Fraud Detection Modeling

Thinkful Supervised Learning Capstone

Jonathan-Cassidy Ong

Background

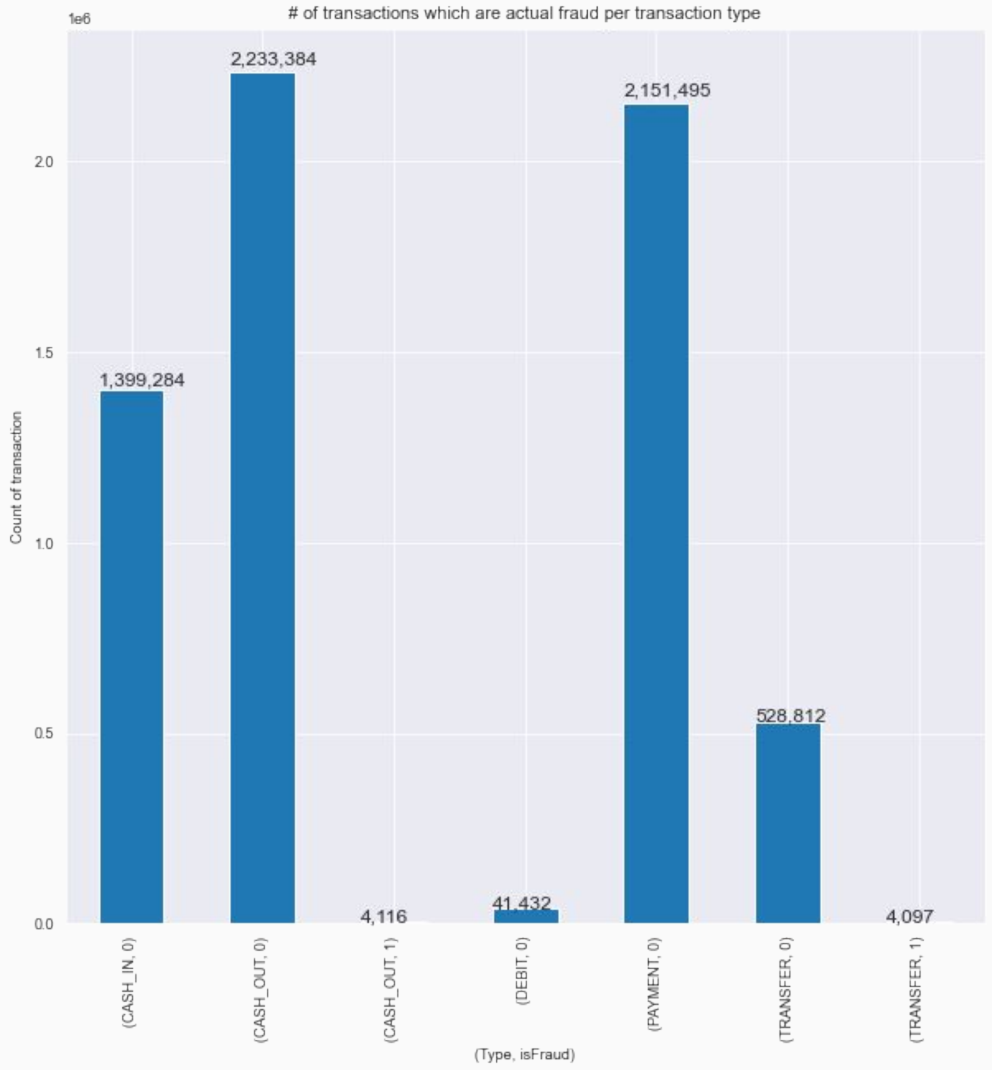
- Kaggle dataset: Synthetic Financial Datasets For Fraud Detection
- Synthetic dataset used in financial analyses for fraud detection using Paysim
- Fraud in the context of this project is defined as fraudulent mobile money transactions where an agent attempts to gain access to a customer's account and empty the funds by transferring to another account and cashing out of the system

How Important is Fraud Detection?

- Mobile money transactions are quickly becoming the most widely used form of transactions
  - Venmo, Zelle, Apple Pay, Samsung Pay, Google Wallet, Ali Pay, Paypal, and the list goes on
  - If we make these forms of payment safer, usage and global acceptance of this practice would increase further
- By creating an accurate classifier, we can further the growth of the mobile money transactions

What are the chances?

- With fraud, the ultimate goal is to TRANSFER and CASH OUT in order to successfully steal money
- If this classifier is able to properly and accurately determine which transactions are fraud, then implementation of the classifier may be done to protect the targeted account



What do we want to accomplish?

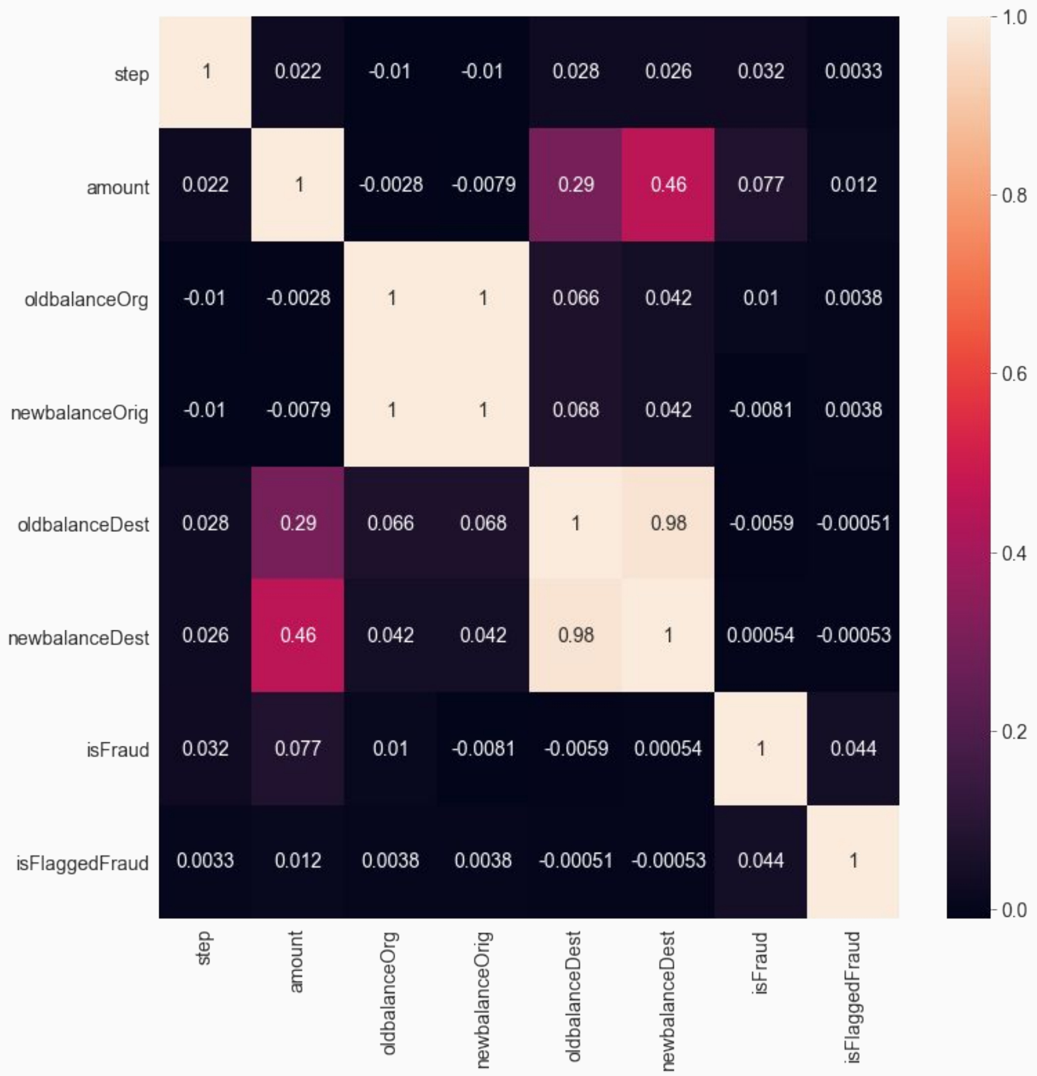
Use the data to create a model with specifications that make it as fast and as accurate as possible. Speed isn’t always the most important factor, but in this case, I want to find the quickest, most accurate, and most practical application of this model, so speed will come into effect here.

Exploring the Data

- There are 11 features in this particular dataset:
  - Continuous: Step (each step is 1 hour of time, totaling to 744 in our 30 day simulation), Old/New Balance Owner, amount, Old/New Balance Destination
  - Categorical: Type, nameOrig (customer who started the transaction), nameDest (recipient of the transaction), Fraud, Flagged as Fraud (isFlaggedFraud)
- We will handle the categorical data by assigning dummies

Correlation in the Dataset

- Here we check if there's too much multicollinearity between variables which would skew our results
- Of the 6 million rows, we had 0 missing values and only 8,213 of those values were flagged as fraud



	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
step	1	0.022	-0.01	-0.01	0.028	0.026	0.032	0.0033
amount	0.022	1	-0.0028	-0.0079	0.29	0.46	0.077	0.012
oldbalanceOrg	-0.01	-0.0028	1	1	0.066	0.042	0.01	0.0038
newbalanceOrig	-0.01	-0.0079	1	1	0.068	0.042	-0.0081	0.0038
oldbalanceDest	0.028	0.29	0.066	0.068	1	0.98	-0.0059	-0.00051
newbalanceDest	0.026	0.46	0.042	0.042	0.98	1	0.00054	-0.00053
isFraud	0.032	0.077	0.01	-0.0081	-0.0059	0.00054	1	0.044
isFlaggedFraud	0.0033	0.012	0.0038	0.0038	-0.00051	-0.00053	0.044	1

Resampling the Data

- Because we have such a high disparity between fraudulent and non-fraudulent cases, any model we train would likely be 99% accurate when sampling from these values.
- Instead, we will resample the data, allowing for us to see the actual effect of the model on the data
  - 100,000 cases each of True and False for fraudulent transactions

The Methods

- We have two main methods in which we will be selecting features
    - SelectKBest
        - SKB will remove everything except for highest scoring features, scored by the f-test to determine if there are any statistical significance in the variance of any possible features
    - Principal Component Analysis
        - PCA uses all the features to transform into brand new features that are completely independent from one another
        - Most variance was explained with up to 4 or 5 components, so we chose to have 4

Performance Expectations

- Find the best parameters with the best classifying score using Grid Search CV
- Average 5 Fold Cross Validation Score to resample data into 5 equal sized samples and running the model through each of them to determine average score
- Classification Report (Average scores between the fraud and non-fraud cases)
    - Precision (TP/(TP+FP)) is the ability of the classifier to not label as positive a sample that is negative
    - Recall (TP/(TP+FN)) is the ability to find all the positive samples
    - F1 is the weighted average of the precision and recall, where an F1 scores reaches its best value at 1 and worst score at 0
- AUC is the Area Under the Curve, which represents both high recall and high precision. In this case, the model has both a low false positive and false negative rates because of high precision and high recall, respectively

Model 1: Gaussian Naive Bayes

Gaussian model was chosen initially due to the presence of more continuous variables than categorical types. So the only way to incorporate the categorical was using the Gaussian model, unlike the Bernoulli and Multinomial NB.

- SKB Results
    - Best Parameters: 3 Features
    - Avg. CV: 77.88% | | Avg. Precision: 85% | | Avg. Recall: 78%
    - Avg F1: 77% | | AUC: 15.21% | | Runtime: Fast
- PCA Results:
    - Avg. CV: 83.63% | | Avg. Precision: 84% | | Avg. Recall: 84%
    - Avg. F1: 84% | | AUC: 39.57% | | Runtime: Fast

Model 2: KNN

To use SKB, the distance was normalized in features

- SKB Results
  - Best Parameters: 8 Features, 2 Neighbors
  - Avg. CV: 99.35% | | Avg. Precision: 99% | | Avg. Recall: 99%
  - Avg F1: 99% | | AUC: 49.70% | | Runtime: Slow, ~2 Hours
- PCA Results:
  - Best Parameters: 2 Neighbors
  - Avg. CV: 98.85% | | Avg. Precision: 84% | | Avg. Recall: 82%
  - Avg. F1: 82% | | AUC: 38.45% | | Runtime: Fast
- Both models overfitted drastically. The overfitting did reduce very slightly with PCA, but not enough to matter

Model 3: Decision Tree

- SKB Results
    - Best Parameters: 12 Features, 8 Max Depth
    - Avg. CV: 98.95% | | Avg. Precision: 99% | | Avg. Recall: 99%
    - Avg F1: 99% | | AUC: 48.70% | | Runtime: Fast, ~30 Seconds
- PCA Results:
    - Best Parameters: 8 Max Depth
    - Avg. CV: 93.26% | | Avg. Precision: 87% | | Avg. Recall: 85%
    - Avg. F1: 85% | | AUC: 40.79% | | Runtime: Fast, 2 Seconds
- Both models seem to do well, but the SKB model again seems overfitted probably due to high depth. The PCA looks slightly better

Model 4: Random Forest

- SKB Results
  - Best Parameters: 9 Features, 50 Trees, 6 Max Depth
  - Avg. CV: 97.50% | | Avg. Precision: 97% | | Avg. Recall: 97%
  - Avg F1: 97% | | AUC: 47.49% | | Runtime: Slow, ~2 Hours
- PCA Results:
  - Best Parameters: 10 Trees, 8 Max Depth
  - Avg. CV: 93.28% | | Avg. Precision: 88% | | Avg. Recall: 87%
  - Avg. F1: 87% | | AUC: 41.80% | | Runtime: Moderate, 40 Minutes
- Both models used the maximum depth allowed with GridSearchCV. PCA model needed a lot less estimators than SKB. This model seems strong due to its high AUC and it is less likely to have overfitted the data

Model 5: Logistic Regression

Used Ridge Regularization due to high amount of rows in order to prevent overfitting as much as possible.

- SKB Results
  - Best Parameters: 8 Features, C (inverse of regularization strength) = 1e-05
  - Avg. CV: 90.51% | | Avg. Precision: 91% | | Avg. Recall: 91%
  - Avg F1: 91% | | AUC: 42.79% | | Runtime: Fast, 2 Minutes
- PCA Results:
  - Best Parameters: C = .001
  - Avg. CV: 82.81% | | Avg. Precision: 84% | | Avg. Recall: 84%
  - Avg. F1: 84% | | AUC: 38.15% | | Runtime:Fast, 4.6 Seconds
- SKB ended up being the better model and does not seem as overfitted.

Model 6: Gradient Boosted Model

- SKB Result
  - Best Parameters: 12 Features, 1000 Estimators, 6 Max Depth
  - Avg. CV: 99.83% | | Avg. Precision: 100% | | Avg. Recall: 100%
  - Avg F1: 100% | | AUC: 0.14% | | Runtime: Very Slow, 8+ Hours (Reads 16 hours in notebook, though this was due to outside factors)
- PCA Results:
  - Best Parameters: 1000 Estimators, 0.3 Learning Rate, 8 Max Depth
  - Avg. CV: 99.41% | | Avg. Precision: 85% | | Avg. Recall: 81%
  - Avg. F1: 81% | | AUC: 39.46% | | Runtime: Long, ~5 Hours
- GBM is extremely greedy. Used maximum amount of parameters allotted in GridSearchCV. The learning rate inhibition did little to prevent overfitting. Would like to use it for prediction over classification

Conclusion

If were to choose one model to classify fraudulent vs. non-fraudulent cases, the choice would be PCA Random Forest because of its superior accuracy, moderate calculation time, and likely the lowest tendency to overfit.

Second choice would likely be SKB Logistic Regression, as it has nearly the same accuracy, higher speed, and possibly lower chance of overfitting as well.

Thanks!

The notebook can be found here:
https://github.com/mynameisjc/thinkful/blob/master/Supervised%20Learning%20Capstone%20-%20Paysim%20Fraud%20Detection.ipynb