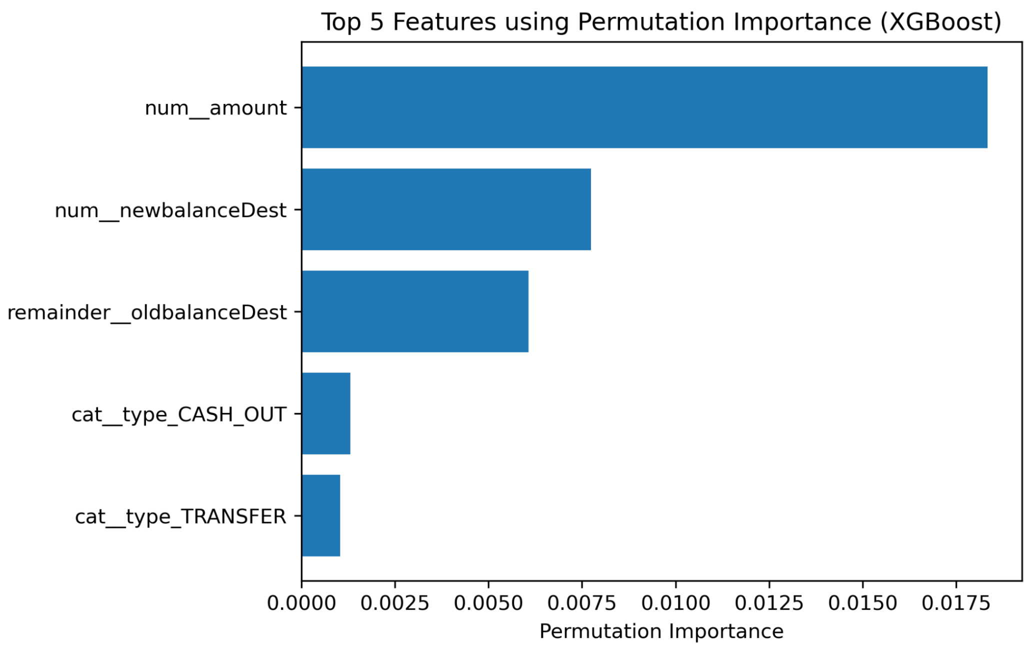


Figure 9. Top 5 permutation importance using XGBoost test sets.

'num_amount', which represents the transaction amount, is the most influential feature for the XGBoost model, suggesting transaction amount is critical in predicting fraud. A higher value for this feature likely indicates a greater impact on the model's prediction, as fraudulent transactions can often involve larger sums of money.

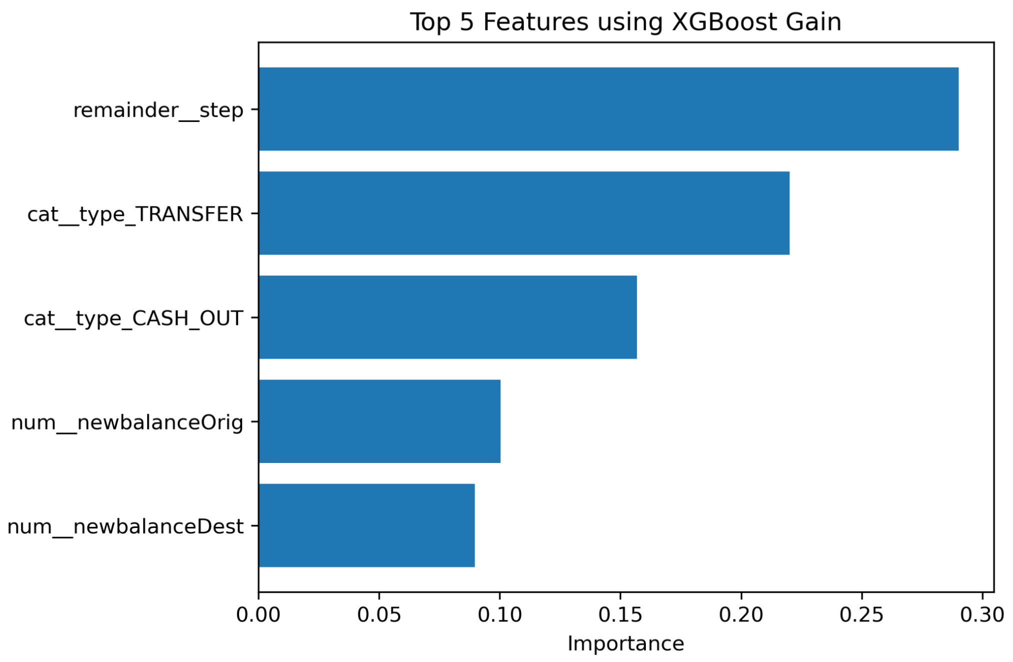


Figure 10. Top 5 most important features as XGBoost relative importance measure - gain

XGBoost Gain identified 'remainder_step' as the top feature, indicating the timing of transactions plays a key role in fraud prediction as it could indicate irregularities in transaction patterns. Fraudulent activities may occur at atypical times or demonstrate unusual frequencies, which can be captured by this temporal feature. Its high gain implies that factoring in the time element improves the model's ability to identify potential fraud.

Moreover, the number of new balance destinations (num_newbalanceDest), representing the updated balance of the recipient's account, has positively contributed to fraud detection; this could indicate that an increase in the recipient's balance post-transaction is a strong predictor of fraudulent activity, potentially due to the pattern that fraudulent transactions often result in significant balance changes for the recipients involved.

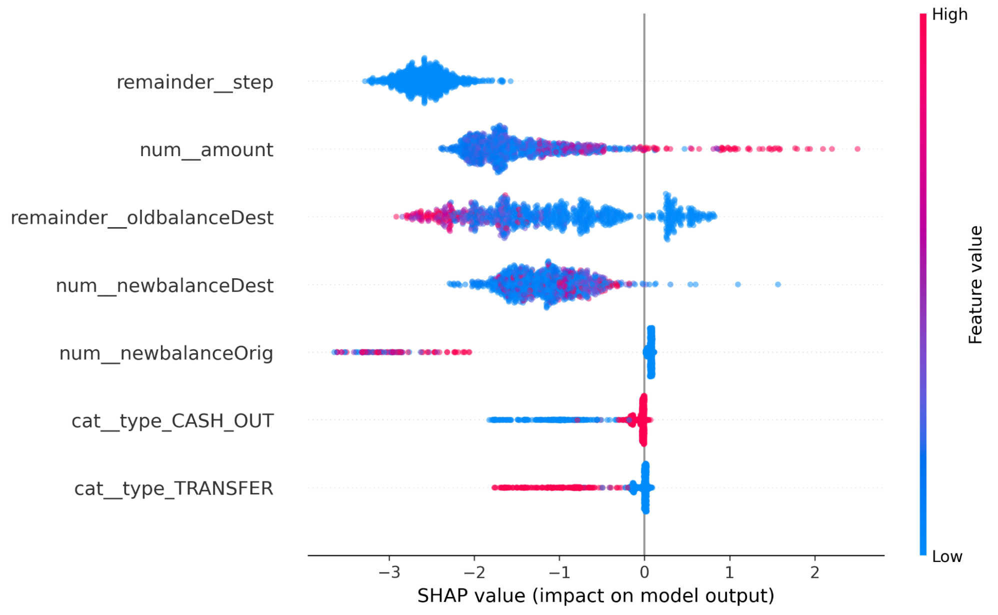


Figure 11. Top 5 most important features per SHAP values.

In Figure 11, the predominance of blue in 'remainder_step' suggests its low values often indicate non-fraudulent transactions. Conversely, 'num_amount' displays a broader spread of SHAP values, implying that higher transaction amounts can both increase and decrease fraud probability, varying with their interaction with other features.

The transaction types 'cat_type_CASH_OUT' and 'cat_type_TRANSFER' mainly show red dots to the right of the zero line in SHAP values, signifying their positive contribution to fraud predictions. These transaction types are more commonly associated with fraudulent activities, hence their strong predictive power for fraud.

4.2 Local Feature Importance

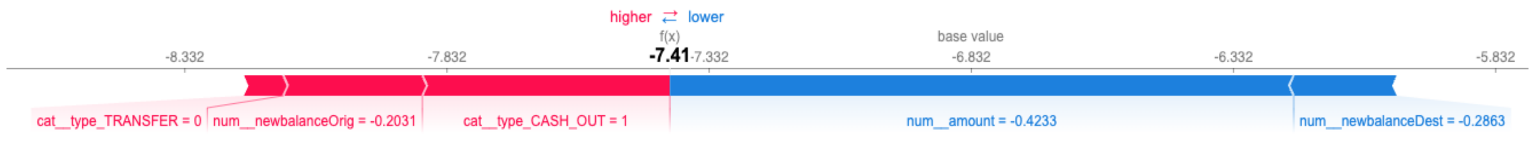


Figure 12. Local feature importance SHAP values.

While the ‘Cash Out’ transaction type shifts the prediction towards fraud, transaction amount (num_amount) negatively impacts the mode’s prediction, diverging from the baseline towards a non-fraudulent classification. This specific transaction corroborates the insight gained from the Global Feature Importance, highlighting the transaction amount's inconsistent effect on the likelihood of fraud.

Moreover, ‘num_newbalanceDest’ has a negative SHAP value, indicating that for this specific transaction, a lower new balance in the destination account (num_newbalanceDest) is pushing the model’s prediction away from fraud. It seems transactions that result in a smaller balance in the recipient’s account are less likely to be fraudulent, possibly because fraudulent transactions typically aim to transfer out larger amounts to deplete the account.

5. Outlook

After an in-depth examination of the fraud detection project, several enhancements and considerations for future work emerge:

- Engage in feature engineering and selection by incorporating diverse data sources and creating new features such as ‘is_weekend’, ‘holder_age’, ‘transaction_location_frequency’ to better capture fraud patterns.
- Implementing the Naive Bayes algorithm could offer a different perspective in modeling, particularly in terms of handling probabilistic outcomes and feature independence (Khatri et al., 2020).
- Address the potential weak spot of static model validation by adopting a rolling or expanding window approach for time series analysis, ensuring the model remains effective against evolving fraud patterns.

6. Reference

Kaggle (Edgar Lopez-Rojas). (2016). Synthetic Financial Datasets For Fraud Detection. Retrieved from https://www.kaggle.com/datasets/ealaxi/paysim1

S. Khatri, A. Arora and A. P. Agrawal, 2020. Supervised Machine Learning Algorithms for Credit Card Fraud Detection: A Comparison. *10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2020, pp. 680-683, doi: 10.1109/Confluence47617.2020.9057851

D. Varmedja, M. Karanovic, S. Sladojevic, M. Arsenovic and A. Anderla, 2019. Credit Card Fraud Detection - Machine Learning methods. *18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, East Sarajevo, Bosnia and Herzegovina, 2019, pp. 1-5, doi: 10.1109/INFOTEH.2019.8717766.