**Predictive Modeling Against Financial Fraud**

By Rehan Khan

**Summary**

PaySim serves as a mobile money simulation tool specifically designed for detecting fraud. It generates data based on actual transactions sampled from a month's worth of financial logs from a mobile money service in an African country. This simulated data closely resembles the original records and proves valuable for conducting fraud analytics, particularly in situations where access to private records is restricted. For our analysis, we utilized one of PaySim's publicly available datasets on Kaggle. By leveraging this synthetic data, our goal is to gain a deeper understanding of features associated with fraudulent activities through data analysis, ultimately enabling us to make accurate predictions on unseen data. We further improved prediction performance by implementing various machine learning algorithms and rigorously evaluating each model.

*Data used for Capstone: https://www.kaggle.com/ntnu-testimon/paysim1*

*Reference: E. A. Lopez-Rojas , A. Elmir, and S. Axelsson. "PaySim: A financial mobile money simulator for fraud detection". In: The 28th European Modeling and Simulation Symposium-EMSS, Larnaca, Cyprus. 2016*

**Section 1: Data Wrangling**

We obtained simulated transaction data from the Kaggle website, saving it as a .CSV file in the 'data' directory. To facilitate analysis, we imported the file into Jupyter Notebooks as a DataFrame object and processed it using the Python programming language. The file was loaded into a DataFrame structure, consisting of 6,362,620 rows and 11 columns. For detailed information on each column, refer to Table 1.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Content** |
| step | integer | unit of time in the real world. 1 step is 1 hour of time. Total steps 744 (30 days simulation) |
| type | string/categorical | transaction type |
| amount | float | transaction amount |
| nameOrig | string | customer initiating transaction |
| oldbalanceOrg | float | initial balance before the transaction |
| newbalanceOrig | float | new balance after transaction |
| nameDest | string | customer who is recipient |
| oldbalanceDest | float | initial balance of recipient |
| isFraud | boolean | new balance of recipient after transaction |
| isFlaggedFraud | boolean | if transaction is flagged as fraudulent |
| newbalanceDest | float | new balance after recipient transaction |

*Table 1. Column Level Information*

The dataset initially contains fraud predictions in the 'isFlaggedFraud' column. The flagging system employed here is based on a straightforward criterion: setting the flag to 1 if a known fraudulent transaction involves a transfer exceeding 200,000 in the local currency. This approach, however, overlooks all other frauds involving amounts less than 200,000. In this report, we address the challenge of enhancing this initial prediction of financial fraud. Our approach involves exploring the dataset, selecting the most relevant variables, and applying various machine learning algorithms to achieve improved accuracy. See below for a snapshot of the first 10 rows of data from the dataset:

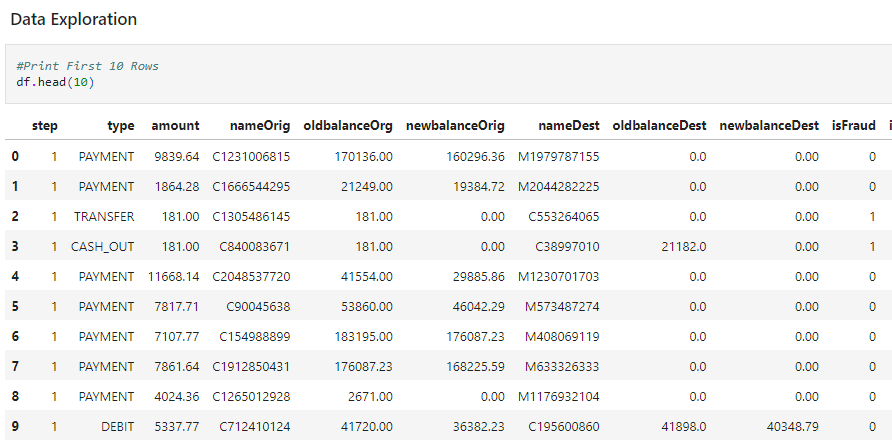


Figure 1: Data Exploration of Fraud Data

**Section 2: Exploratory Data Analysis**

We can see the different types of transactions that occurred with the fraud dataset by looking at Figure 2 below:

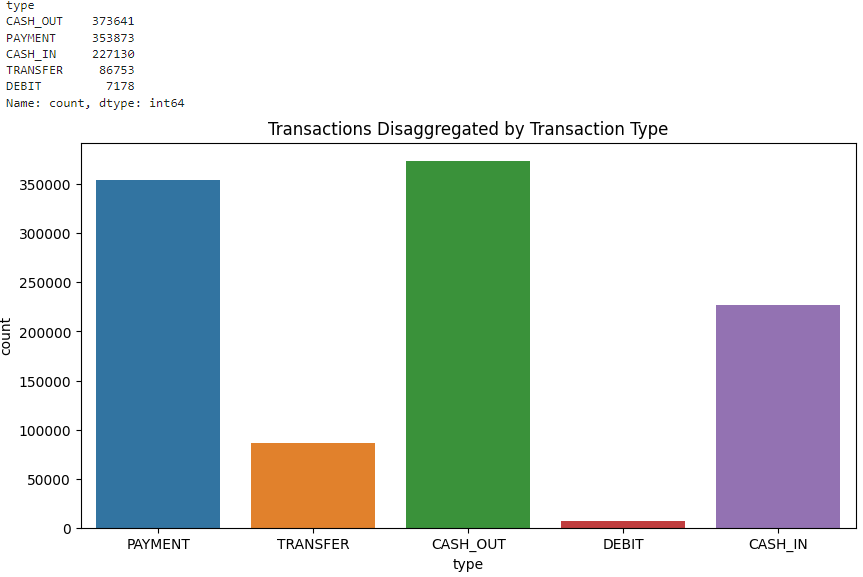


Figure 2: Transaction Types with counts

Here we can see that the largest number of transactions were payments and cash outs. After looking at the overall transaction differences, we wanted to understand the breadth of damage that was done with fraudulent transaction. To do this we looked at counts of the number of fraudulent transactions and how often they occurred. See Figure 3 to see the block of code that was used.

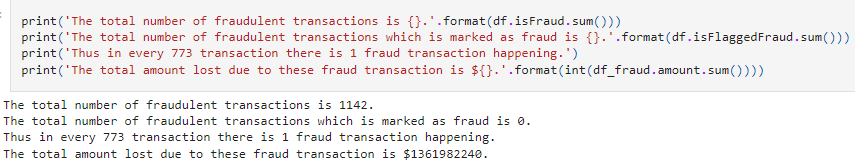


Figure 3: Number of Fraudulent Transactions

A correlation serves as a measure indicating the extent to which two random variables co-vary. In order to quantify the correlated relationships among all variables, we employed the “.corr()” function. This function evaluates the direction and strength of linear relationships, assigning scores between -1 and 1 (refer to Figure 3). Some column pairs exhibit perfect correlation, such as 'newbalanceOrig'-'oldbalanceOrig' and 'newbalanceDest'-'oldbalanceDest,' each with a Pearson’s r value of 1. This correlation is expected, given that these columns pertain to the amount of mobile money saved before and after a transaction.

Similarly, 'amount' demonstrates a relatively strong correlation (r = 0.3 - 0.5) with the aforementioned four columns, as it represents the difference between old and new balances. In the context of this project, which addresses a classification problem involving predicting highly unbalanced binary class labels ('isFraud' with values 0 and 1), selecting relevant features is crucial for constructing an effective prediction model.

Despite the importance of feature selection, using correlation as a metric did not allow us to narrow down our focus to a few features. None of the columns exhibited a satisfactory correlation score with the target variable 'isFraud' (all measured values were below |0.1|)

|  |  |
| --- | --- |
| **Column Name** | **r** |
| amount | 0.077 |
| isFlaggedFraud | 0.044 |
| step | 0.032 |
| oldbalanceOrg | 0.010 |
| newbalanceDest | 0.001 |
| oldbalanceDest | -0.06 |
| newbalanceOrig | -0.008 |

*Table 2. Correlation coefficient between ‘isFraud’ and featured columns*

**Section 3: Data Pre-processing**

Effective data preprocessing stands out as a vital stage in readying raw data for utilization in any machine learning algorithm. Given that the existing data was deliberately generated, it remains relatively clean without any instances of missing values, obviating the need for removal or imputation procedures. However, a crucial step involves transforming categorical data with string values, necessitating the encoding of non-numeric values into numeric equivalents. This transformation becomes imperative, particularly for models like logistic regression and neural networks that exclusively process numeric inputs.

For this step, I combined categorical values converting them to a new “type” to include information to process into the final machine learning algorithms I will use to test and predict our data. I did this using a numpy function to initialize a new “Type2” column while using the “.loc” function. See below:

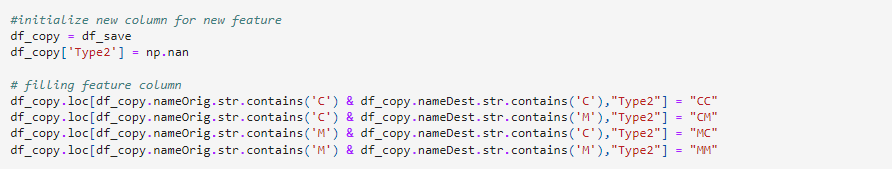


Figure 4: Data preprocessing in preparation for model training

Since not all features are providing information relevant to detect fraud, it is necessary to select only the best indicators for our target and build a model based on these features only. For this report, we selected following 10 features:

● Type2

● type

● amount

● oldbalanceOrg

● newbalanceOrig

● oldbalanceDest

● newbalanceDest

● validTransactionType

Columns that were excluded for generating models are: ‘isFlaggedFraud’, ‘nameOrig’ and ‘nameDest’. 'nameOrig' and 'nameDest' are nominal values containing the identity of each individual involved in a transaction and these are better represented in a newly created column, ‘validTransactionType’. Therefore, we decided to remove the name columns to reduce noise. In addition, ‘isFlaggedFraud’ was excluded from the feature list because it was irrelevant to our model.

**Section 3: Modeling**

Binary target label prediction through classification falls within the realm of supervised learning. In this paradigm, a model learns from labeled training data and subsequently predicts labels for unseen data. Given the singular dataset at hand, it becomes essential to partition it into two subsets: one for training the model (80% of the data) and another for testing its performance (20% of the data). This division was achieved using the train\_test\_split method provided by scikit-learn.

In the preceding step, we established both a training and testing set, laying the groundwork for constructing our initial and fundamental model as a baseline. This baseline serves as a reference point, offering valuable insights into potential enhancements for our subsequent models. To establish this baseline, we initiated the process by standardizing the data, ensuring a uniform input regularization. Subsequently, we applied the logistic regression algorithm from the scikit-learn library.

Standardizing the data involves rescaling all variables to a comparable range, resulting in a mean of 0 and a standardized distribution with a standard deviation of 1 for all variables. This standardization ensures that variables, originally measured in different units, now have a similar scale. Using the StandardScaler from the scikit-learn library was our choice among various available standardization techniques (e.g., MinMaxScaler, RobustScaler, Normalizer). The StandardScaler guarantees that all features share the same mean and unit variance.

For this model building exercise, three different algorithms were used. These are algorithms are Random Forest, Decision Tree Classifier and Logistic regression. Random Forest is an ensemble learning algorithm used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees. A decision tree is a tree-like model used for both classification and regression tasks in machine learning. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes.A widely-used and straightforward method for class prediction involves utilizing a logistic function, specifically a sigmoidal function. Logistic regression, a classification model, predicts the probabilities of a sample observation being labeled as class value 1. If the probability is equal to or greater than 0.5, the outcome is '1'; otherwise, it is '0'. In this context, we instantiated the Logistic Regression object from the scikit-learn library and fitted it with our training data. Following the fitting process, we generated predictions using features from the test data.

**Model Evaluation**

After we built the models, we evaluated which model was the most accurate for predicting cases of fraud. Find the results of the three models below:

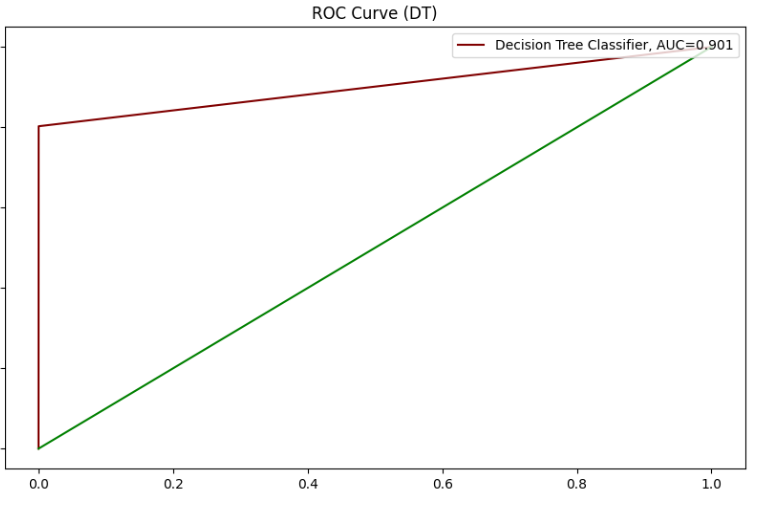


Figure 5: Model 1- Decision Tree

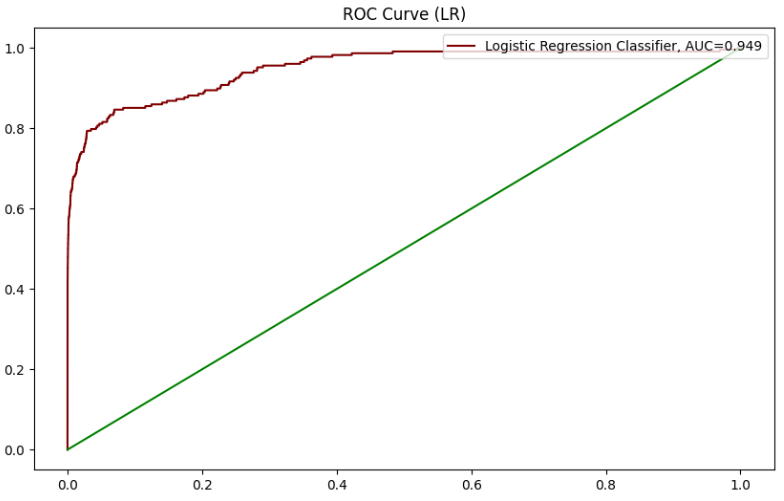
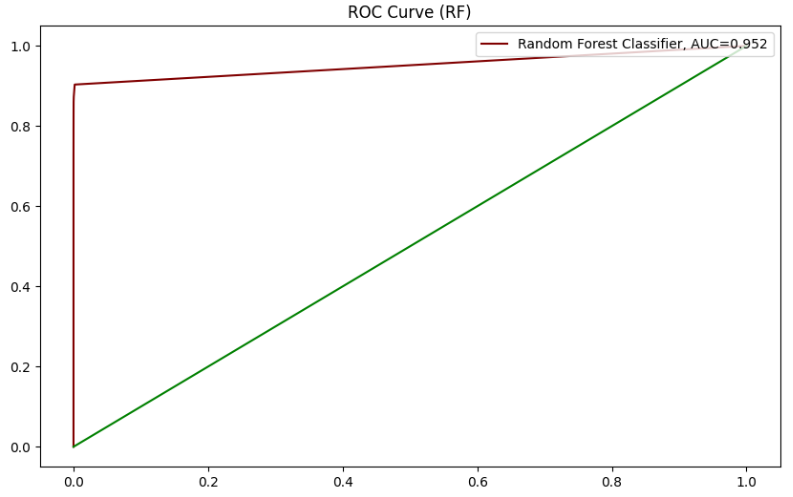


Figure 6 Model 3 - Random Forest

Figure 7 Model 2 - Logistic Regression

**Section 4: Conclusion**

In summary, we crafted advanced predictive classifier models to enhance the efficacy of our financial fraud detection system. The outcome was a notable improvement in detecting fraudulent operations, achieving higher recall scores (up to 0.99) across all three distinct algorithms compared to the original 'isFlaggedFraud' or baseline model. Here are our key findings:

1. We found that the Random Forest Algorithm Performed the best out of the three algorithms with an AUC score of 0.952.

2. Decision Tree algorithm performed the worst with an AUC score of 0.901.

3 RF need to be optimized by further hyperparameter tuning.The total number of fraud transactions were 1142 transactions. These fraud transactions were either Transfer or Debits

By optimizing the present models and by introducing more complicated dataset with extra features, we expect to address a practical solution for mobile money transaction business to detect fraud with high accuracy.