



# FRAUD DETECTION IN THE FINANCIAL INDUSTRY

THINKFUL SUPERVISED LEARNING CAPSTONE

KEVIN LAM

# BACKGROUND

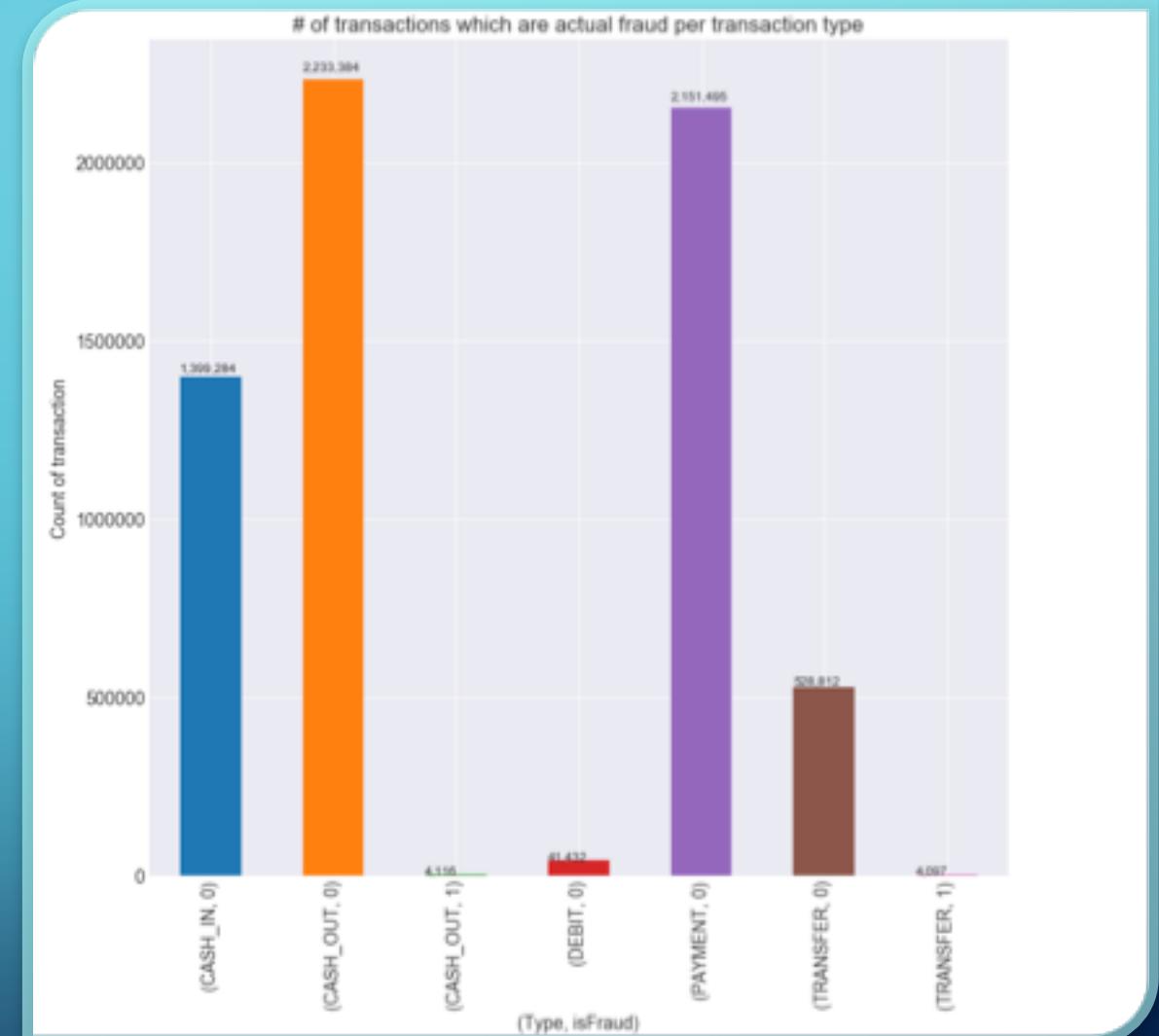
- Found on Kaggle: <https://www.kaggle.com/ntnu-testimon/paysim1>
- Synthetic financial dataset created for fraud detection generated by Paysim.
- Fraud- fraudulent mobile money transactions where an agent attempts to gain access to a customer's account and empty the funds by transferring to another account and cashing it out of the system.

# WHY CARE ABOUT FRAUD DETECTION?

- Mobile money transactions are prevalent in today's society, it might even become the economic standard.
  - Ex: Venmo, Zelle, Apple Pay, Samsung Pay, Google Wallet, Ali Pay, Wechat Pay
  - Making it safer to use and more secure could increase usage and acceptance of these mobile products.
- Creating an accurate classifier will help boost the trust in these products and attract more users.

## WHY CARE PART 2

- We see that no matter how intricate the scam may be, the ultimate goal is to TRANSFER and then CASH OUT.
- If I can create a classifier that can accurately and quickly deduce whether a transaction is fraudulent or not, there can be an implementation where the money is frozen so that the fraud doesn't occur.



# GOAL

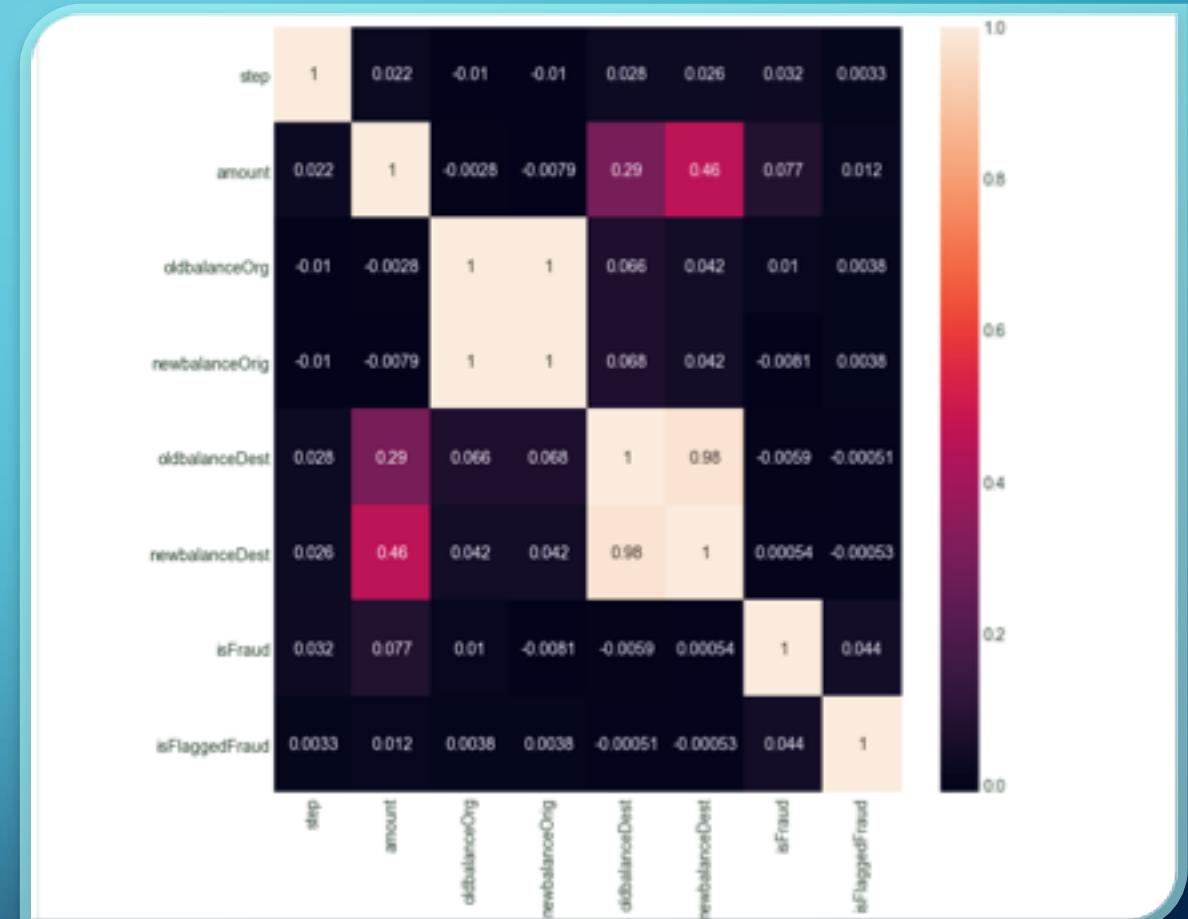
- Two type of models:
  - I want to testify, I need to CLASSIFY
- Find the best model to accurately label data as fraud.
  - BEST – Not only fast, but accurate too.

# DATA EXPLORATION

- 11 features in the dataset:
  - Categorical: Type, nameOrig (customer who started the transaction), nameDest (recipient of the transaction), Fraud, Flagged as Fraud (isFlaggedFraud)
  - Step, Old/ New Balance Owner, amount, Old/New Balance Destination
- What to do with the categorical data?
  - Get dummies for the type column.
  - Drop the names.

# DATA EXPLORATION CONT'D

- Created a heat map to see if any of the variables had any sort of unexpected multicollinearity.
- Dataset had 6,362,620 rows originally.
- There weren't any missing values.
- Suffering from class imbalance:
  - Only 8213 out of the 6,362,620 were fraud.



# CLASS IMBALANCE

- Class Imbalance, so what?
  - Most of the data is non-fraudulent, not balancing the data will skew any model and make it 99% accurate even though it isn't.
  - Resample: 100,000 cases of true and false.

# BUT WAIT... THERE'S MORE

- Feature Selection Method
  - SelectKBest vs Principal Component Analysis
    - SKB removes all but the highest scoring features.
    - Scoring in this case is determined by the `f_classifier` aka using the f-test to determine if there is any statistical significance in the sample variance of the dataset.
    - Principal component analysis transforms the all the features in the dataset and gives entirely new features that are independent from each other.
    - I chose to use 4 Principal Components.
  - \*SPOILER ALERT\*
    - The feature selection method I liked the most was PCA. I'll talk about it as we move through the models.

# PERFORMANCE RUBRIC

- Best Parameters – parameters that allow for the best classifying score
- Average 5 Fold Cross Validation Score – resampling the dataset into 5 equal sized samples and running through the model again.
- Classification Report (Average scores between the fraud and non fraud cases)
  - Precision –  $(TP/(TP+FP))$  Ability of the classifier not to label as positive a sample that is negative.
  - Recall –  $(TP/(TP+FN))$  Ability to find all positive samples.
  - F1 - a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.
- AUC – area under the curve. another method to capture the performance of precision and recall in each model.

# MODEL 1: GAUSSIAN NAÏVE BAYES

- Why Gaussian?
  - Because there were initially more continuous variables before I got dummies for the categorical type data and that's the only way I can incorporate them unlike Bernoulli and Multinomial NB.
- SKB Results:
  - Best Parameter: 3 Features
  - Avg. CV: 77.90% || Avg. Precision: 85% || Avg. Recall: 78%
  - Avg. F1: 77% | | AUC: 15.44% | | Runtime: Fast
- PCA Results:
  - Avg. CV: 83.43% || Avg. Precision: 83% || Avg. Recall: 83%
  - Avg. F1: 83% | | AUC: 37.62% | | Runtime: Fast

## MODEL 2: KNN

- Had to prep this when using SKB by normalizing distance in features.
- SKB Results:
  - Best Parameters: 8 features, 2 neighbors
  - Avg. CV: 99.37% | | Avg. Precision 99% | | Avg. Recall 99%
  - Avg. F1: 99% | | AUC: 0.51% | | Runtime: Fast <3 mins
- PCA Results:
  - Best Parameter: 2 neighbors
  - Avg. CV : 98.84%| | Avg. Precision: 83% | | Avg. Recall 81%
  - Avg. F1: 81%| | AUC: 37.67%| | Runtime: Faster <30 seconds
- Overall: Both models overfitted. I managed to reduce a little bit of overfitting by using PCA, but this model still isn't quite there.

# MODEL 3: DECISION TREE

- SKB Results:
  - Best Parameters: 10 features, max depth of 8
  - Avg. CV : 98.76% || Avg. Precision 99% || Avg. Recall 99%
  - Avg. F1: 99% || AUC: 48.95% || Runtime: Fast
- PCA Results:
  - Best Parameter: max depth of 8
  - Avg. CV : 93.13% || Avg. Precision: 87% || Avg. Recall 84%
  - Avg. F1: 85% || AUC: 41.22% || Runtime: Faster
- Overall: Although SKB looks superior, the possibility of the model overfitting is quite high so I would rather go with the PCA model here.

# MODEL 4: RANDOM FOREST

- SKB Results:
  - Best Parameters: 9 features, 75 trees, max depth of 6
  - Avg. CV : 97.46% | | Avg. Precision: 98% | | Avg. Recall: 98%
  - Avg. F1: 98% | | AUC: 48.25% | | Runtime: Slow, ~2.5 hrs
- PCA Results:
  - Best Parameters: 100 trees, max depth of 8
  - Avg. CV : 93.27% | | Avg. Precision: 89% | | Avg. Recall: 87%
  - Avg. F1: 87% | | AUC: 42.30% | | Runtime: Moderate, 18 minutes
- Overall: Both used maximum depth allotted in GridSearchCV. PCA model needed 25 more trees. PCA model less likely to have overfitted. This is just a bunch of decision trees so results being similar to DTC is no surprise. A con to this model is that it's a black box model where you can't see the calculation/voting process.

# MODEL 5: LOGISTIC REGRESSION

- Used the Ridge Regularization due to the amount of rows. Had to prevent the overfitting of the coefficients.
- SKB Results:
  - Best Parameters: 8 features, C (inverse of regularization strength): 1e-05
  - Avg. CV : 90.73% || Avg. Precision: 91% || Avg. Recall: 91%
  - Avg. F1: 91% || AUC: 43.22% || Runtime: Fast, 2 mins
- PCA Results:
  - Best Parameter: C = 1e-05
  - Avg. CV : 83% || Avg. Precision: 84% || Avg. Recall: 84%
  - Avg. F1: 84% || AUC: 37.85% || Runtime: Faster, 9.2s
- Overall: SKB definitely is the superior model this time. The time difference is negligible. Performs slightly weaker than the other models but definitely the penalty has stopped this model from overfitting.

# MODEL 6: GRADIENT BOOSTED MODEL

- SKB Results:
  - Best Parameters: 12 features, 1000 estimators, max depth of 6
  - Avg. CV : 99.85% | | Avg. Precision: 100% | | Avg. Recall: 100%
  - Avg. F1: 100% | | AUC: 0.12% | | Runtime: LONG, 8 hours
- PCA Results:
  - Best Parameters: 1000 estimators, 0.3 learning rate, max depth of 8
  - Avg. CV : 99.33% | | Avg. Precision: 66% | | Avg. Recall: 65%
  - Avg. F1: 64% | | AUC: 22.84% | | Runtime: Faster, 5 hours
- Overall: GB is a greedy learning model. Used the maximum amount of parameters to achieve the best score. Tried using a learning rate to inhibit overfitting but it wasn't effective. It would be a great model for prediction, but classifying limits its inherent strength.

# WHERE IS SUPPORT VECTOR MACHINE?

- It took too long to run.
- Even tried to use a linear kernel but to no avail.

# CONCLUSION

- In production, if only one model is allowed to be used to classify whether an activity is fraudulent or not:
  - PCA Random Forest is the way to go. Superior accuracy with a moderate calculation time.
  - Runner up: SKB Logistic Regression given its superior run time and moderate accuracy.

# QUESTIONS?

Find the notebook at: <https://github.com/lamka/Thinkful-Unit-3-Capstone/blob/master/Unit%203%20Capstone%20Project.ipynb>