Fraudulent Transactions Classifications Using KNN and Decision Tree

***Abstract*— The classification of fraudulent charges is dependent on many different attributes to the transactions. Some of the main ones are the amount, date, and location. With the many different classification models at our disposal, we decided on Decision Tree, KNN, and Bayes to make our predictions for our dataset. With the process of applying all these models we see which might work the best and produce minimal error and the most accuracy. Through various python libraries available to us we were able to build and run models which are seen in this report.**

I. Background

Credit card transactions are now a vital component of our everyday lives and have become the preferred use of payment with the rise of online shopping and digital transactions. According to Statista, the total value of credit card transactions globally in 2019 was approximately 28.7 trillion dollars [1]. But as credit card use has increased, so has the rise of fraudulent transactions.

You might be wondering, what is credit card fraud? Credit card fraud is the use of an individual's information for other purposes. Individuals get this information through the use of phishing, skimming, and data breaches all throughout the internet. According to a report from Javelin Strategy & Research, the total losses from credit card fraud worldwide were around 28 billion U.S. dollars in 2018 [2]. This immense value highlights the need for an approach to prevent fraudulent activities.

Credit card fraud has taken a toll on everyone, affecting not just individuals, but also financial institutions, businesses, and the economy. For individuals, fraudulent transactions can result in financial loss, identity theft, and a damaged credit history.

Thus, developing and deploying an efficient approach in order to avoid these fraudulent transactions requires the use of machine-learning methods that would recognize and identify these fraudulent transactions through individuals' statements. Catching these transactions before they go through is required, as the rate of transactions is projected to grow in the future, which will allow for the rate of fraudulent activity to decrease.

II. Dataset

The dataset we have is a simulated credit card fraud created by the Sparkov Data Generation tool by Brandon Harris [3]. From the Kaggle source we downloaded the dataset from, it says it was generated with the parameters of 1000 people making purchases at around 800 different places. The dates also range from January 1st, 2019 to December 31st, 2023. This set has many different attributes that help us decide whether the transaction is fraudulent or not. The attributes are date, credit card number, merchant, category, amount, first name, last name, gender, street, city, state, zip, lat, long, population, job, and date of birth [4]. Some of the main ones we aim to use are merchant, date, and amount. The data we are using here was formatted into a csv file for use. It was also split into a train and test set for us. Finally, we choose this dataset because of its many different attributes for use in our model

III. Methods and Measures Used

Our project utilized the Python language to implement a decision tree classification algorithm. For our fraud detection model, it was the perfect environment for it. Through the vast world of Python's libraries, using Pandas for data manipulation, Sklearn to build the model, and LabelEncoder to turn the variables into a machine-readable format, all three combined were the perfect approach for a decision tree algorithm. The efficiency of our model was assessed through multiple vital components. These metrics offered a concise assessment of the model's performance in detecting fraudulent transactions.

We aim to test a range of models from linear regression, decision trees, and neural networks. Methods such as linear regression and decision trees are more interpretable, but simultaneously less robust than neural networks. With regards to neural networks, there are several approaches to be had, the first is a simple fully connected multi-layer perceptron which has the added benefit of being able to learn non-parametric functions. A secondary method, which poses a more interesting implementation focuses on the idea that fraud is likely clustered, in that someone has to commit the fraud and where fraud is committed there is likely more fraud. The latitude and longitude for the location of the person and the merchant may be translated into distance vectors or even directly mapped to a grid. This implicit 2-dimensional structure means that we may implement structures such as LSTM or possibly even CNN methods to flag fraud more accurately. Further, given the nature of distance metrics included in the data, we must consider k-nearest Neighbors or other kernel methods as intuitive options. Given the wide breadth of models and subsequently the large amounts of hyperparameter selection for such models, we aim to implement the simple perceptron, possibly extending to the subsequent LSTM, CNN, and kernel method models assuming sufficient implementation is developed for our use case.

IV. Results and Interpretations

A. Decision Tree Classifier

After training the model and running on the transactions data set, it produced the following results:

- Accuracy: 0.9982509145809303
- Precision: 0.8217224355458036

- Recall: 0.6983682983682984
- F1 Score: 0.7550403225806451
- Mean Absolute Error: 0.00174908541906
- Mean Squarerd Error: 0.00174908541906
- Root Mean Error: 0.04182206856516915
- Mean Absolute Percentage Error: 2633830908958
- Confusion Matrix:

  [[553249, 325]

  [647, 1498]

When it comes to the accuracy of our decision tree model, it showcased an outstanding accuracy of 99.83%, indicating how highly effective the model was in classifying the transactions in their corresponding variables. Although, we cannot only rely on the accuracy percentage as the sole indicator of performance. Thus, with a precision rate of 82.17%, the model correctly predicted fraudulent transactions more than half of the time. On the same note, the recall rate of 69.83% shows the model's percentage in determining the majority of fraudulent transactions. This is vital in a fraud detection model as not being able to detect actual fraudulent transactions can be more harmful than labeling legitimate transactions as fraudulent. By combining both the recall and the precision rate, we get an F1 score of 75.50%, which shows a good balance of the model in correctly determining fraudulent transactions, from classifying legitimate ones as fraud. Finally, we get to the confusion matrix.

The matrix has four major parts, the true and false positives and negatives (TN, FN, TP, FP):

- True Negatives (TN): 553249 legitimate transactions were correctly classified
- False Positives (FP): 325 legitimate transactions were incorrectly labeled as fraudulent
- False Negatives (FN): 647 fraudulent transactions were not detected
- True Positives (TP): 1498 fraudulent transactions were correctly detected

The confusion matrix is vital in our model as it is evidently informative in showing the model's strengths and weaknesses.

B. Other Models

Comparing the previous results with the approach of linear regression, k-NN, and Naive Bayes we get the following:

Linear Regression:

- Mean Absolute Error: 0.009958284012577116
- Mean Squared Error: 0.0038499234134898807
- Root Mean Squared Error: 0.06204775107519918
- R2 Score: -0.0012893538391397942

K Nearest Neighbors Classifier

- Accuracy: 0.9954599356869209
- Precision: 0.029850746268656716
- Recall: 0.005594405594405594
- F1 Score: 0.009422850412249707
- Mean Absolute Error: 0.004540064313079092
- Mean Squared Error: 0.004540064313079092
- Root Mean Squared Error: 0.0673799935499474
- Confusion Matrix:

  [[553184, 390]

  [2133, 12]]

Naïve Bayes

- Accuracy: 0.9961401355721147
- Mean Absolute Error: 0.0038598644278853163
- Mean Squared Error: 0.0038598644278853163
- Root Mean Squared: 0.062127807203258965
- Mean absolute Percentage Error: 0.00385986
- Confusion Matrix:

  [[553574, 0]

  [2145, 0]

Comparing the results of the decision tree model to the three other models, we found that in all of the metrics, the decision tree was far more accurate. As shown throughout the results above, all models have a high accuracy rate, although it cannot be the sole indicator of a model's performance. When it came to the decision tree model, the accuracy, precision and recall were much higher than any of the other models, thus indicating the positive overall performance of the model. The biggest indicator that proves how well the DT model performed in comparison to the other models is the matrix. As shown above, the DT model shows a good balance of false positives and false negatives, incorrectly classifying (325) legitimate transactions and (647) fraudulent transactions. On the other hand, k-NN had a higher number of false negatives (2133), which is particularly concerning in fraud detection because it means more fraudulent transactions are going undetected. Naive Bayes, despite having no false positives, has a very high number of false negatives (2145), indicating it's very conservative and likely to miss actual frauds.

V. Difficulties With Modeling and Key Takeaways

The most important difficulty we encountered during this project was within the data set. Simply, there were many more instances of non-fraudulent charges than there were fraudulent

charges, which makes sense fundamentally as we do not expect a large number of fraudulent charges — our test set only had 0.3% of observations as fraudulent. The implication of this was that it became extremely easy for most models to not predict any instance as fraudulent. For the multi-layer perceptron, we see in the confusion matrix every data point in the test set was predicted as non-fraudulent.

With respect to the K-Nearest Neighbors model we acknowledge that due to the extremely low proportion of fraudulent transactions any remotely large k would likely only predict non-fraudulent, so we opted to use K = 1 and K = 3. The implication of using K = 1 is that we are saying that if the single nearest neighbor a fraudulent transaction, then we predict the same, this then means that unlike the Multi-Layer Perceptron we are guaranteed to predict some transactions as fraudulent. In the confusion matrix we see that in large this condition does not work as the model only predicted 12 of the 2145 fraudulent transactions, it is also important to note that this model did have better precision and recall than the MLP.

Further for K = 3 we get the same number of False Positives and True Negatives, but more False Negatives, which just means that as we increase K we are adversely affected by the fact that there are so few fraudulent transactions. Finally, for linear regression under the premise that at high dimensionality we are able to linearly separate the data with some fitted hyperplane, it is easy to see that for our data set with widely distributed occurrences of fraudulent transactions (as seen with the results for the) and an extremely low proportion of fraudulent transactions that it is very unlikely that even at high dimensionality finding a hyperplane to separate the data is unlikely. The secondary implication of having such a large data set — with the training set having 1.3 million rows and the testing set having 555 thousand rows — is that the runtime for most of the models is long enough to where it makes running model selection on the hyperparameters very difficult as we have to balance our own local computing resources with our goal of finding the best implementations for each model. Still, we ran a handful of tests for each model, but we were not able to run a thorough model selection method.

Contributions of the Authors

The group members added the following contributions to the classification of fraudulent transactions. Ahmed Aldhaheri completed the background research of our topic, and all model explanations section of the final report. Ahmad also completed all slides in the presentation that corresponded to his sections completed of the report. Devin Thakker did the initial clean-up of the data set in preparation for the models. Devin also wrote the python code with sklearn for the Decision Tree, KNN, Linear Regression, and Naïve Bayes. Devin also completed the corresponding slides on the presentation along with formatting the report in the IEEE format. Yash Gollapudi completed the background research on the methods we implemented and finished the Methods, Difficulties, and Takeaways section of the report. Yash also completed the corresponding slides.

References

[1] Statista. (2021). Value of credit card transactions worldwide from 2017 to 2019 (in trillion U.S. dollars). Retrieved from: https://www.statista.com/statistics/1112599/global-credit-card-transaction-value/

[2] Javelin Strategy & Research. (2019). 2019 Identity Fraud Study: Fraudsters Seek New Targets and Victims Bear the Brunt. Retrieved from: https://www.javelinstrategy.com/coverage-area/2019-identity-fraud-study-fraudsters-seek-new-targets-and-victims-bear-brunt

[3] Github. (2022). Sparkov Data Generation Repository. Retrieved from: https://github.com/namebrandon/Sparkov_Data_Generation

[4] Kaggle. (2020). Credit Card Transactions Fraud Detection Dataset (Simulated Credit Card Transactions generated using Sparkov) Retrieved from: https://www.kaggle.com/datasets/kartik2112/fraud-detection/data