Fraud Detection in Synthetic Mobile Money Transactions with Machine Learning Classifiers

Bernard Wittmaack

Problem Statement and Goals

While mobile payment services (MPS) are quickly gaining global popularity, especially in countries without much banking penetration, they are susceptible to fraud that costs both the MPS company and its users. Fraud must be detected both quickly and accurately to ensure that MPS are perceived as reliable and low-risk ways to do business. Using a synthetic dataset modeled after actual mobile money transactions, we address the question of whether fraud can be consistently predicted from the available data. Three classifiers are used: logistic regression, random forest, and support vector machines. We find that support vector machines have the highest average precision for fraud prediction and conclude with a summary of each model's performance, advantages, and drawbacks.

Introduction

Mobile payment services such as PayPal's Venmo, offer unprecedented ease of transferring funds, splitting and paying bills, and even tipping musicians and other performers. Developing nations have seen MPS vitalized commerce and boost GDP as these services are especially attractive to populations where mobile phones are significantly more prevalent than individual bank accounts. [1] McKinsey & Company projects a combined $3.7 trillion increase in GDP by 2025. [2] Unfortunately, MPS transactions are also ripe for exploitation and fraud. The MPS giant Venmo reported an operating loss of close to $40 million in early 2018 largely due to fraud. [3] As MPS become widely accepted and continue to drive economic growth, transfers need to be both secure and fast. Any fraudulent transaction should ideally be recognized in the instant it is confirmed so that payment is safely held. At the same time, if the algorithm to detect fraud is too zealous, many legitimate transactions will be put on hold while undergoing further review and stifle enterprise.

Dataset

Finding the optimal balance between over- and under-predicting fraud requires carefully training (and re-training) machine learning algorithms on realistic data. However, due to the sensitive nature of financial transactions data coupled with personal information has resulted in the dearth of usable datasets

accessible to fraud-risk researchers outside of the corporate setting. One way to circumvent this obstacle is with a synthetic dataset. Researchers at the Blekinge Institute of Technology and The Norwegian University of Science and Technology used a sample of real logs of an African MPS company to simulate a high-fidelity dataset that serves as an excellent substitute for actual transactions data. [4]

A version of this synthetic dataset is hosted on Kaggle, [5]. Consisting of 6.36M records, the data spans a period of 31 days. The table below provides the names and description of the dataset's 11 columns.

Table 1: List and description of all columns in the synthetic dataset.

Column	Description	Values and Notes
Step	One step represents an hour in simulated time. No further context is provided.	Range of 1-743, representing a 31-day period.
Type	Type of mobile money transaction.	Takes on 5 values: • CASH-IN is the process of increasing the balance of account by paying in cash to a merchant. • CASH-OUT is the opposite process of CASH-IN, it means to withdraw cash from a merchant which decreases the balance of the account. • DEBIT is similar process than CASH-OUT and involves sending the money from the mobile money service to a bank account. • PAYMENT is the process of paying for goods or services to merchants which decreases the balance of the account and increases the balance of the receiver. • TRANSFER is the process of sending money to another user of the service through the mobile money platform!
amount	Amount of transaction	Unspecified currency with range from 0 to ~92M
nameOrig	ID of the originator	IDs that start with "C" are customers and those beginning with "M" are merchants.
oldbalanceOrig	Balance of the originator's account before transaction	Values range from 0 to ~60M (unspecified currency)
newbalanceOrig	Balance of the transaction originator's account after the transaction	Values range from 0 to ~50M (unspecified currency)

nameDest	ID of the recipient	IDs that start with "C" are customers and those beginning with "M" are merchants.
oldbalanceDest	Balance of the destination account before transaction	Values range from 0 to ~356M (unspecified currency)
newbalanceDest	Balance of the destination account after the transaction	Values range from 0 to ~356M (unspecified currency)
isFraud	Is the transaction fraudulent	Binary value of 0 or 1
isFlaggedFraud	Is transaction flagged as fraudulent	Simple rule that any transaction greater than 200K is flagged as potential fraud. Binary value of 0 or 1

Exploratory Data Analysis

No anomalies or significant outliers were detected in any of the columns of the dataset, owing to its synthetic origin. Some important notes and observations:

- The dataset is highly imbalanced. Only 1/775 transactions are fraudulent.
- Over 99.9% of the sender (*nameOrig*) and recipient (*nameDest*) IDs were linked to only one transaction, precluding any detailed social network analysis.
- Only the transaction types of CASH_OUT and TRANSFER were linked to fraudulent transfers. The remaining three transaction types were only observed for legitimate money transfers.
- Fraud never involves merchants (IDs starting with an "M"). Merchants are not involved with either CASH_OUT or TRANSFER payment types.
- The column *isFlaggedFraud* woefully underpredicts fraud. Fraud is flagged approximately in 1/400,000 transactions. This is 500 times less than the actual fraud rate, making it clear that such a simple algorithm is insufficient to detect a great majority of fraud.

Feature Selection and Pre-processing

Features Associated with Fraud

Both the amount and all balance fields are highly right-skewed (the mean is an order of magnitude greater than the median). Therefore, a logarithmic transformation is applied to these data to produce an approximately normalized distribution. Examining the diagonal of the pair plot of the numeric variables (Figure 1) reveals that all their distributions appear to differ significantly between fraudulent and legitimate transactions. To better compare these two types of distributions in this imbalanced dataset, the pair plot shows a random sample of legitimate transactions equal in number to the fraudulent transactions. The distributions of the log-transformed transaction amounts, while having considerable overlap between the fraudulent and legitimate transactions, indicate that transaction amounts on the

order of a couple hundred thousand have proportionally more fraud than at transactions at lesser amounts. Examining the *oldbalanceOrg* we notice originators with high starting balances over 100,000 are disproportionately at risk for fraud. Furthermore, the distribution for *newblanceOrig* has a pronounced peak at zero (note that in pair plot that $\log(-4) \equiv 0$ due to an added 10^{-4} offset for zero values). Another striking feature in the pair plot is the interaction between *amount* and *oldblanceOrig* for fraudulent transactions. There is a strong linear relationship between these two fields for fraud but no discernable relationship for legitimate transactions. Putting these pieces of evidence together immediately suggests that fraudsters will attempt to empty their victim's entire account. Hence, one piece of essential information needed for detecting fraud is whether the originators account balance is zero after the transaction. Analysis of all fraudulent transactions confirmed that over 97% of fraud resulted in a zero balance for the originator. A binary feature, *drain_Orig_account* was created to store this information for each record.

Notice that the destination balances are omitted from the pair plot. In the synthetic dataset, the destination balances are not updated after every transaction but rather only at the end of a step (i.e., an hour of simulated time), during which multiple transactions may occur. This lack of real-time balance data artificially weights the distribution since the same destination ID will have the same destination balances over a step no matter how many transactions occurred during that time. As previously mentioned, the majority of fraudulent transactions can directly be tied to whether the originator's account is emptied, and therefore, for this analysis, the destination balance amounts are neglected.

With the engineered feature of whether the originator's account is emptied or not in the transaction, the actual account balances become much less critical for detecting fraud. Of course, the exact amount of the transaction can still be important, but to simplify the model, we chose to drop all account balances from the final feature set.

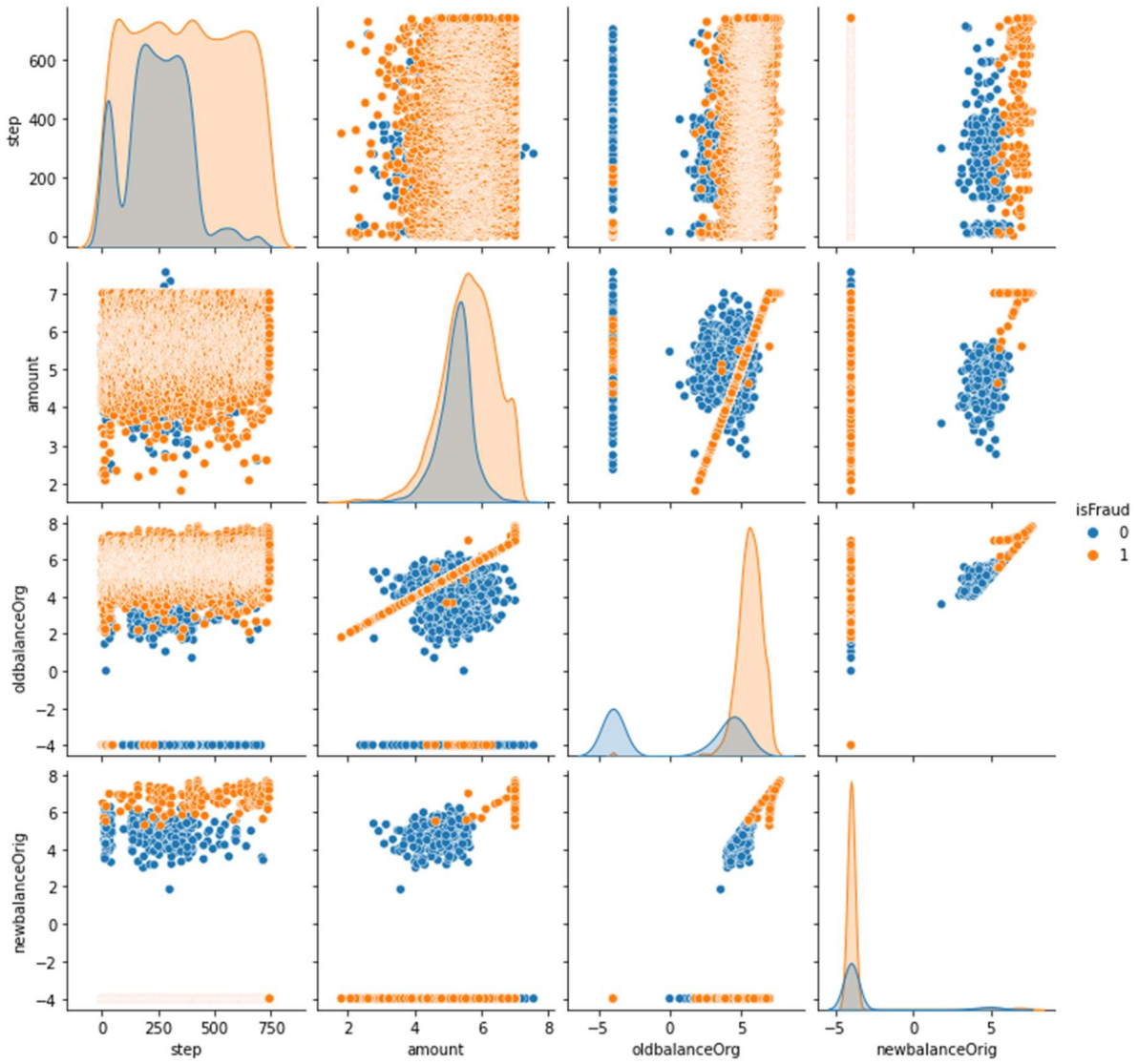

Figure 1: Pair plot of the distributions of numerical variables colored by whether it is for fraudulent (orange) or legitimate (blue) transactions.

The Role of Transaction Time in Fraud Detection

So far, we have discussed the association of monetary amounts and balances with fraud. However, we have not directly addressed the *step* column. From Figure 1, it is evident that fraud occurs almost uniformly throughout the simulation. In contrast, legitimate transactions are highly localized towards the earlier to middle steps. In reality, the time and date of a transaction often factor into assessing fraud risk. Whether a transaction occurred during regular hours, on a weekday or weekend, or during a holiday are all potentially important factors in predicting whether it is fraudulent or not. Unfortunately,

this synthetic dataset provides no context for the *step* field other than that each unit corresponding to one hour. No start time or date is given. Furthermore, we do not know the country or population the synthetic dataset was modeled after. The pair plot does not reveal any apparent daily or weekly trends for the legitimate transactions.

The *step* distribution does appear to differ between fraudulent and legitimate transactions markedly, but on the other hand, it is difficult to generalize any machine learning model with *step* included as it is given. Without any kind of feature engineering, we rely on some form of the absolute value of step (ranging from 1 to 743). Clearly, if we were to extend a model fitted on *step,* absent any feature engineering, we would expect that this feature's inclusion would only hurt the model's predictive power on unseen data. However, for the purposes of this project, *step* is included and unaltered as a final feature in the model with the aforementioned caveat on generalizability.

Final List of Features

The final list of model features (prior to any preprocessing) is as follows:

- Step (no scaling)
- Transaction Type (one-hot encoded)
- Is Originator's Account Emptied (one-hot encoded)
- Transaction Amount (power scaled)

Under-Sampling to Deal with the Imbalanced Data

Since only two transaction types are associated with fraud (CASH_OUT and TRANSFER), all other records were removed for subsequent EDA and analysis. Even with only these two transaction types, the data remains imbalanced with fraud accounting for only 0.3% of the records, meaning that a hypothetical model could predict that all transactions are legitimate and still be right 99.7% of the time. Since we are particularly interested in correctly flagging fraud and because most machine learning algorithms do not perform well with highly imbalanced data, we need to address this issue somehow. The abundance of records (2.8 million) makes under-sampling legitimate transactions an attractive option. With this approach, a random sample of legitimate transactions is chosen without replacement equal to the number of fraudulent records (8,213), yielding a final, perfectly balanced dataset with 16,246 observations.

A key assumption with the under-sampling method is that the randomly selected subset is statistically indistinguishable from the population it was drawn from. To test this, the distributions of the features among the sample and large population of legitimate transactions was compared. In the cases of

the categorical features, the ratios of samples in each group were effectively identical in both groups. For the numerical features, a Kolmogorov-Smirnov test indicated the equality of the respective distributions.

Preprocessing: Categorial Encoding and Numerical Scaling

The categorial features (transaction type and if the originator's account was emptied) are nominal with low cardinality. Hence, they are one-hot encoded (as either 0 or 1) so that can easily be ingested by most common machine learning packages (e.g., *Scikit-learn [6]*).

As previously discussed, the amount is highly right-skewed. The power transform scaler provided in *Scikit- learn's* preprocessing module was used to scale this feature. No transformation was done on *step*.

Training and Testing Data

A 70%-30% train-test splitting of the final data is selected. This ratio of training and testing data provides a good balance between having sufficient records for the model to train on and a large enough subset to reliably infer the model's ability to generalize to unseen data. It is also a widely used ratio in the machine learning community.

Modeling

Candidate Models

Table 2 describes the three machine learning classifiers selected to classify the mobile money transactions of the synthetic dataset as either fraudulent or legitimate, along with some of their respective advantages and disadvantages. The set of models, logistic regression, random forest, and support vector machines was chosen because each member is a well-known classifier with unique advantages. The Scikit-learn [6] package is used to implement each of the three models for the fraud classification.

Logistic Regression

Logistic regression is a linear model with the key advantage that it is easily interpretable. The log odds is a linear combination of explanatory variables, and their corresponding coefficients describe the weight of each variable on the predicting the label. Furthermore, a *p*-value can be calculated for each coefficient to assess the statistical significance of their direction of correlation with a particular outcome.

Multiple logistic models were fitted to the training data using 5-fold cross-validation for an array of hyperparameters. The best model had strong regularization (C=0.1) and used an Elastic-Net (linear

combination of lasso and ridge penalties) with an l1 ratio of 0.1 (indicating that the ridge (L2) penalty is more strongly weighted). Interestingly, all logistic models have similar predictive accuracy on the test data of approximately 80-81%. The best model won out with 81.2% for the accuracy on the test data. Furthermore, the test and training accuracies in all cases were remarkably close to one another, suggesting minimal overfitting.

Table 2: Machine learning models selected for classifying transactions along with their primary advantages and disadvantages.

Classifier	Advantages	Disadvantages
Logistic Regression	• Linear model and easy to regularize. • Probabilistic interpretation and can easily update classification threshold. • Easy to update model with new data	• Assumes linearity. • Not good for complex relationships between features and label.
Random Forest	• Ensemble method with low bias and moderate variance • Robust to outliers • Simple to implement "out of the box" classifier with minimal preprocessing. • Parallelizable	• Interpretability not as good as decision trees • Model can be sensitive to introduction of new data. • Manual transformations
Support Vector Machines	• Highly accurate with good kernel. • Deals well with non-linear data. • Stable to changes in data. • Effective in high-dimensional spaces. • Can deal with cases where number of dimensions exceeds the number of observations.	• Difficult to interpret. • Can be difficult to find good kernel function. • Memory intensive. • Long training time. • Need to scale data. • Need to carefully tune hyperparameters

The coefficients and *p*-values for the model are listed in Table 3. The variables *type_CASH_OUT* and *drain_Orig_account_No* (the transaction did not empty the originator's account) negatively correlate with fraud, whereas the remaining variables all were positively correlated with fraud. The associated *p*-values for the direction of correlation were calculated by bootstrapping training data and fitting 1,000 different logistic models. In all cases, the *p*-values are less than 1/1000 since never did the coefficients for the variables cross zero (see Figure 2).

Table 3: Logistic regression coefficients and *p*-values on all explanatory variables

Explanatory Variable	Coefficient	p-value
step	0.0030	< 0.000
type_CASH_OUT	-1.12	< 0.000
type_CASH_IN	0.23	< 0.000
drain_Orig_account_No	-2.21	< 0.000
drain_Orig_account_Yes	1.33	< 0.000
Amount_scaled	0.34	< 0.000

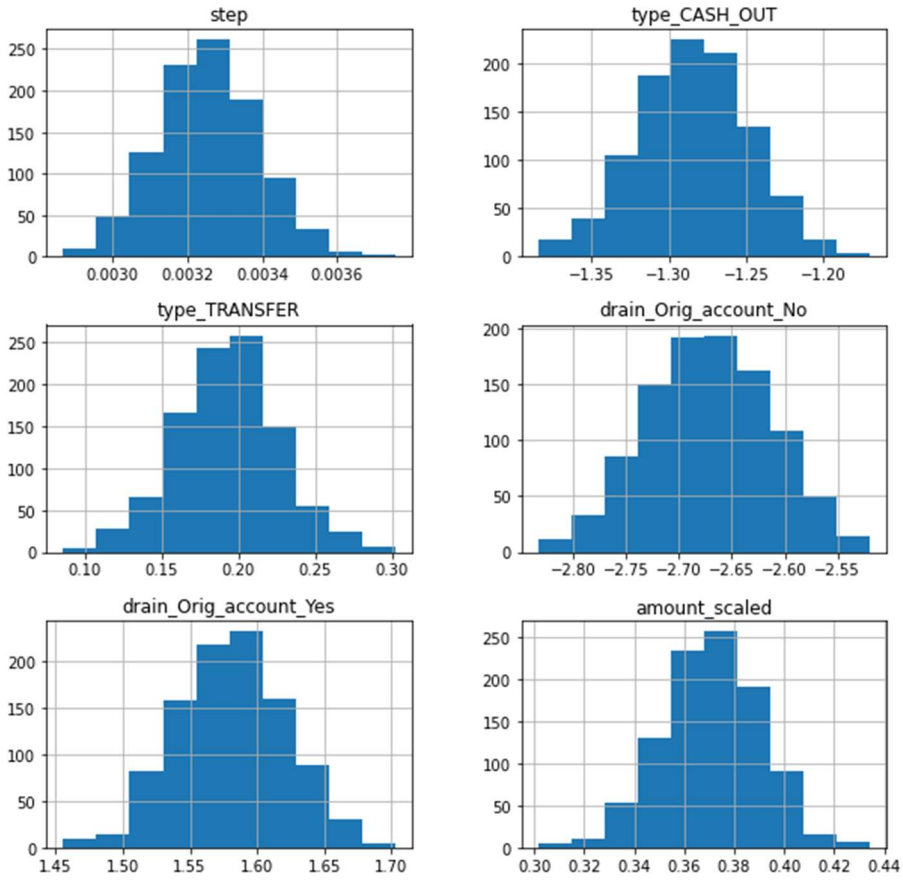

Figure 2: Bootstrapped (n=1000) distributions for logistic regression coefficients.

Although, the best logistic model's overall accuracy was 81.2%, computing the confusion matrix (Figure 3) revealed that this model classified fraud correctly 87% of the time while only predicting legitimate transactions correctly in three-quarters of the cases. Since fraud is very costly for MPS companies, it is good that transactions are predominantly correctly labeled. However, legitimate transactions should also be classified correctly lest they be held up for additional scrutiny and interfere with commerce. Since legitimate transactions comprise the vast majority of the data, having a high type 1 error rate (false positives) can therefore also be costly.

An additional step was performed to evaluate the importance of the *step* variable using a logistic model. Removing *step* resulted in a model that had a significantly lower true negative rate (65%) compared to the model that included the variable, but the accuracy for predicting fraud slightly improved to 88%. The result implies that *step* is important for identifying legitimate transactions rather than fraudulent ones. However, as pointed out in the Feature Selecting and Pre-processing section, while *step* is useful for classifying fraud for this set of data, it is unlikely to help model performance on unseen data occurring outside of the month-long time range that the model was trained on.

Random Forrest Ensemble

The random forest ensemble method builds a forest of decision trees with bootstrapped data (known as bagging) and then uses a random subset of features to train each tree. Random forests are becoming ubiquitous as a simple, out-of-the-box classifier or regressor for a variety of predictive challenges. They also are non-parametric (unlike logistic regression) and can deal well with non-linearly separable data.

Optimizing the hyperparameters with a grid search and 5-fold cross-validation, the best random forest model yielded an overall accuracy of 89.6% on the test data and 95.5% on the training data. The confusion matrix (Figure 3) shows a considerable improvement over logistic regression in the ability to correctly classify legitimate transactions. Furthermore, the random forest model also edges out logistic regression with fraud identification (91%).

From the Gini importance, defined as the total decrease in node purity weighted by the probability of reaching that node averaged over all trees in the forest, the relative importance of each feature can be gleaned. We see that *step* is the most important feature closely followed by the transaction amount. Next in importance are the features related to whether the originator's account was emptied. Finally, transaction type is the least important set of features in the random forest model.

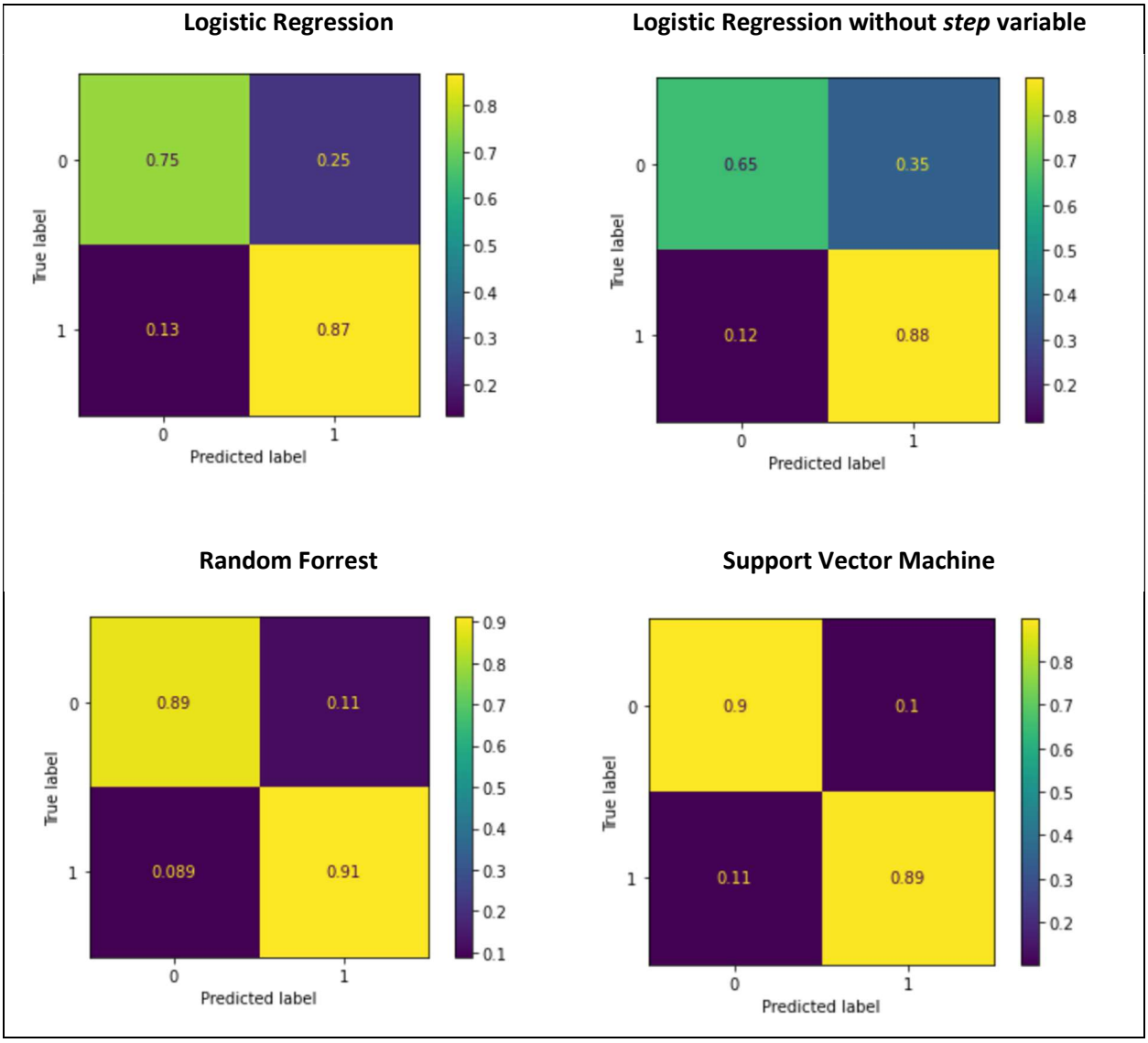

Figure 3: Confusion matrix for each model. Logistic regression has an additional confusion matrix for the case where the explanatory variable *step* is omitted.

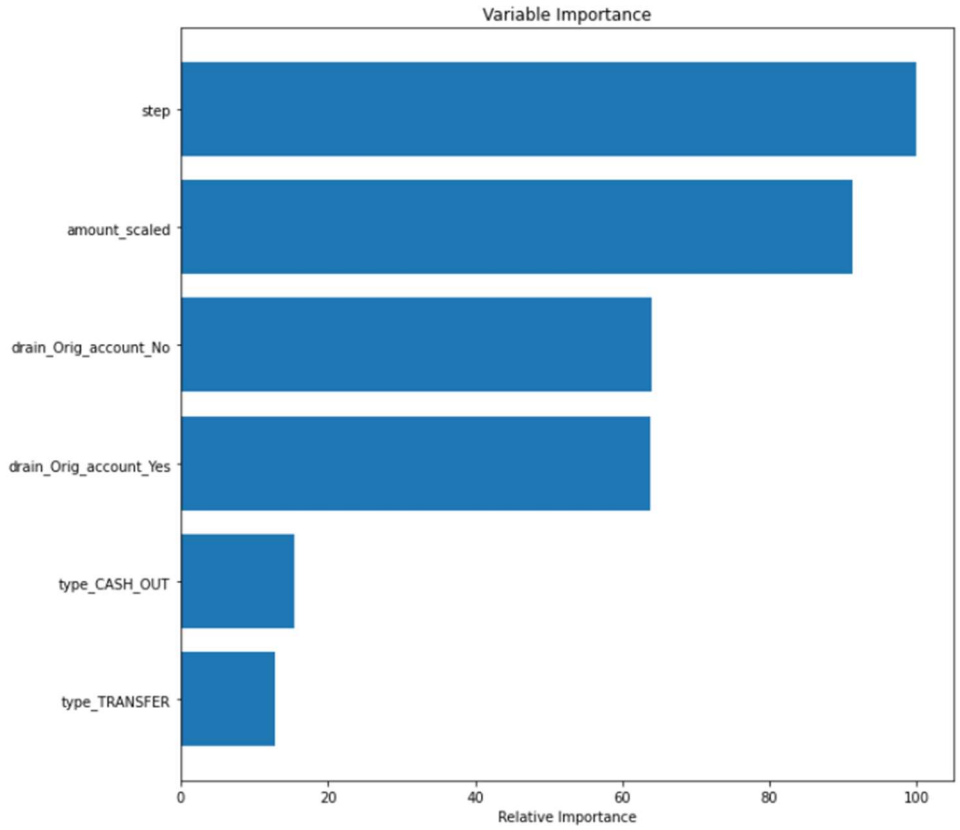

Figure 4: Gini importance plot ranking the importance of each feature in the best random forest model.

Support Vector Machines

The final model used to predict fraud in mobile money transfer transactions was the support vector machine (SVM) classifier. SVMs are among the best performing classifiers but can require careful hyperparameter tunning. SVM classifiers can separate even high dimensional data with a linear boundary using the so-called *kernel trick*, where the inner product of the model inputs is computing in a higher-dimensional space without the need to explicitly map the inputs to that same space. The major advantage of this trick is that the data becomes linearly separable in higher dimensions without the computational cost associated with explicitly transforming the data as only the chosen kernel function is computed.

With the usual methodology, a 5-fold cross-validation grid search was performed to find the best SVM model. Only a radial basis function (RBF) kernel was considered in the grid search, varying the hyperparameters C (regularization term defining the tolerance for misclassification and simplicity of the decision boundary) and gamma (unique to RBF, it determines the effect of a single training example on the model). Selecting appropriate C and gamma hyperparameters is paramount for the performance of

the SVM model. The best performing model has a low regularization penalty (C = 1000) and low gamma (0.01), suggesting that the optimal decision boundary is not smooth but that a single training example influences a large region around it.

The overall accuracy of the best SVM model is 89.6% on the test data, equal to our best random forest model. Whereas the SVM was slightly better at classifying legitimate transactions, the random forest was better at classifying fraud. However, without any hyperparameter tuning, the SVM model only achieved an accuracy of 70% on the test data, highlighting the importance of the tunning process.

Model Selection and Conclusions

By overall accuracy, the SVM and random forest models performed almost equally at close to 90% of the testing data correctly classified. Logistic regression fared markedly worse, with only about 81% of the test data correctly labeled. Another way to gauge the general performance of each model is from the precision-recall curves plotted in Figure 5. Precision-recall curves are favored over the more typical receiver operating characteristic (ROC) curves with class imbalanced data because of the role true negatives play. Although care was taken to balance the training and testing data, when evaluating the model's performance on more realistic imbalanced data, an ROC curve could look overly optimistic due to the high true negative rate. Precision-recall curves avoid the true negative rate entirely, and hence present a more realistic picture of the model's performance. One metric that can quantify the model's performance is its average precision (AP), which is the average precision of the model over a range of decision thresholds. The AP metric is the area under the precision-recall curve and is analogous to the area under the ROC curve. The AP from the prediction scores for the logistic model is 0.88, 0.95 for the random forest model, and 0.96 for the SVM. Therefore, the SVM model can be crowned winner using the AP score.

Although the logistic model did not perform as well as the random forest or SVM models, its predictions are more interpretable. We saw that only the CASH_OUT transaction type and transactions that did not empty the originator's account were negatively correlated with fraud, while the remaining explanatory variables were positively associated with fraud.

From the AP scores, the SVM model can be judged to be the best among the group. However, without extensive hyperparameter tuning, the model would have easily been the worst. Furthermore, the training for the SVM model took the longest. In contrast, the random forest model saw only marginal gains with hyperparameter tuning and was much faster to train since the process can easily be parallelized.

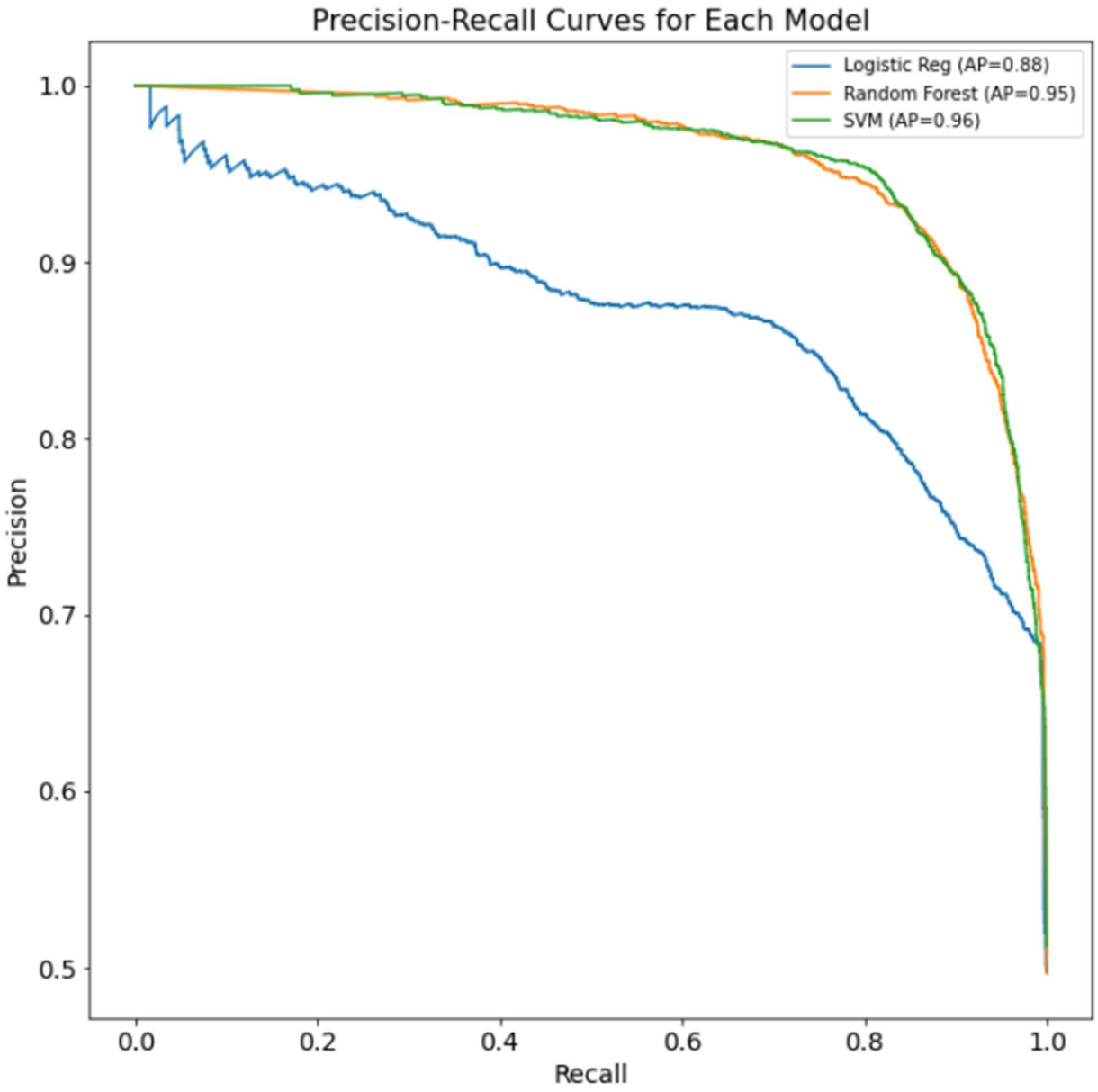

Figure 5: Precision-recall curves for best logistic regression, random forest, and support vector machine models.

Since the random forest model is more robust to the hyperparameter choices and produces essentially the same AP as the SVM, it can be considered a better model. In summary, there is no free lunch. Each model has its advantages and drawbacks, and the choice of model depends strongly on the objective.

Future Work

The logistic regression model did not consider any interaction between explanatory variables. It would be interesting to understand the type and nature of the interactions between the features (e.g., transaction type and amount) to better predict fraud. Including the interaction terms would also likely improve the performance of the logistic regression model.

Another important avenue for improvement would be to gain a better understanding of the *step* feature. In the random forest model, *step* had the highest Gini importance, and the exact time and date of the interaction can be strongly indicative of fraud if it does not align with a customer's previous habits. Any future work based on *step* would necessitate a more thorough understanding of the dataset that the PaySim simulator [4] was built on.

Social network analysis on the transactions could also identify suspicious individuals or groups in a population, providing a more sophisticated and real-time approach to identifying fraud as opposed to only analyzing patterns on a stale dataset. If location data could be made available in the mobile money transactions data, the coordinates of the phone associated with each transaction could also be used to identify fraud and those behind it better.

References

[1] Mobile financial services in Africa: Winning the battle for the customer, McKinsey & Company; https://www.mckinsey.com/industries/financial-services/our-insights/mobile-financial-services-in-africa-winning-the-battle-for-the-customer

[2] Digital Finance For All: Powering Inclusive Growth in Emerging Economies, McKinsey & Company; https://www.mckinsey.com/global-themes/employment-and-growth/how-digital-finance-could-boost-growth-in-emerging-economies

[3] Venmo takes losses after payments fraud; https://www.marketwatch.com/story/venmo-takes-losses-after-payments-fraud-2018-11-24

[4] PaySim: A Financial Mobile Money Simulator for Fraud Detection; https://www.researchgate.net/publication/313138956_PAYSIM_A_FINANCIAL_MOBILE_MONEY_SIMULATOR_FOR_FRAUD_DETECTION

[5] Synthetic Financial Datasets For Fraud Detection; https://www.kaggle.com/ealaxi/paysim1

[6] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.