Building a model for identifying credit card fraud

Kamil Madey

The Problem

Industry:

Over $28 billion lost to credit card fraud in 2019 and has been increasing ever since, up 44.7% from that since COVID-19.

Our business:

Fraud detection system broken/not effective.

$12 billion lost of fraudulent transactions with a fraud loss rate of $1.37 per $100.

1. What factors are correlated to fraud transactions?
2. Can we develop a model using these to classify new transactions?

Data Information

Synthetic dataset generated based on real world data taken from a private dataset that resembles normal transactions through a simulator called PaySim.

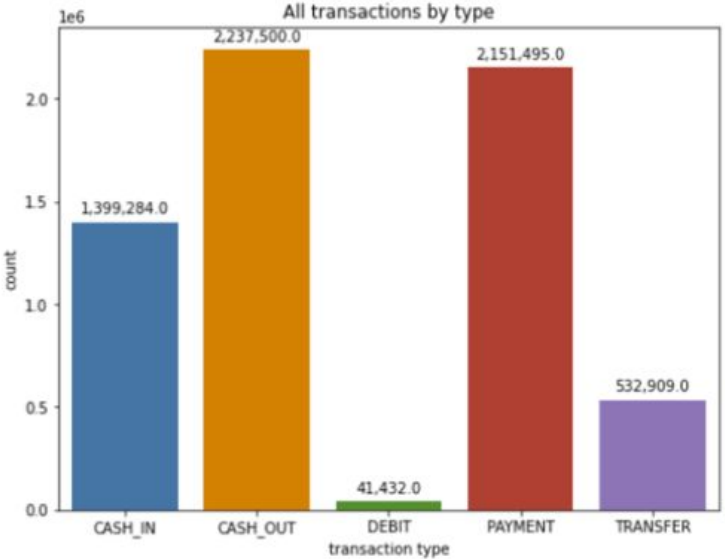
The real world data is a sample of transactions taken from financial logs from a mobile money service implemented in an African country.

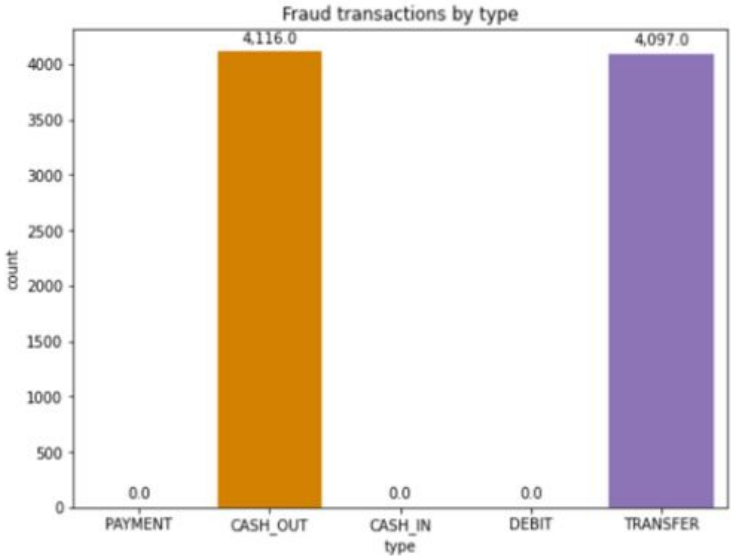
- 6.3 million transactions over a 1 month period
- 8213 fraudulent transactions

Challenges:

- *Synthetic data may not fully work in a model like real world data*
- *High class imbalance: only 0.13% tagged class 1*

Data Exploration

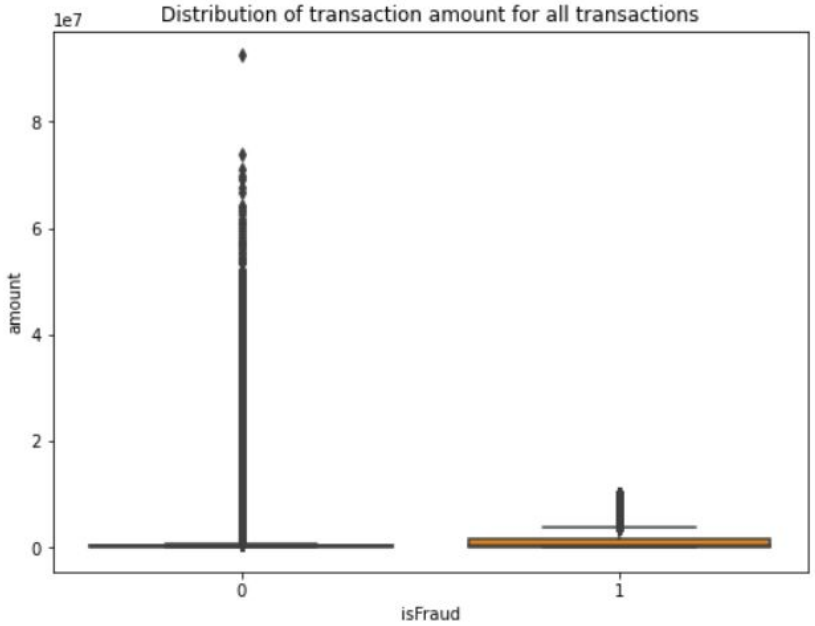


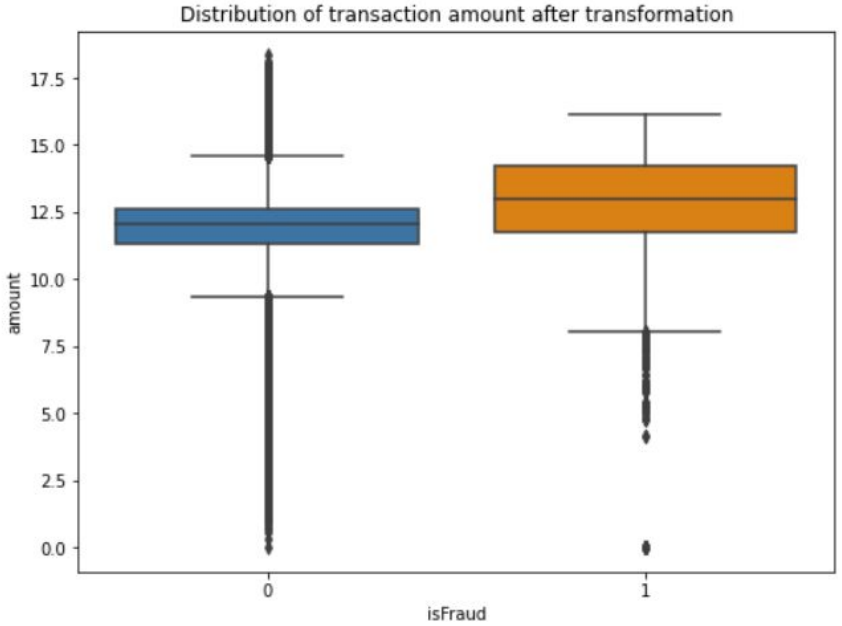


Not all transactions have fraud. We can reduce our data from 6.3 million to 2.7 million reducing our dataset to cash outs and transactions only.

Slightly reduces the magnitude of imbalance.

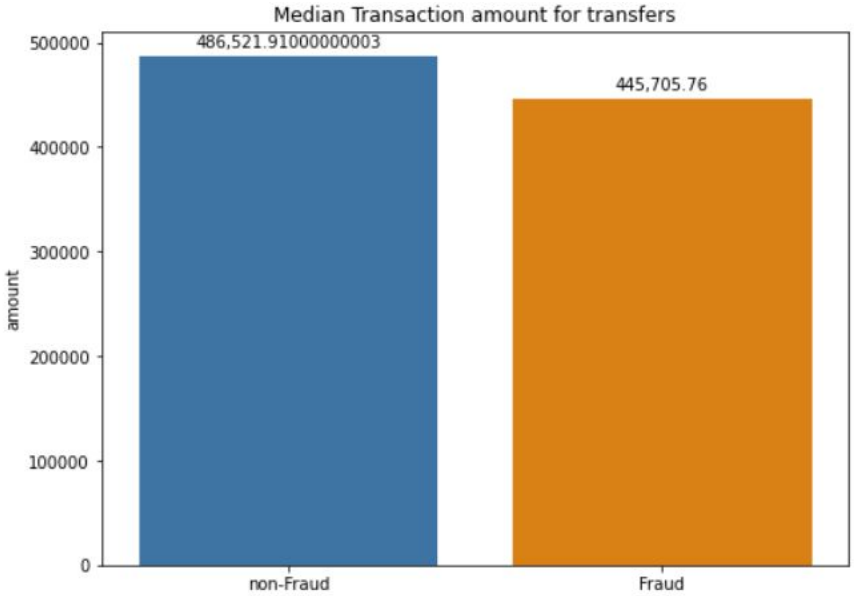
Data Exploration

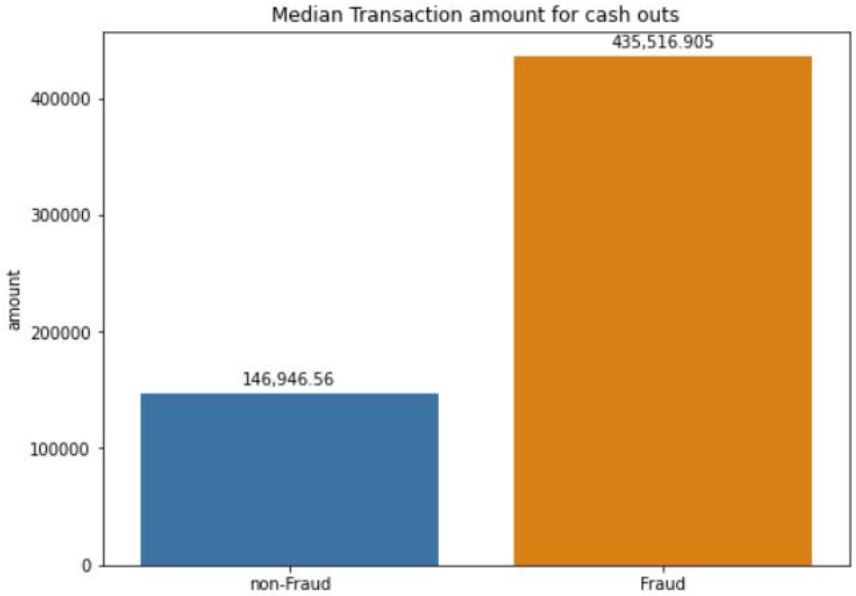




The amount distributions and outliers for both fraudulent and valid transactions.

Data Exploration



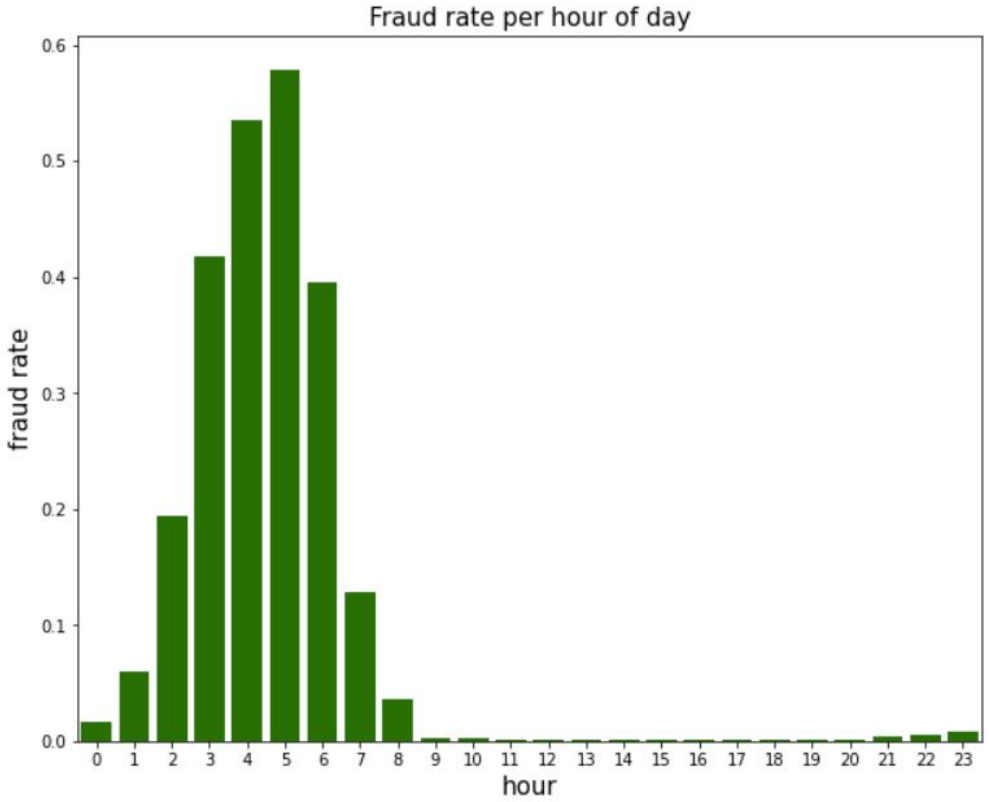


Because of the amount of outliers. We can use median to get a sense of central tendency as it pertains to the amounts.

Notice for cash outs the fraud transactions are generally much higher amounts.

Data Exploration

Looking at when transactions occurred, we see that the vast majority of fraud transactions occurred between midnight and 8am.



Machine Learning Modeling Overview

Type: Supervised Learning

Tools: Python's scikit learn and imblearn

Binary classification: 1 for fraud transaction, 0 for valid transaction

Class imbalance: Only 0.13% class 1. Needs to be addressed.

Modeling steps

Data Pre-processing:

1. Label encoding
2. Data split using train/test split (70/30 split)
3. Resampling or weighing to solve for class imbalance.
4. Scaling if needed.

Cross-validation for hyperparameter tuning:

1. Using GridSearchCV from scikit learn
2. Evaluation metric: f1 score

Train classifier using optimized parameters and 70% of data (training data).

Performance evaluation using holdout set.

Steps piped using imblearn Pipeline class.

Strategies for dealing with class imbalance

Resampling techniques (imblearn):

- Random Undersampling

Weighting technique (sklearn):

- Using class_weight (= 'balanced') in several sklearn classifier implementations.

Classification algorithms used:

- *Dummy classifier*
- *Logistic regression (with both)*
- *Decision Tree (with both)*
- *Random Forest with undersampling*
- *Random Forest with weighting technique*
- *Random Forest with 2 most important features*
- *Gradient boosting (with undersampling)*

Model comparisons

Prioritizing recall score. We want to optimize the accuracy of correctly classifying fraudulent transactions.

Winning model based on prioritizing recall score accuracy over time = *Random Forest with Weighting technique*

	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

Winning model parameters:
Max depth: 10
N_estimators: 100
Sampling technique: weighting

Winning model performance evaluation

Winning model: Random Forest with weighting technique

Used on the holdout set:

Total transacted amount: $2.65 billion

Amount lost to fraud: $3.1 million

Fraud loss rate: 0.00118 per $100

Our new model performs much better now, we are able to catch nearly all of the fraudulent transactions.

Implications

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.

Conclusions/Future Improvements

- In the future, I would love to spend more time trying different techniques for dealing with imbalanced data, such as SMOTE, Tomek Links, etc.

- This fraud detection system could also be improved by introducing more data to it. I would like to find a way to generate the synthetic data myself to feed into the model and test it.

Thank you!