Capstone Final Report – Fraud Detection in Online Transactions

I. Executive Summary

This project uses a dataset of labeled transactions to build a machine learning model that predicts whether an online transaction is legitimate or fraudulent. We use the AUC score to compare our machine learning model to a baseline model and evaluate performance in terms of approval rate, chargebacks, and missed revenue. Our tests found that a Neural Network model with 1 hidden layer performed the best, with a $1,799.816.17 increase in revenue and $227,446.73 decrease in chargebacks compared to the baseline model.

II. Background

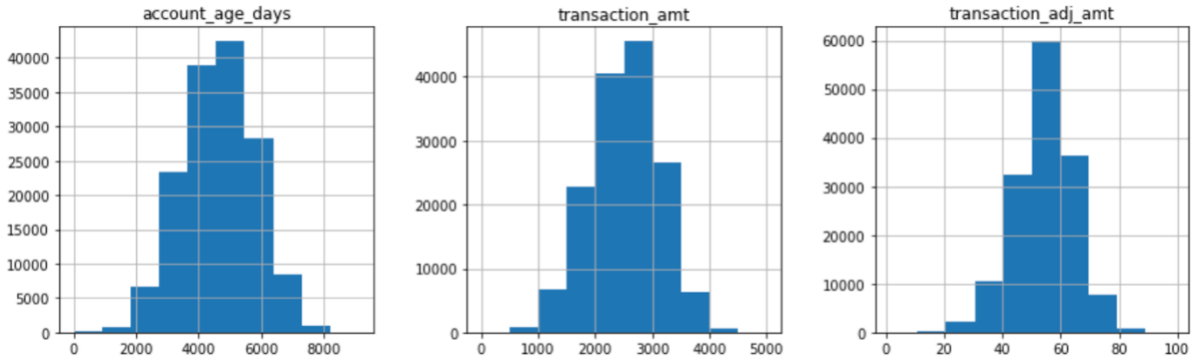
We live in a world where online shopping is becoming so prevalent, and ecommerce businesses are growing by the day. Especially in recent years due to the COVID-19 pandemic, consumers are migrating to remote shopping services to prevent the spread. Unfortunately, this shift in the ecommerce landscape has also opened up many new opportunities for fraud. It is more important than ever for businesses to adopt an efficient fraud prevention strategy to keep their customers safe costs low.

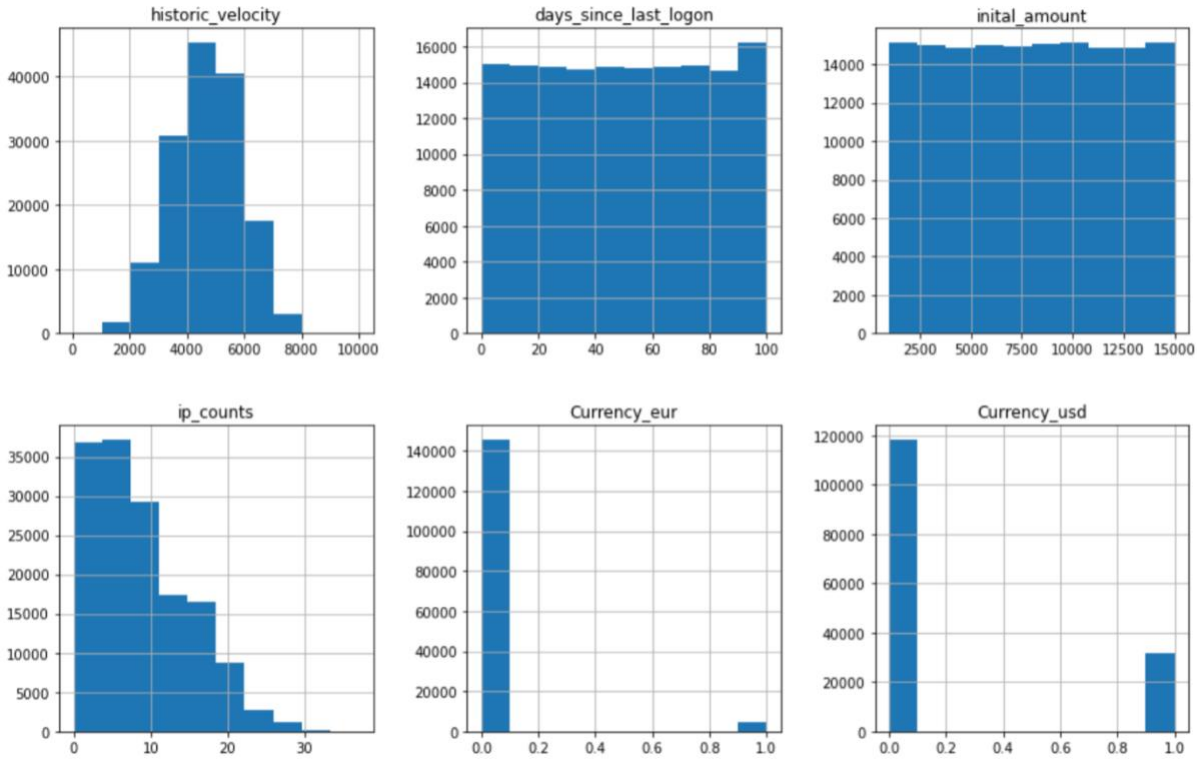
III. Data

The dataset used in this project was taken from Kaggle, which contains the information of 150,000 online transactions from Oct 2020 to Oct 2021 including transaction amount, customer information, and session details.[1] We dropped some customer-specific and unclear columns that are not useful for our model, and interpolated null values with the median based on their distributions. We transformed the IP address column to display the number of times that the IP address has appeared in the past[2], and created dummy variables for the currencies[3].

Below are the histograms for our final variables to be used in the model.

Figure 1 – Distributions of Input Variables

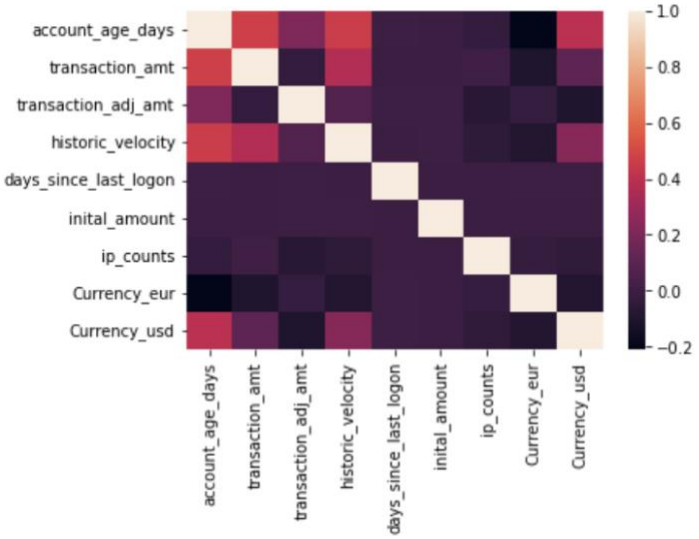




As you can see, account age, transaction amount, adjusted transaction amount, and historic velocity follow a normal distribution; number of days since last logon and initial amount follow a roughly uniform distribution, and counts of IP address follows a Poisson distribution. The dummy variables have binary outputs of either 0 or 1.

We also take a look at the correlations between the input variables.

Figure 2 – Correlation Heatmap of Input Variables



It looks like most of the features are not correlated with one another, meaning they are independent. Account age is slightly correlated with transaction amount and historic velocity, which makes sense because the longer an account has been opened, the more transactions and velocity associated with it.

After performing EDA on our cleaned data, we are ready to create the machine learning model.

IV. Model

In the preprocessing phase, the data is separated into features and targets, and the target variable is mapped to legit=1 and fraud=0. We then split our data into training and testing sets, with 80% in training and 20% in testing. Finally, the input variables are standardized due to their varying ranges.

Several classification models are tested on this set of data to determine the best performing model, including Logistic Regression, Random Forest, Gradient Boosting, XGBoost, and finally several Neural Network models.

Each model is first tuned separately to find their best performing hyperparameters[4], and then the AUC score is calculated as the metric for comparison between models.

V. Results

Out of all the models tested, the Neural Network model with 1 hidden layer resulted in the highest AUC score. Out of the tree-based models, the XGBoost model performed the best with a similar AUC score.

Additionally, we created a baseline model to compare our tuned models to, which consisted of a simplified Random Forest model with a max depth of 2[5].

Table 1 – AUC Scores of Various Classification Models

Model	**AUC Score**
Baseline	0.894
Logistic Regression	0.914
Random Forest	0.925
Gradient Boosting	0.926
XGBoost	0.929
Neural Network	0.930

Finally, we calculated the business impact of the best performing models (XGBoost and Neural Network) compared to the baseline model. We calculated precision and recall for every

threshold – for the baseline model, we set the approval rate to 81.34% (recall of 0.85). This approval rate resulted in a chargeback rate of 1.29%.

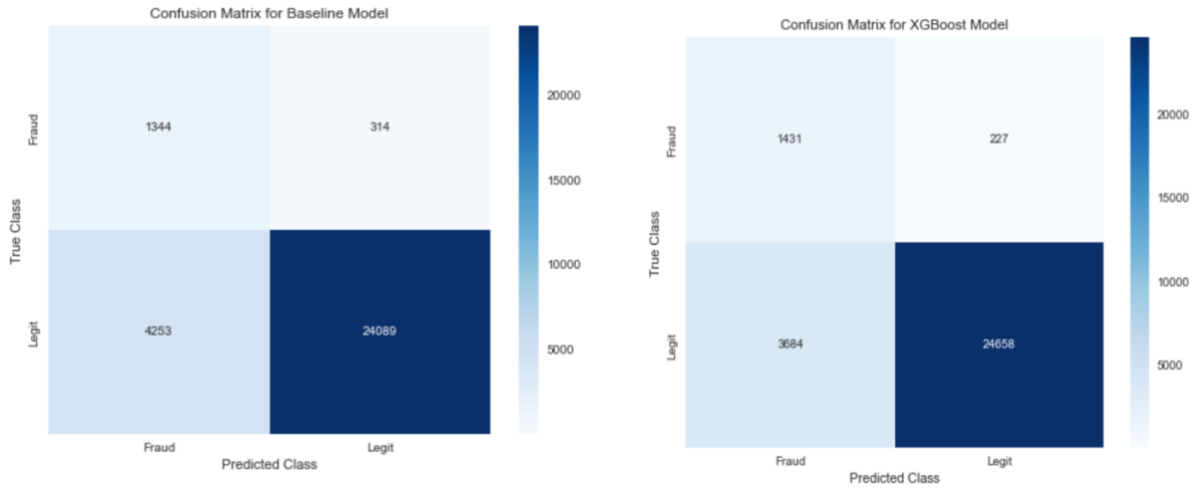
For the XGBoost model, we increased approval rate to 82.95% (recall = 0.87), and the chargeback rate ended up decreasing to 0.912%. In the Neural Network model, we were able to further increase approval rate to 83.41% (recall = 0.875), and still got a slightly lower chargeback rate of 0.899%.

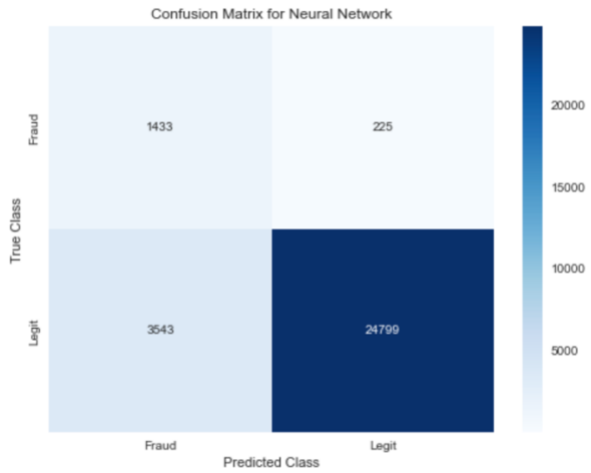
Table 2 – Business Impact of Fraud Detection Models

	Baseline	**XGBoost**	**Neural Network**
Order Volume	$75,959,876.00	$75,959,876.00	$75,959,876.00
Approval Rate	81.34%	82.95%	83.41%
Approved Volume	$61,785,763.14	$63,008,717.14	$63,358,132.57
Chargeback Rate	1.29%	0.912%	0.899%
Chargeback Volume	$797,036.34	$574,639.50	$569,589.61
Revenue	$60,988,726.79	$62,434,077.64	$62,788,542.96
Missed Revenue	$10,771,110.42	$9,327,872.77	$8,970,861.36

We used the confusion matrix of each model to calculate the approval rate, chargeback rate, revenue, and missed revenue.[6]

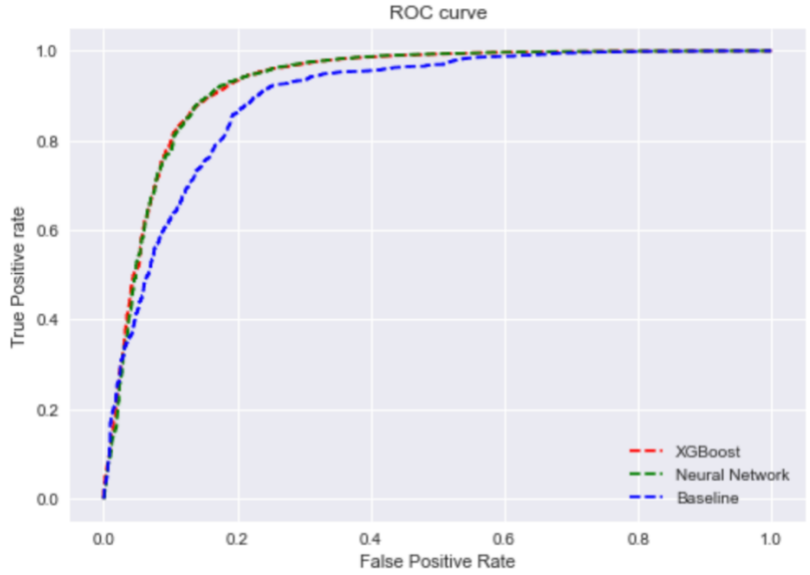
Figure 3 – Confusion Matrices of Best Performing Models and Baseline





Taking a look at the ROC curves of each of the models, we see that XGBoost and Neural Network models have similar performances, and both outperform our baseline model.

Figure 4 – ROC Curves of Best Performing Models Compared to Baseline



VI. Conclusion

After testing several classification models on our data, we found that the Neural Network model with 1 hidden layer performed the best, and was able to both increase the approval rate and decrease the chargeback rate compared to the baseline model. This shows that our input variables relating to the transaction amount and account activity for each transaction do have predictive power in determining whether a transaction is legitimate or fraudulent.

VII. Appendix

1. Dataset from Kaggle:

   Fraud Challenge Data: https://www.kaggle.com/ban7002/fraud-challenge-data

2. We transformed the IP address column into number of times that IP address has shown up in the past with the following pseudocode:

   - sort the data by timestamp
   - add a new column ip_count=1 for every row [number of times that ip address has shown up]
   - add new column ip_counts equal to the cumulative sum of ip_count for each ip address
   - if row doesn't have timestamp, ip_counts=1
   - if row doesn't have ip address, ip_counts=0

3. The following dummy variables were created from the currency column:

   - Currency_usd
   - Currency_eur
   - Currency_cad

   We dropped the "Currency_cad" column to prevent correlation between the dummy variables.

4. Hyperparameter tuning – we tuned the following hyperparameters for each model as follows:

   - Gradient Boosting:
     - 'max_depth': [3, 10, 1000]
     - 'min_samples_leaf': [1, 50, 100]
     - 'n_iter_no_change': [5]
   - Random Forest:
     - 'max_depth': [3, 10, 15]
     - 'min_samples_leaf': [1, 50, 100]
     - 'n_estimators': [100]
   - XGBoost:
     - 'max_depth': [3, 6, 10]
     - 'min_child_weight': [1, 50, 100]
     - 'n_estimators': [100]
   - Neural Network:
     - Input layers: [9, 5]
     - # of hidden layers: [0, 1]

5. Baseline model parameters – for our baseline model, we used a Random Forest model with the following parameters:

    - 'max_depth': [2]
    - 'n_estimators': [1000]

6. Approval / chargeback / revenue metrics are calculated as follows:

    - Order Volume = Sum of all Transaction Amounts
    - Approval Rate = (TP+FP)/Total # of Orders
    - Approved Volume = Approval Rate * Total Order Volume
    - Chargeback Rate = FP/(TP+FP)
    - Chargeback Volume = Chargeback Rate * Approval Volume
    - Revenue = Approval Volume – Chargeback Volume
    - Missed Revenue = (FN/Total # of Orders) * Total Order Volume

Model	**AUC Score**
Baseline	0.894
Logistic Regression	0.914
Random Forest	0.925
Gradient Boosting	0.926
XGBoost	0.929
Neural Network	0.930