# Credit card fraud identification in financial payment services

Kamil Madey

# Introduction:

*As technology has evolved and more people use the internet for transactions, credit card fraud has continually increased. Since the COVID-19 pandemic, credit card fraud has increased even more than normal, up to 44.7%. The United States has ranked among the top countries where this occurs. Companies are continuously searching and improving fraud detection systems. The main crux of the issue is that transactional data tends to be private to a company so improving upon real data must come from within a company. Other areas where data is publicly available have more robust solutions to their relative problems. In the case of credit card fraud, synthetic data is often used to help develop models that can work well on real world data. In this project, I create a fraud detection system that helps accurately classify a transaction as fraud/non-fraud and reduce the fraud loss rate.*

# Data:

*Since we have a simulated dataset available to us, we can use this to construct our model. PaySim utilizes aggregated real-world data taken from a private dataset to generate synthetic transactional data resembling normal transactions. This real-world data is a sample of real transactions extracted from financial logs from a mobile money service implemented in an African country. Since the data is synthetic, it does contain irregularities. Some of which include the numbers not adding up between transactions, features with no correlation, etc. These are addressed in the EDA and feature engineering sections.*

# Problem statement:

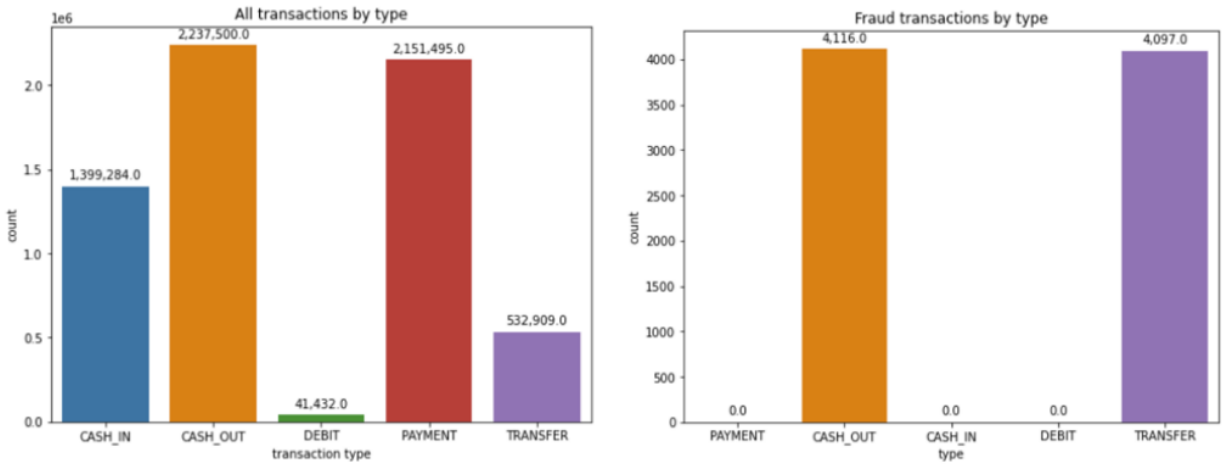
*The most common strategy in place for identifying potentially fraudulent transactions is to place a limit on the transaction amount; this is what the PaySim system attempts to do. This approach can work but it does have downsides and can be exploited and negatively impact the customer experience.*

*In the current dataset, system in place has a fraud loss of just over $12 billion. In our business scenario, we work for a payment processing company which has identified a major flaw in its technology. Too many of their customers are losing money to fraudulent transactions. Specifically, the system is not catching fraud transactions and thinks a lot of them are legitimate.*

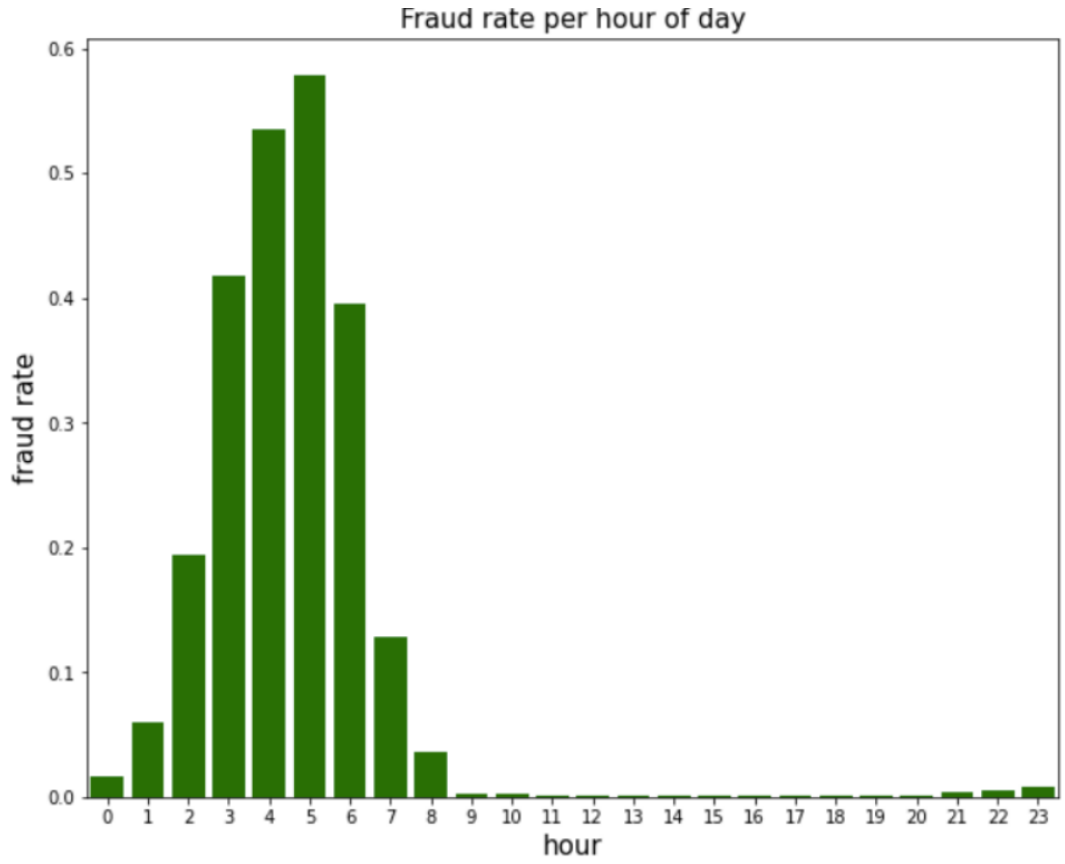
*The goal is to create a more robust and intelligent classifier to be able to catch these transactions and reduce the fraud loss rate to within market levels. Currently, the system in place is broken so the fraud loss rate per $100 is absurdly high at $1.37, where the average in the market ranges from $0.06 to $0.28.*

# EDA:

*We see the frauds only occur from two types of transactions. Reducing our data to these two reduces our heavy imbalance by lowering data size from 6.3 million to 2.7 million transactions.*




*We can also look at how fraud occurs depending on the time of day. We see a normal distribution of fraud occurring in the early morning hours. This is used to generate one of the features, which is based on when the transaction occurs.*



# Feature Engineering:

*We use the above to generate a binary feature where a label of 1 is given if the transaction occurred during peak times (between midnight and 8 am) and 0 elsewhere.*

*We also noticed that for the fraud transactions, almost all of them were a transfer of a certain amount followed by a cash out. We turned this into a binary feature as well where if this was the order, then the transaction and cash out were given a 1.*

*We created dummy variables for the 'type' of transactions. This was our only categorical variable.*

*Additionally, in the EDA section we identified that there were balance errors for most of the transactions. The difference between the originating and destination accounts did not match up with the amount transferred. Rather than a binary variable, this was made into a numeric variable where the difference between the amount and the origin and destination accounts was input.*

# Random sampling and scaling:

*We introduce scaling and undersampling into our models.*

*Generally, with skewed datasets such as the fraud dataset we are working with we run into the issue of imbalanced data. This imbalanced data influences the performance of a ML model. We need to use methods such as undersampling or oversampling so that our model isn't influenced by the highly skewed data.*

*Here we use random undersampling since we have a large amount of the majority data. Random sampling is a simple technique that assumes nothing of the data. There are other methods for undersampling and oversampling that may be used if random sampling is deemed too simple.*

*Performing feature scaling is dependent on which ML model you use. In some cases, scaling the features can help model performance, in other cases it does not.*

*Gradient descent based algorithms require features to be scaled, and this is why it is used here with logistic regression and with the gradient boosting model. On the other hand, tree-based algorithms are not affected by scaling, this is why it is not used in the random forest model.*

# Metrics for success:

*Selecting the correct metric to use for our model is very important. Since our data is highly imbalanced, accuracy would not be a good metric to evaluate our models. Additionally, since we are dealing with a binary classification (either fraud or non-fraud) instead of probabilities, then we can rule out using metrics like AUC curve or PR score, which are more useful for probabilities. With imbalanced data, we should focus on using metrics such as precision, recall, or F1 score.*

*The metric we use is dependent on the business problem. In our case, the impact of mislabeling a fraud transaction as nonfraud is much more damaging than having a nonfraud labeled as Fraud. Thus, using recall as a metric, (defined as TP / TP + FN) is what we will use. A secondary option would be to use F1 as a metric, which acts as a harmonic mean between precision and recall. This metric may be a better option to use in another business context.*

# Modeling and evaluation:

*We compare several models and the results of them are shown below:*

| | Recall_score | precision_score | fit_time | test_time |
|---|---|---|---|---|
| **Dummy_classifier** | 0.001623 | 0.001610 | 0.078020 | 0.026092 |
| **Logistic_regression** | 0.857955 | 0.036657 | 30.059483 | 0.072475 |
| **Decision_Tree** | 0.995942 | 0.792892 | 68.321522 | 0.082258 |
| **RF_us** | 0.996347 | 0.930277 | 409.496463 | 4.973349 |
| **RF_no_us** | 0.996347 | 0.999186 | 1539.640552 | 4.822654 |
| **RF_2_feat** | 0.994318 | 0.133326 | 117.292306 | 0.605890 |
| **Gradient_Boosting** | 0.996753 | 0.924003 | 230.783098 | 0.991730 |

*In our business context we are focusing on minimizing the number of fraud transactions that are misclassified as non-fraud. Thus, as mentioned earlier, we focus on recall score. In another context, f1 may be a good score to use to optimize the misclassified transactions overall, including both precision and recall.*

*There are several algorithms we've chosen that have high recall, which is what we were looking for! The top models are:*

*- The decision tree*

*- The random forest with undersampling*

*- The random forest with class weight*

*- Gradient boosting*

*All of these have recall accuracy around 99.6% or higher. The difference then falls to precision and computation time. Precision in our case is being able to correctly classify nonfraud transactions so we don't have a situation where a legitimate transaction is blocked because its flagged as fraud by our system. Our choices for best model reduce to:*

*- Random Forest with class weight*

*- Random Forest with undersampling*

*- Gradient boosting*

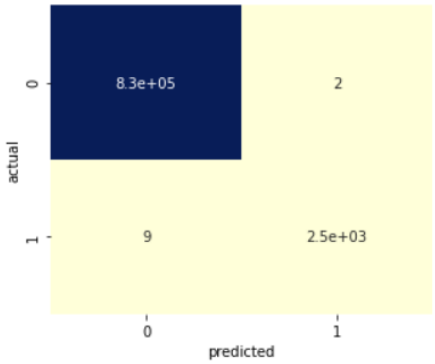
*The best one here is the random forest with class weight. If we were ignoring precision, we could choose gradient boosting since it had a better recall and is computationally less expensive. However, 8% of*

*precision accuracy is more valuable to us than 4 seconds of computation time. Our winning model becomes random forest with class weight.*

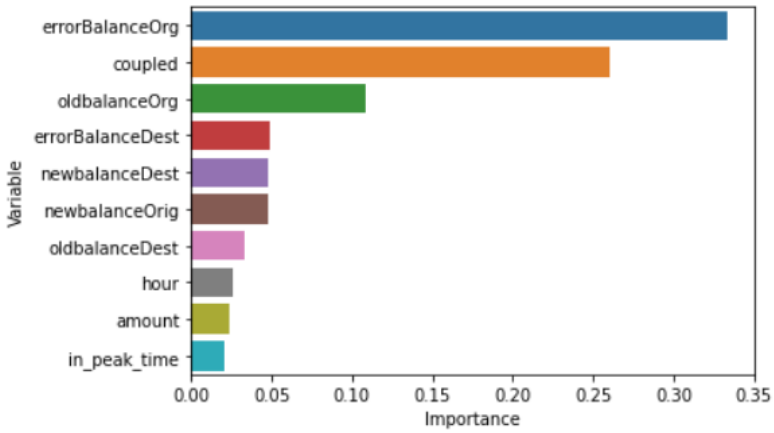
# Winning model:

*Our winning model is the random forest with class weight. We see its metrics below:*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 828659 |
| 1 | 1.00 | 1.00 | 1.00 | 2464 |
| accuracy | | | 1.00 | 831123 |
| macro avg | 1.00 | 1.00 | 1.00 | 831123 |
| weighted avg | 1.00 | 1.00 | 1.00 | 831123 |




*We see the confusion matrix and classification report above. We can also get a better look at the feature importances below. Notice that the features we created play a big role in model performance.*




# Conclusion:

*The measure we were looking to calculate was fraud loss rate per $100 transacted. Before building our model, our loss rate was $1.37 per $100 transacted. This was several magnitudes worse than industry average.*

*A top company like AMEX has a fraud loss rate of 0.06, PayPal has one at 0.28.*

*For our model we were able to get a fraud loss rate of 0.0012, which would be a standard in the industry.*

*One thing to note however, is that the top companies rank this based on a year basis. The dataset used was limited to a one-month period. Additionally, the dataset is synthetic, it would be nice to be able to work using real-world data and with a dataset going further back than a month so we can get a more accurate representation of the effectiveness of our model.*