CS471: Introduction to Artificial Intelligence Project Progress Report

Rogelio Linares Rodriguez, Kento Uematsu, Adolfo Sanpedro Gante

December 6, 2024

Motivation:

We are trying to solve the problem of fraudulent transactions by using prediction on the mode in order to determine if the recipient is valid or not as a way to predict fraudulent transactions. Our group's plan is to prevent fraudulent activity from bad actors by developing a model to simulate a scenario regarding the danger of suspicious behavior with fraud agents.

Prior Research:

In another work, we found a github repository that handled a similar dataset that worked with fraud detection dataset that isn't the exact dataset, however it is still relevant to the topic of discussion because they had the same goal of determining whether a transaction is fraud or not, investigate fraud behavior, find key factors of fraud, make fraud detection models in order to crack down on fraud activity. They used gradient boosting, neural networks, decision trees, and random forest for their model.

Source:

GitHub - kteppris/fraud-detection-financial-transactions:
https://www.kaggle.com/datasets/goyaladi/fraud-detection-dataset

Goals:

For our goals, we constructed some goals that contain a different vision compared to prior research that aligned with our motivation of spreading awareness of fraudulent transactions regarding scenarios where bad activity could develop and attempt illegal activity with the owner's balance. Also, we want to heavily emphasize the importance of more efficient and respectable learning models that institutions can employ for their business and customers as rule-based systems are considered vulnerable to such attacks that demand for more relevant models to protect against unethical behavior. We also wanted to apply rule-based and two other algorithms in order to determine which algorithm is more effective at finding fraudulent transactions. We want to demonstrate that we can detect fraudulent transactions without the need for neural networks or gradient boosting as this problem is considered a binary classification problem whether it is fraud or not.

Dataset:

The dataset includes 11 features:

- Temporal information (step)
- Transaction details (type, amount)
- Origin account information (nameOrig, oldbalanceOrg, newbalanceOrig)
- Destination account information (nameDest, oldbalanceDest, newbalanceDest)
- Target variables (isFraud, isFlaggedFraud)

The data of fraudulent transactions contain 11 unique features including: step which maps a unit of time in the real world, type which represents CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER, amount represents amount of the transaction in local currency, nameOrig represents customer who initialized the transaction, oldbalanceOrg represents initial balance pre-transaction, newbalanceOrig represents new balance post-transaction, nameDest represents customer that identifies as the recipient of the transaction, oldbalanceDest represents initial balance recipient before the transaction, newbalanceDest represents new balance recipient after the transaction, isFraud represents transaction conducted by fraudulent agents within the simulation, and isFlaggedFraud represents the business model aims to monitor massive transfers from one account to another and flags illegal attempts. Moreover, an illegal attempt means an attempt of transfer regarding transactions greater than 200.000 dollars.
Source:
https://www.kaggle.com/datasets/vardhansiramdasu/fraudulent-transactions-prediction/data

Exploratory data analysis:

We performed analysis using built-in functions in order to understand the data statistically & visually for us to interpret how to model our algorithms and develop a sophisticated solution.

```
import matplotlib.pyplot as plt
import pandas as pd
training =
pd.read_csv('/kaggle/input/fraudulent-transactions-prediction/Fraud.csv')
```

training.head()

The head() function outputs the first few rows of the dataset which was 5 rows of data that detailed the features like type and amount with its respective data of a given row.

training.tail()

The tail() function outputs the last few rows of the dataset which was 5 rows of data that detailed the features like nameDest and oldbalanceDest with its respective data of a given row.

training.describe()

The describe() function outputs summary statistics regarding count, mean, std, min, 25% of data, 50% of data , 75% of data, and max with its respective features.

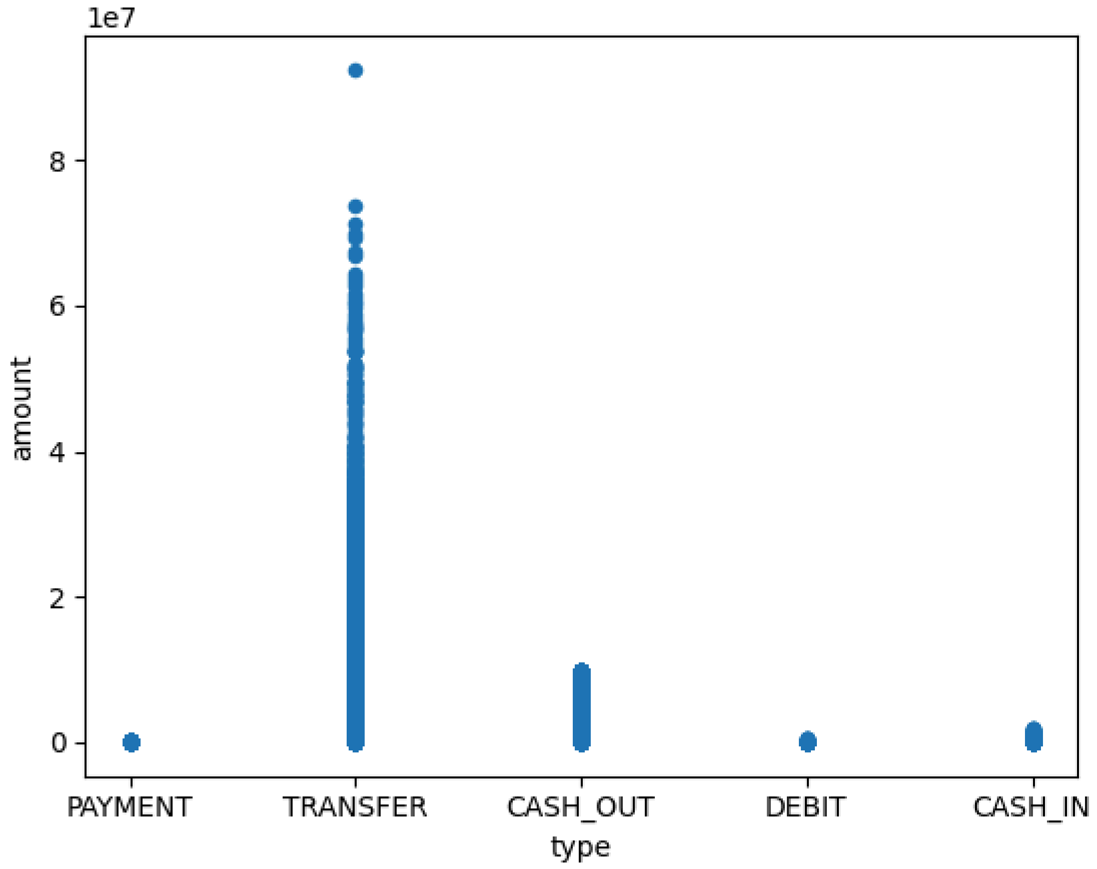
training.isnull().sum()

This function checks for missing values in the entire dataset and all columns have a sum of 0 meaning there are no missing values in this entire dataset of fraudulent transactions. This is good because we don't need to deal with missing data which can cause issues with analysis & modeling of the model.

training.dtypes

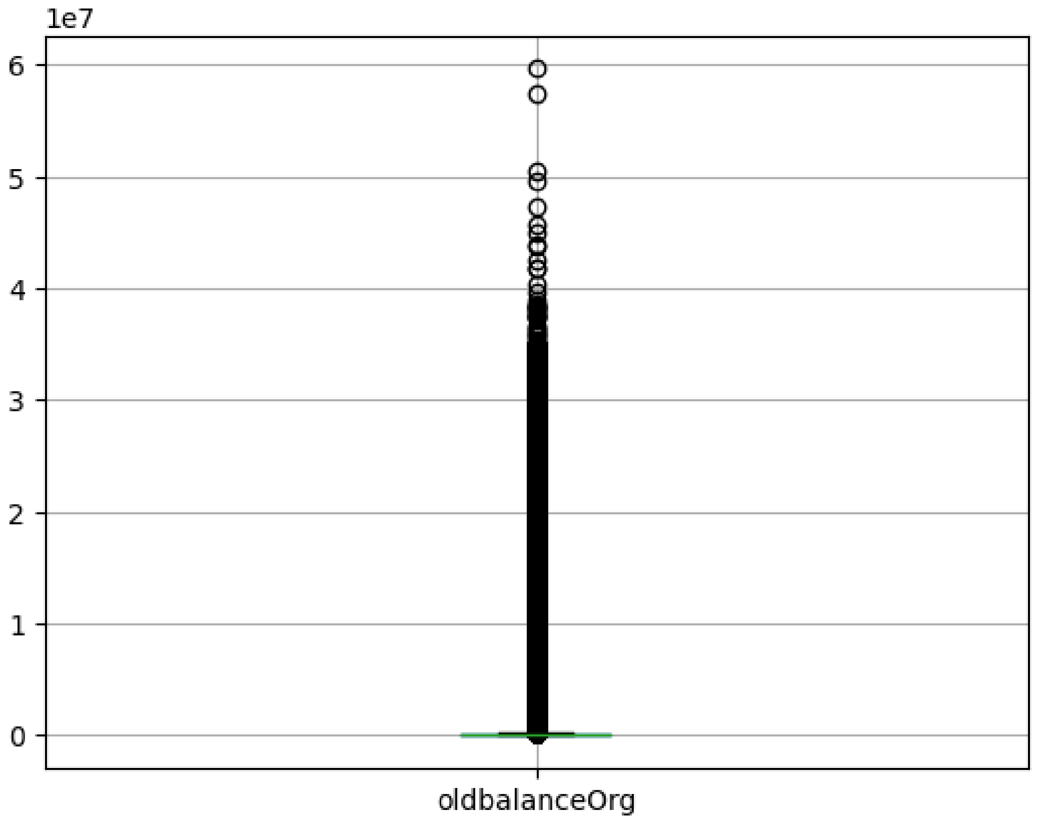
This function gets the data types of each column and the output displays a diverse list of data types like int64 for step, object for type, float64 representing amount for the features and dtype itself is an object overall.

```
training.plot(kind='scatter', x='type', y='amount')
plt.show()
```



A scatter plot shows a relationship between type & amount that visualizes the type of transaction process was conducted as the x-axis and amount of money used in order to make that type of transaction as the y-axis. Furthermore, there is one outlier detected in the transfer type.

training.boxplot(column='oldbalanceOrg')



This boxplot helps us visualize one of the features like oldbalanceOrg in order to understand its distribution. Also, there are two outliers in the oldbalanceOrg feature.

Pre-processing:

For data cleaning which involves finding and fixing inaccuracies and inconsistencies in the data, we used **training.isnull().sum()** in order to determine if there were any missing values in the entire dataset and the output showed a sum of 0 meaning our dataset is free of missing values. In addition, we used **training.drop_duplicates(inplace=True)** to find and remove any duplicate rows in the dataset.

For feature engineering which involves making new features based off existing data in order to improve performance, we used the original features and created new features like:
training['balanceOrigDiff'] = training['oldbalanceOrg'] - training['newbalanceOrig']

This transaction difference feature focuses on balanceOrigDiffer representing the difference between old and new balance of the original account.

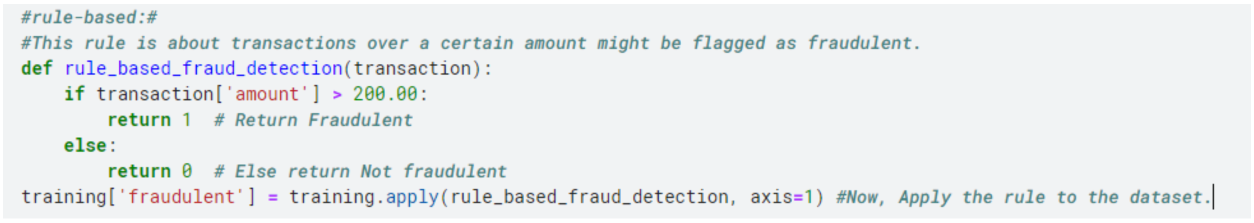
training['balanceDestDiff'] = training['oldbalanceDest'] - training['newbalanceDest']

This transaction difference feature focuses on balanceDestDiff representing the difference between old and new balance of the destination account.

One-Hot Encoding: training = pd.get_dummies(training, columns=['type'])

We can also use one-hot encoding to create one-hot encoded columns for the type column because type column is a categorical variable meaning it contains discrete categories like CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER. This is important because one-hot encoding is designed to handle categorical data by making binary columns for each category that allows ML algorithms to understand these categories without any confusion since the encoding makes sure there is no ordinal relationship assumed between categories.

Algorithm 1 (rule-based approach):

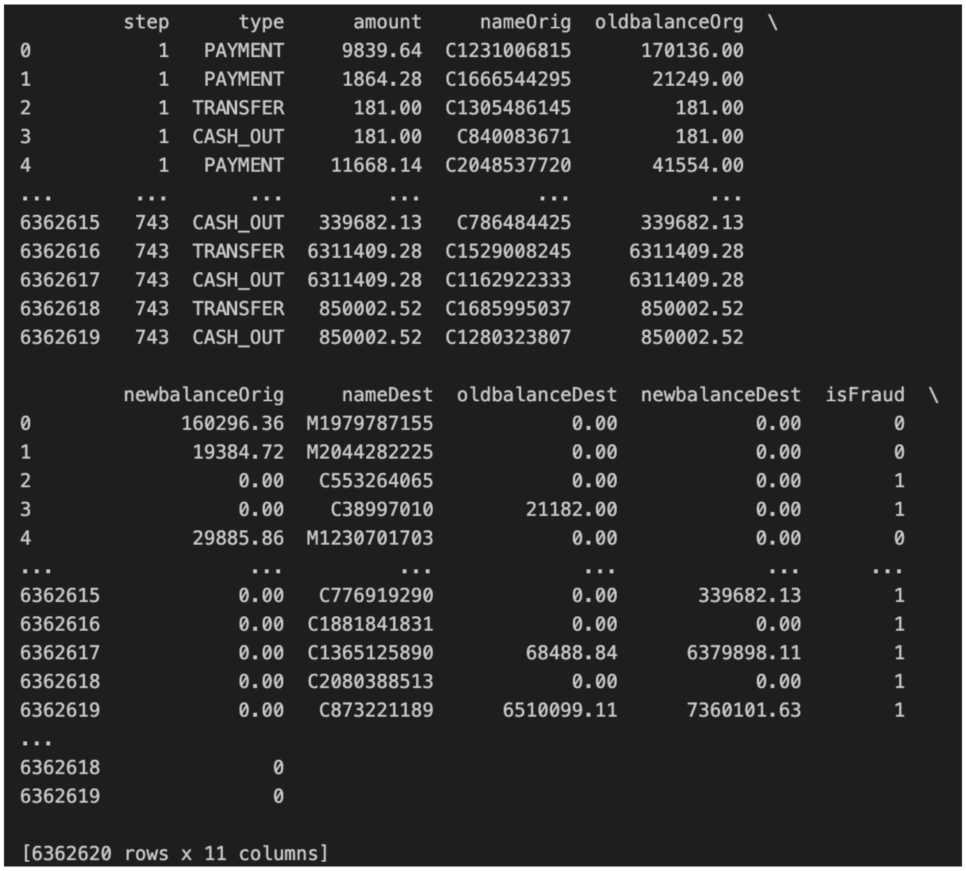


```python
#rule-based:#
#This rule is about transactions over a certain amount might be flagged as fraudulent.
def rule_based_fraud_detection(transaction):
    if transaction['amount'] > 200.00:
        return 1  # Return Fraudulent
    else:
        return 0  # Else return Not fraudulent
training['fraudulent'] = training.apply(rule_based_fraud_detection, axis=1) #Now, Apply the rule to the dataset.
```

This rule-based approach function focuses on transactions over a certain amount that might be flagged as fraudulent if the amount is greater than $200.

The benefits of this rule-based approach are simplicity, transparency, and doesn’t need a large dataset to train the model because our group can understand and code up the rules similar to this rule and the decision process is clear. However, there are negatives regarding this rule that are scalability, flexibility, and performance because the dataset could be complex and complicated if we add more rules similar to this, this type of rule is considered not adaptive to new patterns or adjustments in data, and may not be accurate.

Code link:
https://www.kaggle.com/code/rogeliolinares/cs471-final-project-team-5-progress-report



```
        step      type      amount     nameOrig  oldbalanceOrg  \
0          1   PAYMENT     9839.64  C1231006815      170136.00
1          1   PAYMENT     1864.28  C1666544295       21249.00
2          1  TRANSFER      181.00  C1305486145         181.00
3          1  CASH_OUT      181.00   C840083671         181.00
4          1   PAYMENT    11668.14  C2048537720       41554.00
...      ...       ...         ...          ...            ...
6362615  743  CASH_OUT   339682.13   C786484425      339682.13
6362616  743  TRANSFER  6311409.28  C1529008245     6311409.28
6362617  743  CASH_OUT  6311409.28  C1162922333     6311409.28
6362618  743  TRANSFER   850002.52  C1685995037      850002.52
6362619  743  CASH_OUT   850002.52  C1280323807      850002.52

        newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest  isFraud  \
0            160296.36  M1979787155            0.00            0.00        0
1             19384.72  M2044282225            0.00            0.00        0
2                 0.00   C553264065            0.00            0.00        1
3                 0.00    C38997010        21182.00            0.00        1
4             29885.86  M1230701703            0.00            0.00        0
...                ...          ...             ...             ...      ...
6362615           0.00   C776919290            0.00       339682.13        1
6362616           0.00  C1881841831            0.00            0.00        1
6362617           0.00  C1365125890        68488.84      6379898.11        1
6362618           0.00  C2080388513            0.00            0.00        1
6362619           0.00   C873221189      6510099.11      7360101.63        1
...
6362618              0
6362619              0

[6362620 rows x 11 columns]
```

Algorithms 2 and 3 (machine learning approaches):

Algorithms 2:

We first started with a simple decision tree algorithm. Considering that we are categorizing the data as either fraud or not fraud, a decision tree seems like a good starting point due to its explainability and interpretability.

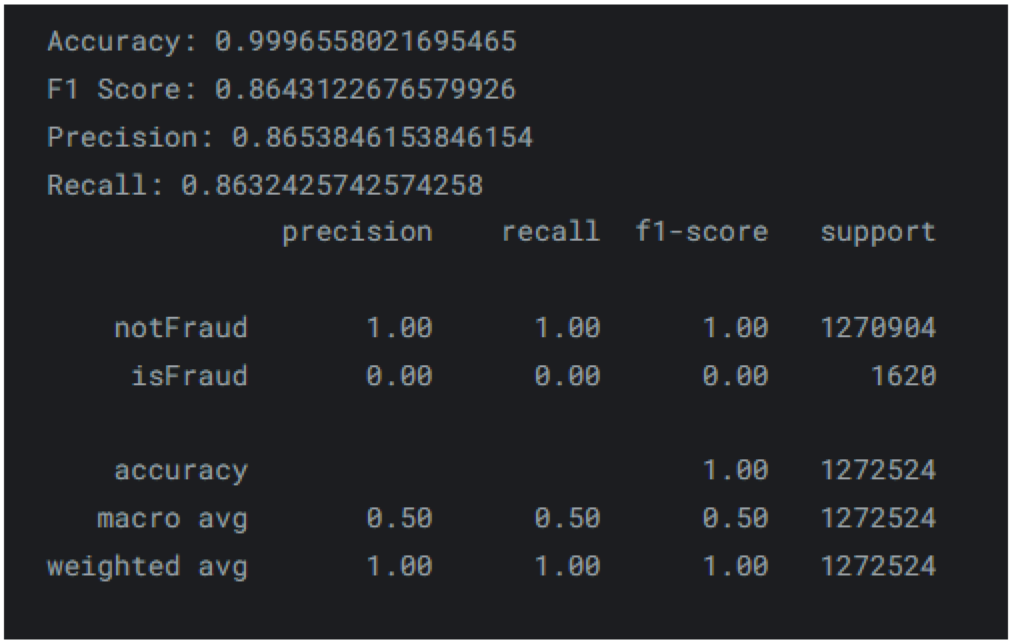
Experiments:

For the decision tree we decided to use label encoding to avoid dropping any features like nameDest and nameOrig. The first couple of confusion matrices drop these features and use one-hot encoding the second set of confusion matrices use label encoding and do not drop any features.

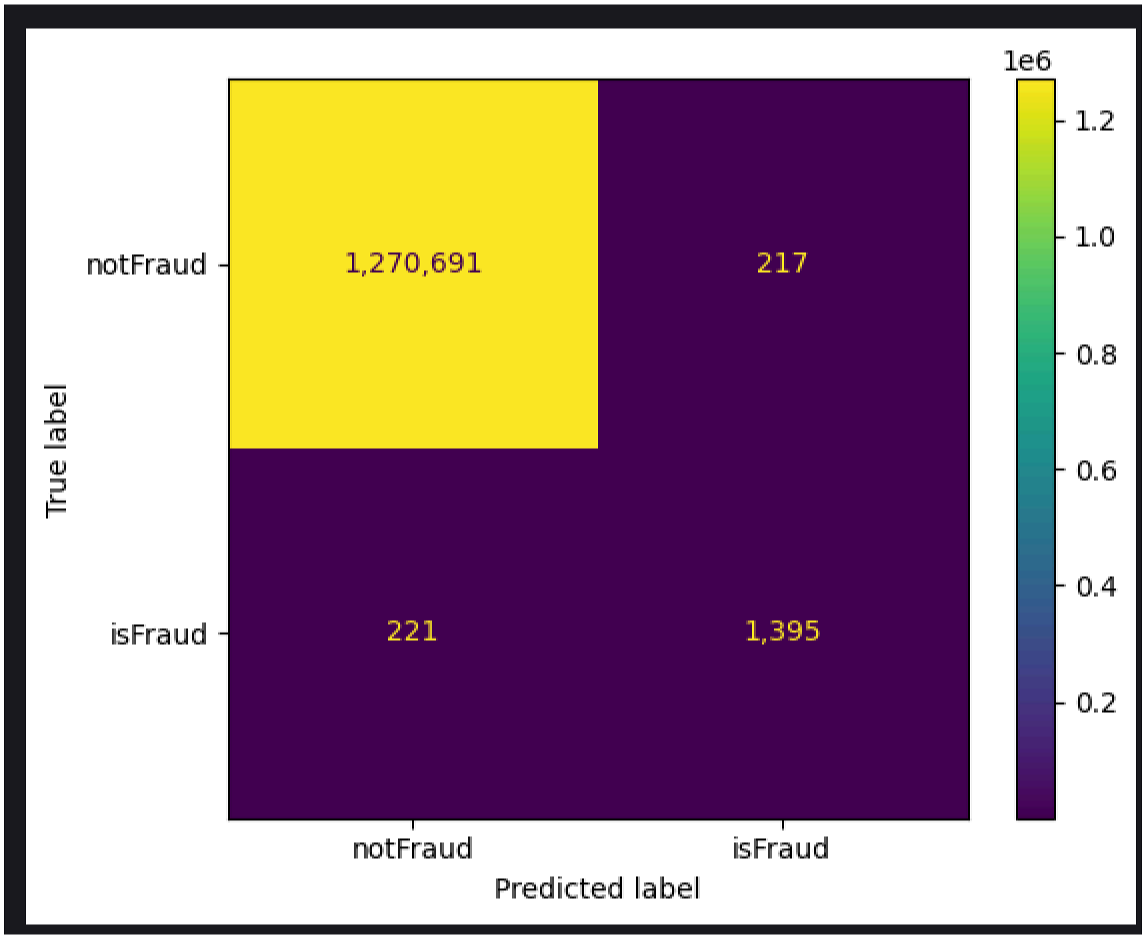
One-hot encoding w/ feature drop

Code Link:
https://www.kaggle.com/code/adolfosanpedrogante/final-project-decision-tree

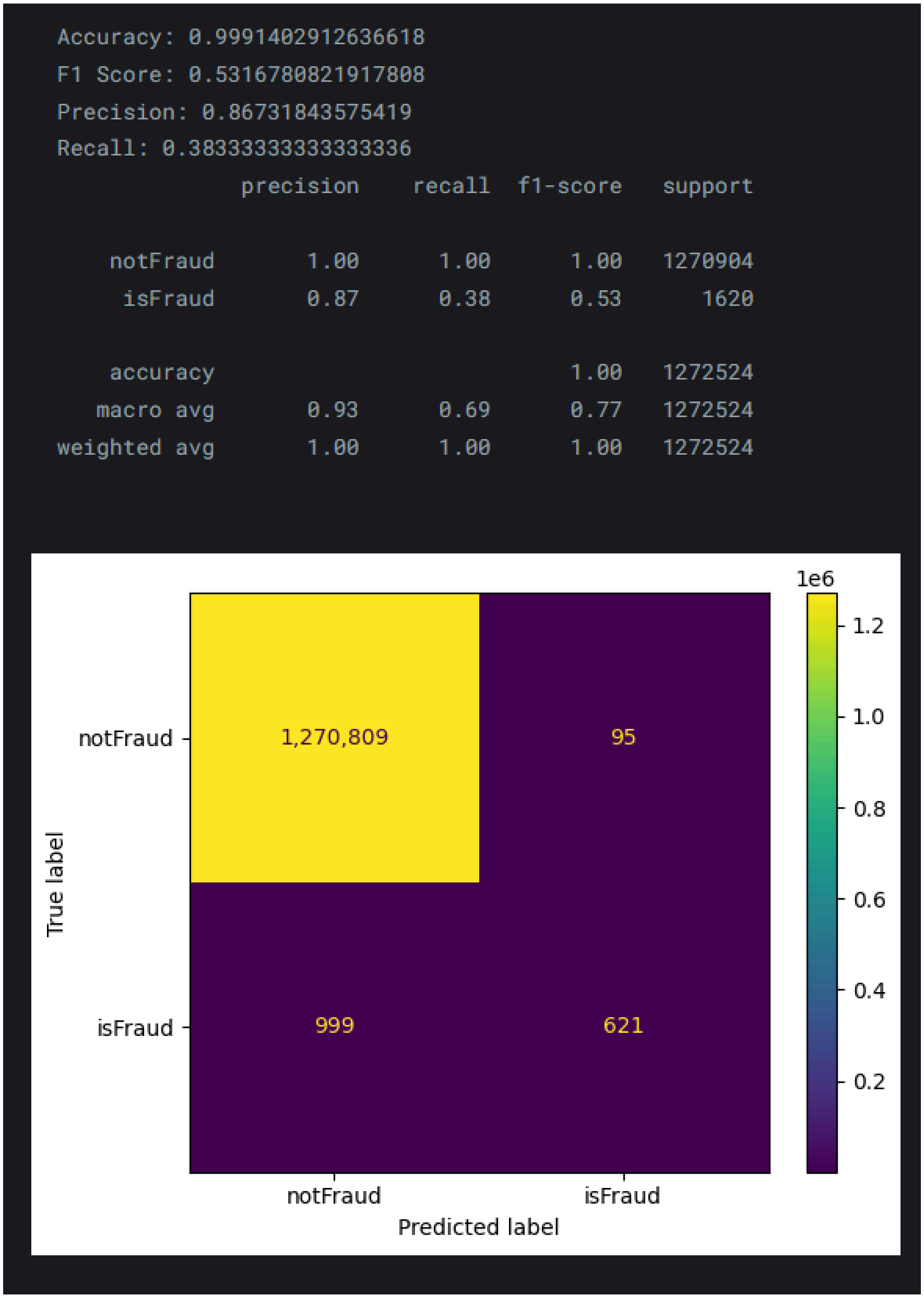
Training:



```
Accuracy: 0.9996558021695465
F1 Score: 0.8643122676579926
Precision: 0.8653846153846154
Recall: 0.8632425742574258
              precision    recall  f1-score   support

    notFraud       1.00      1.00      1.00   1270904
     isFraud       0.00      0.00      0.00      1620

    accuracy                           1.00   1272524
   macro avg       0.50      0.50      0.50   1272524
weighted avg       1.00      1.00      1.00   1272524
```


1e6
1.2
1.0
0.8
0.6
0.4
0.2
True label
notFraud
isFraud
1,270,691
217
221
1,395
notFraud
isFraud
Predicted label

Testing:

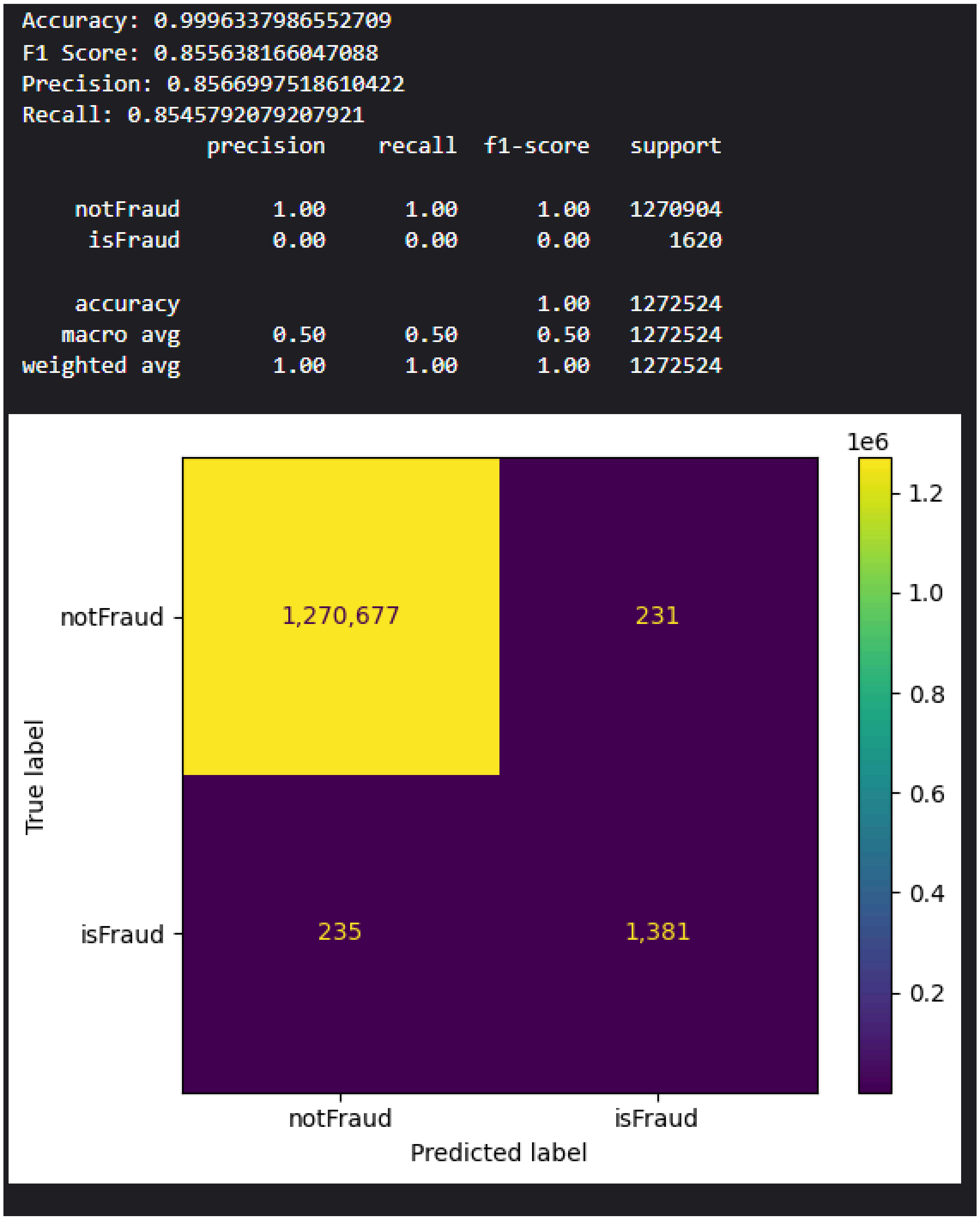


```
Accuracy: 0.9991402912636618
F1 Score: 0.5316780821917808
Precision: 0.86731843575419
Recall: 0.38333333333333336
              precision    recall  f1-score   support

    notFraud       1.00      1.00      1.00   1270904
     isFraud       0.87      0.38      0.53      1620

    accuracy                           1.00   1272524
   macro avg       0.93      0.69      0.77   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Code Link:

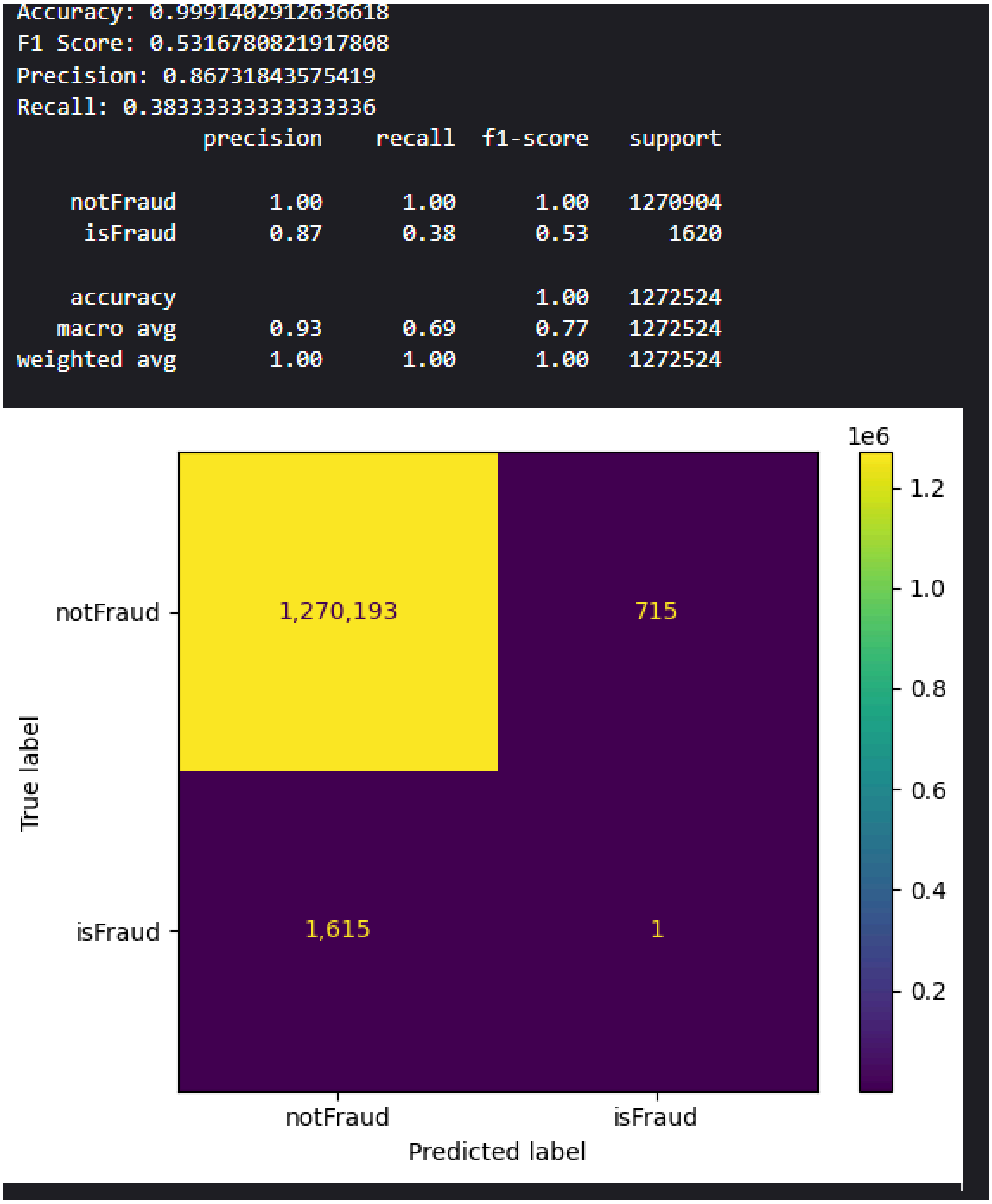
https://www.kaggle.com/code/adolfosanpedrogante/final-project-decision-tree-label-encoder

Training:



```
Accuracy: 0.9996337986552709
F1 Score: 0.855638166047088
Precision: 0.8566997518610422
Recall: 0.8545792079207921
              precision    recall  f1-score   support

    notFraud       1.00      1.00      1.00   1270904
     isFraud       0.00      0.00      0.00      1620

    accuracy                           1.00   1272524
   macro avg       0.50      0.50      0.50   1272524
weighted avg       1.00      1.00      1.00   1272524
```

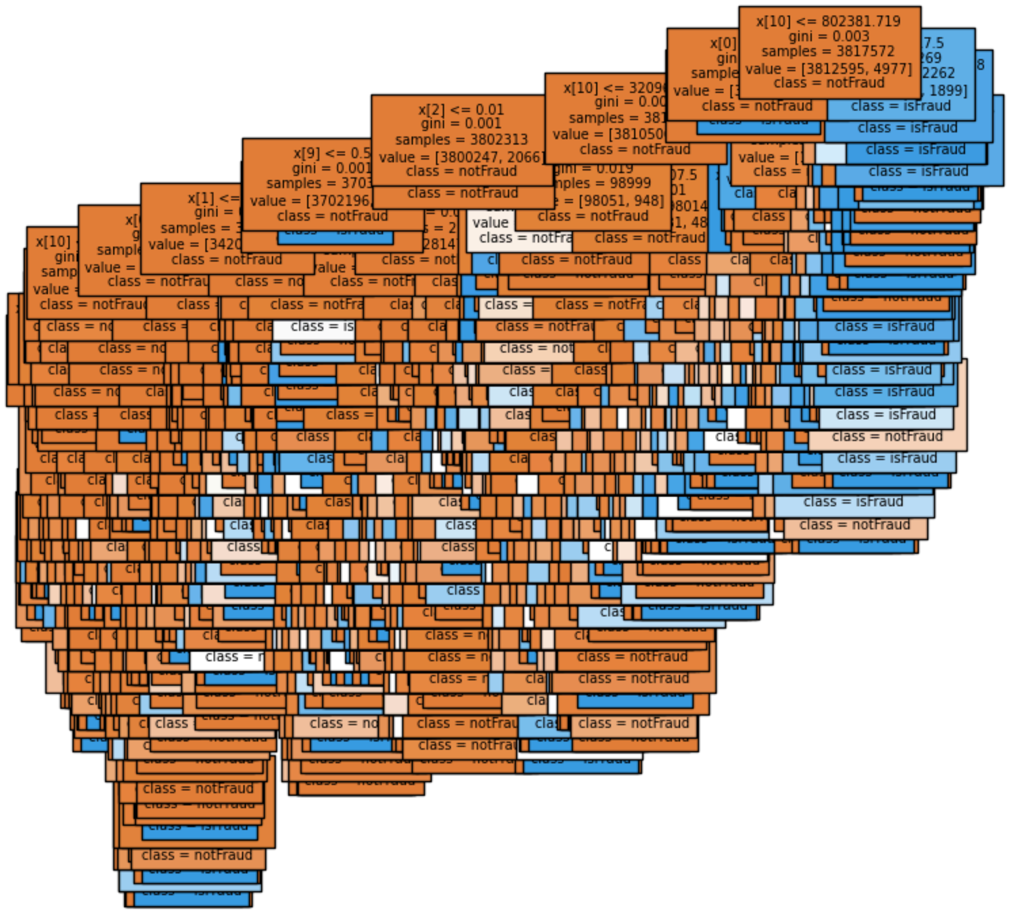
Testing:



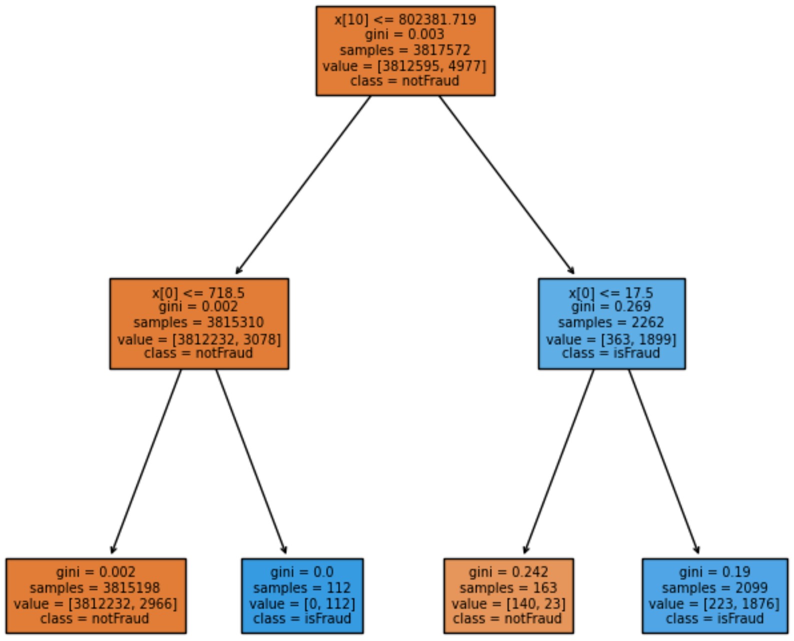
```
Accuracy: 0.9991402912636618
F1 Score: 0.5316780821917808
Precision: 0.86731843575419
Recall: 0.38333333333333336
              precision    recall  f1-score   support

    notFraud       1.00      1.00      1.00   1270904
     isFraud       0.87      0.38      0.53      1620

    accuracy                           1.00   1272524
   macro avg       0.93      0.69      0.77   1272524
weighted avg       1.00      1.00      1.00   1272524
```

Both models used the same tuned hyperparameters which pruned the decision tree.

Before



After



Results Analysis:

You can see that the recall score and precision score do not change from the label encoding w/ no feature drops vs. the one-hot encoding w/feature drops. Precision is too high and recall is too low. The decision model is not successful in predicting fraud transactions. We want recall to be as high as possible because we are trying to catch ALL FRAUDULENT transactions. This is why we decided to use random forest with one-hot encoding and dropping the features nameDest and nameOrig.

Algorithms 3:

We used a random forest classifier in order to demonstrate and analyze the dataset's prediction and accuracy results. Also, we used this approach because it will aim to reduce overfitting by averaging multiple trees which will provide a better measurement of performance & accuracy.

This denotes the fact that it can handle feature importance, however, it does contain drawbacks including: it needs lots of computational resources and hyperparameter tuning such as number of trees and max depth for optimal performance of the model.

Experiments:

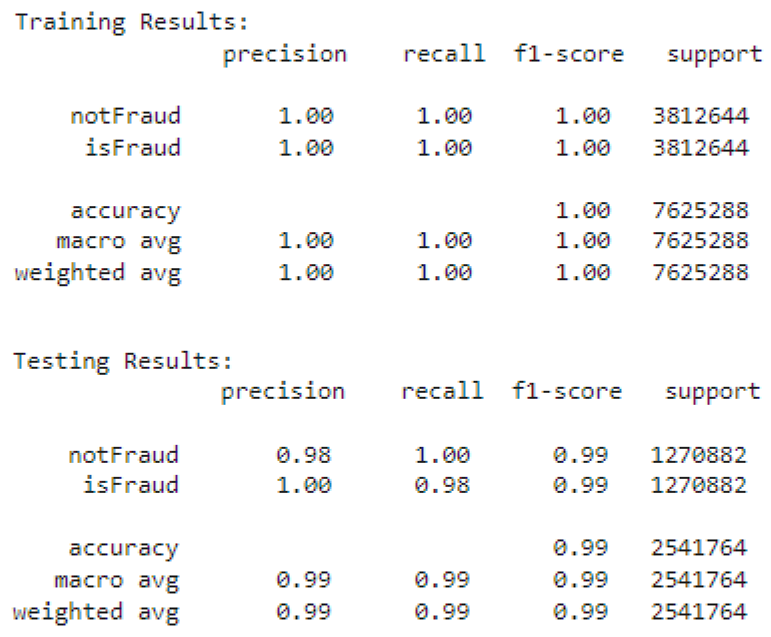
For random forest, we tried to add more features to the model in order to improve the performance of fraudulent transaction prediction, however we realized that adding too many features that didn't reflect the most important features didn't impact the dataset overall meaning that experiment wasn't the best option as we reduced the amount of features after receiving feedback.

Model Comparison:

The random forest performed better than the decision tree from the following result analysis you can see that recall was vastly improved which is important because we want to catch ALL fraudulent transactions.

Result analysis:

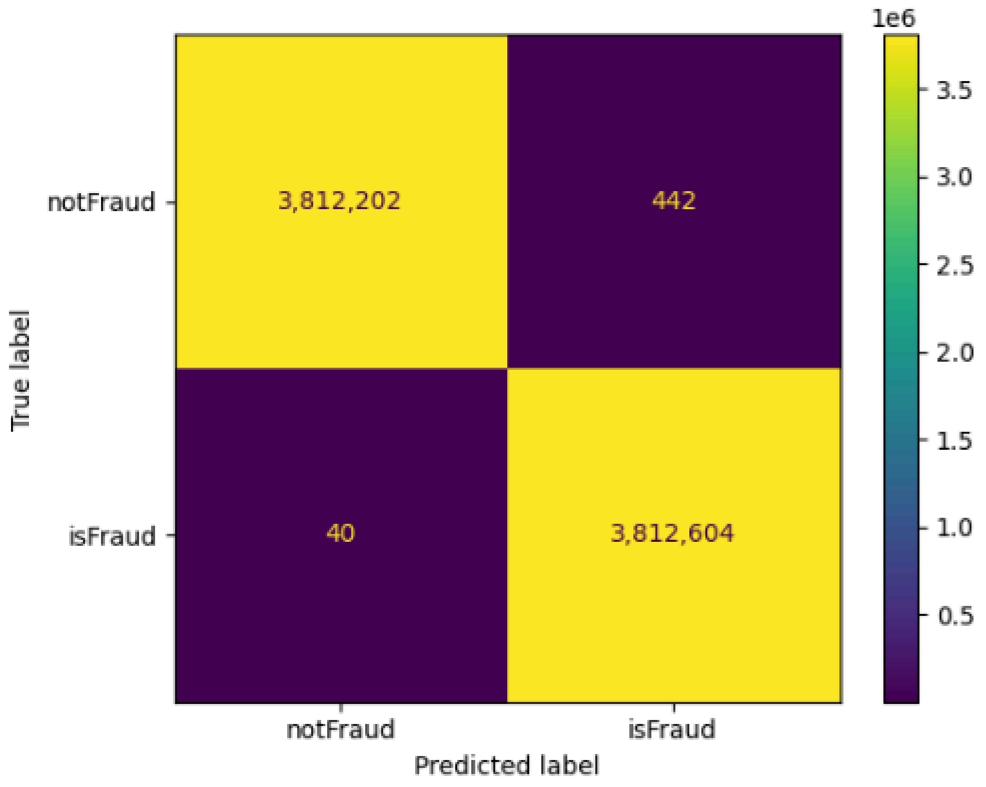
https://colab.research.google.com/drive/1FKqmYg4U02UTCxyWe0pqzJdz_PdaLqVr?usp=sharing



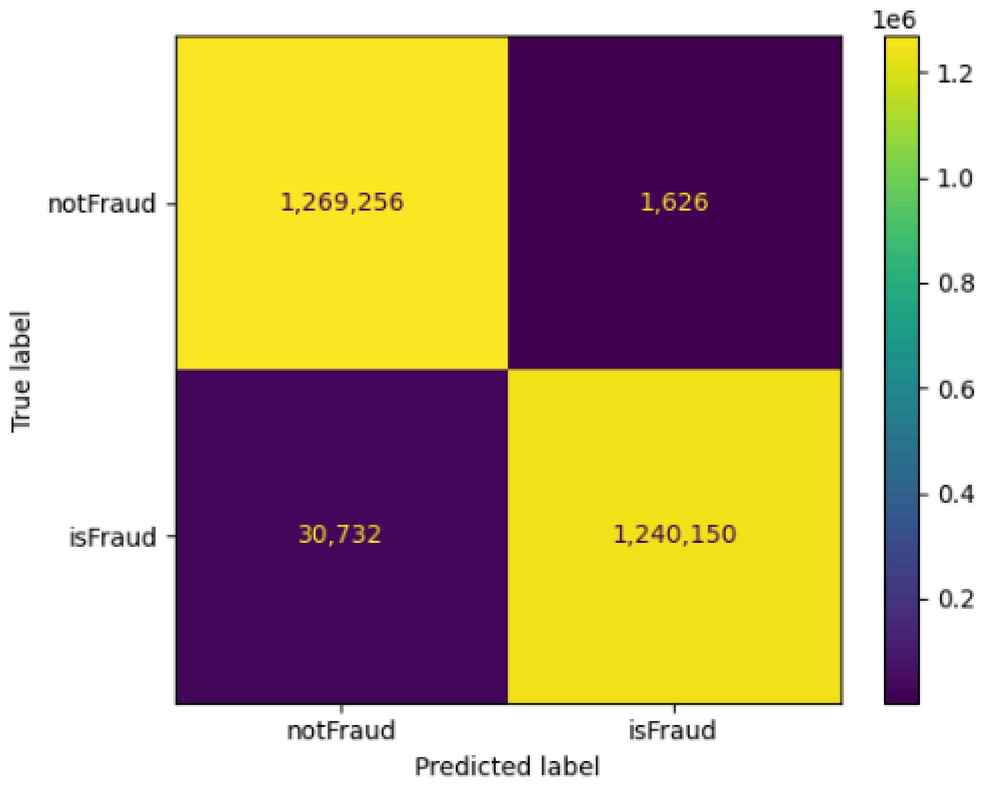
```
Training Results:
              precision    recall  f1-score   support

    notFraud       1.00      1.00      1.00   3812644
     isFraud       1.00      1.00      1.00   3812644

    accuracy                           1.00   7625288
   macro avg       1.00      1.00      1.00   7625288
weighted avg       1.00      1.00      1.00   7625288


Testing Results:
              precision    recall  f1-score   support

    notFraud       0.98      1.00      0.99   1270882
     isFraud       1.00      0.98      0.99   1270882

    accuracy                           0.99   2541764
   macro avg       0.99      0.99      0.99   2541764
weighted avg       0.99      0.99      0.99   2541764
```

After receiving feedback from the presentation, I decided to make the classes of notFraud and isFraud as balanced because we want to eliminate bias entirely in order to improve the model's prediction regarding fraudulent transactions. This means that the model will perform better than previously so the prediction model should focus on recall than precision as recall determines fraudulent transactions.

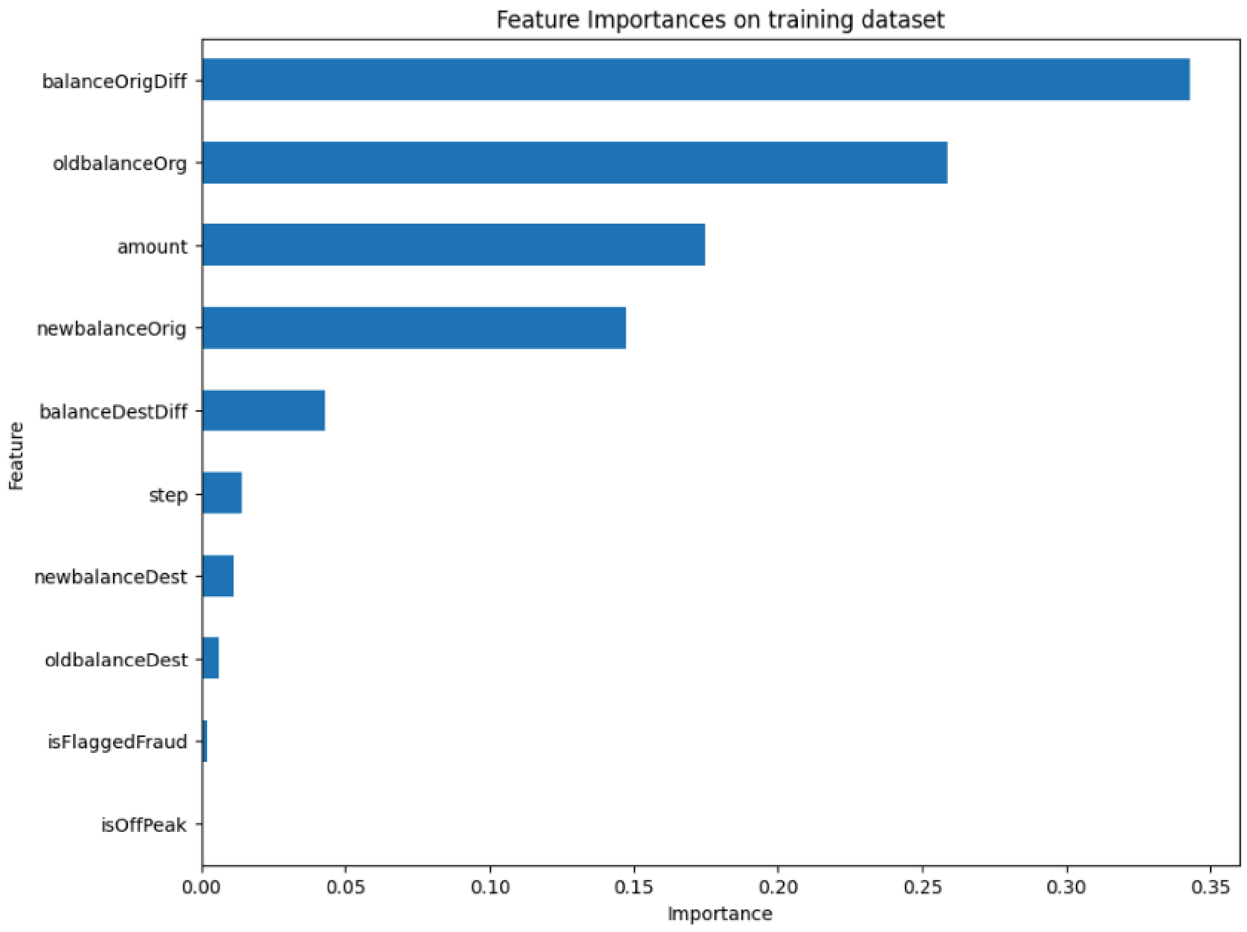
```
# evaluate model with Confusion Matrix on training data
cm = confusion_matrix(y_train_smote, y_train_pred, labels = y_train.unique())
class_labels = ['notFraud', 'isFraud']
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = class_labels)
disp.plot(values_format = ',d')  # set values_format to 'd' to display as whole numbers
plt.show()
```

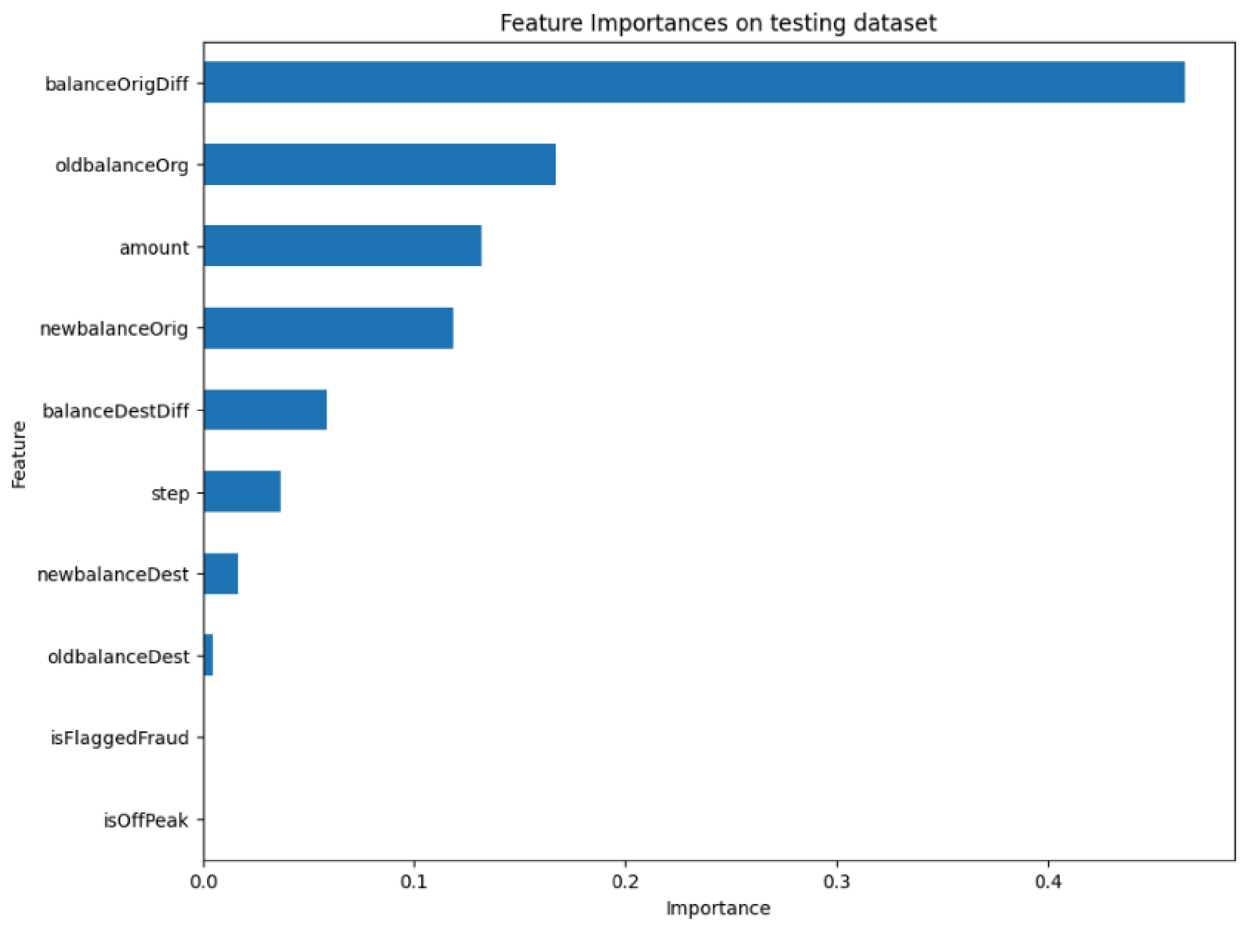


```
# evaluate model with Confusion Matrix on testing data
cm = confusion_matrix(y_test_smote, y_test_pred, labels = clf_optimized.classes_)
class_labels = ['notFraud', 'isFraud']
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = class_labels)
disp.plot(values_format = ',d')  # set values_format to 'd' to display as whole numbers
plt.show()
```

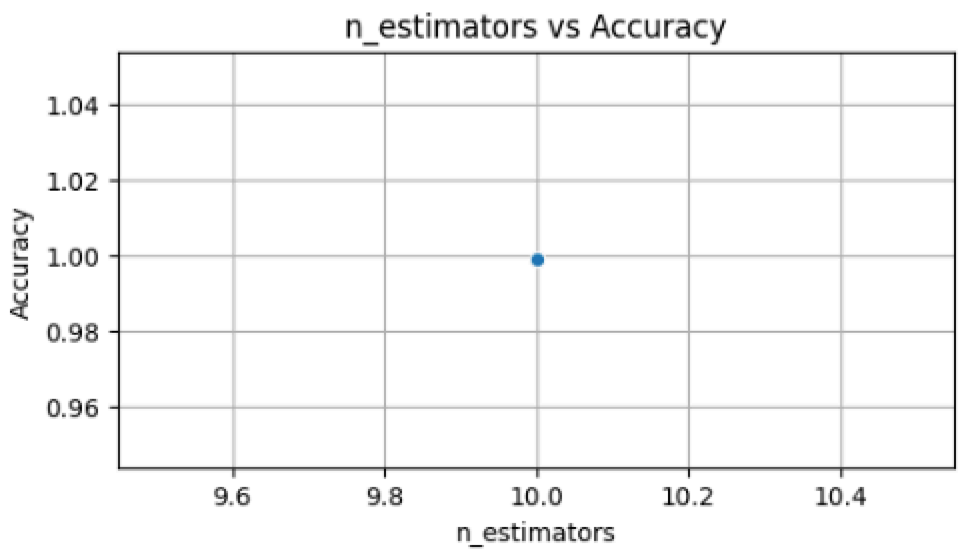


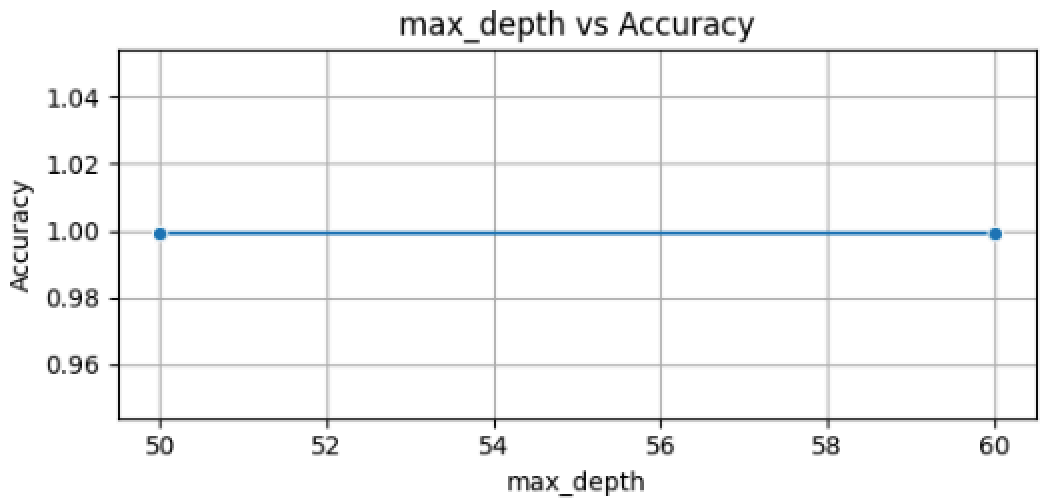
Looking at the updated confusion matrices, for the training confusion matrix, there is high accuracy which means overfitting since it performed well on the training dataset with very few false negatives and false positives. Moreover, for the testing confusion matrix, there was a drop in accuracy because there was an increase in false negatives meaning those are real fraudulent scenarios that the model failed to predict which is bad if people rely on this model for real world situations overall. The recall is affected by false negative count meaning the model technically missed some fraudulent transactions. In addition, there is still overfitting since the model has perfect accuracy on the training set, however performance suffers on the testing set since it cannot generalize unknown data. Finally, the false-negative on the test set is extreme since this can lead to misidentifying real fraud is critical to determining fraudulent transactions.

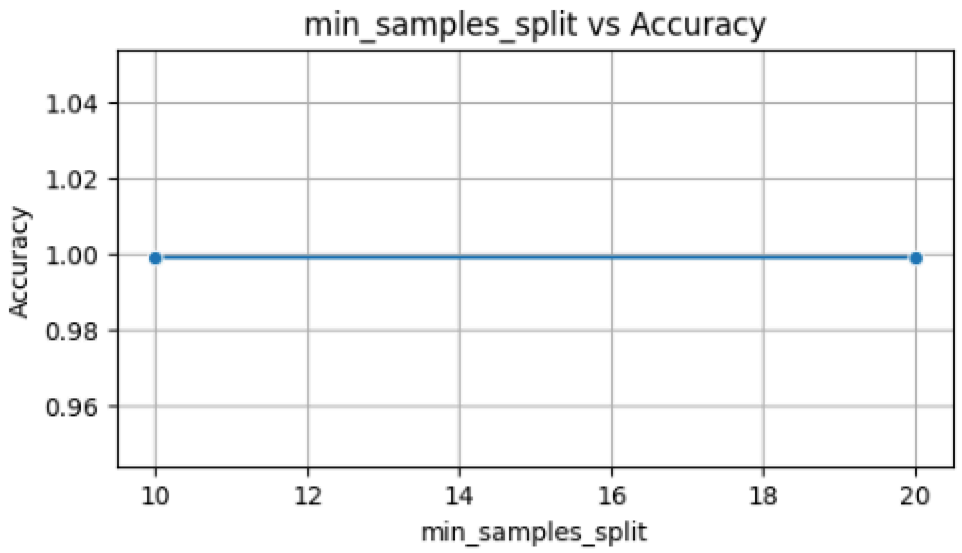

Feature Importances on training dataset
balanceOrigDiff
oldbalanceOrg
amount
newbalanceOrig
balanceDestDiff
step
newbalanceDest
oldbalanceDest
isFlaggedFraud
isOffPeak
Feature
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Importance

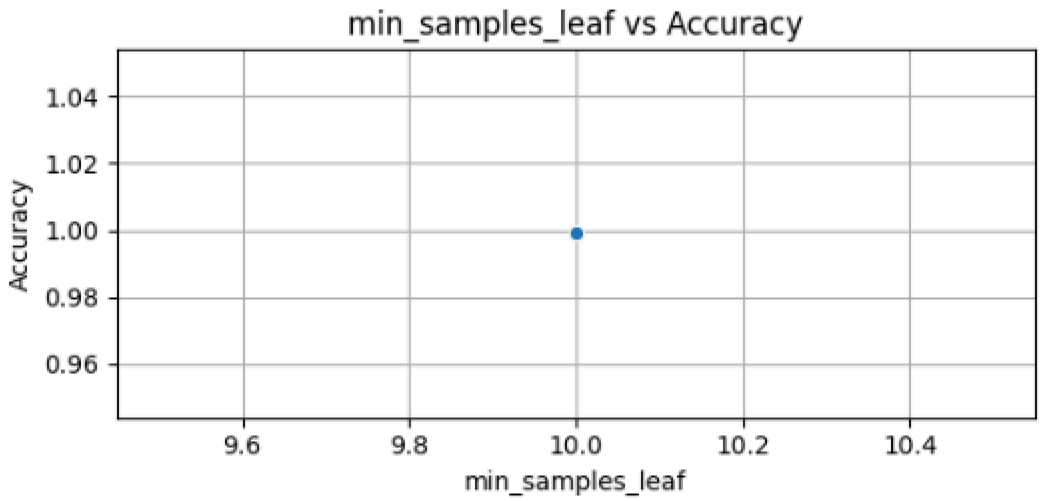


After that experiment, we decided to remove features that weren't necessary in order to improve the model's prediction on fraudulent transactions overall and attempted to plot some relationships between some top features from the testing dataset.

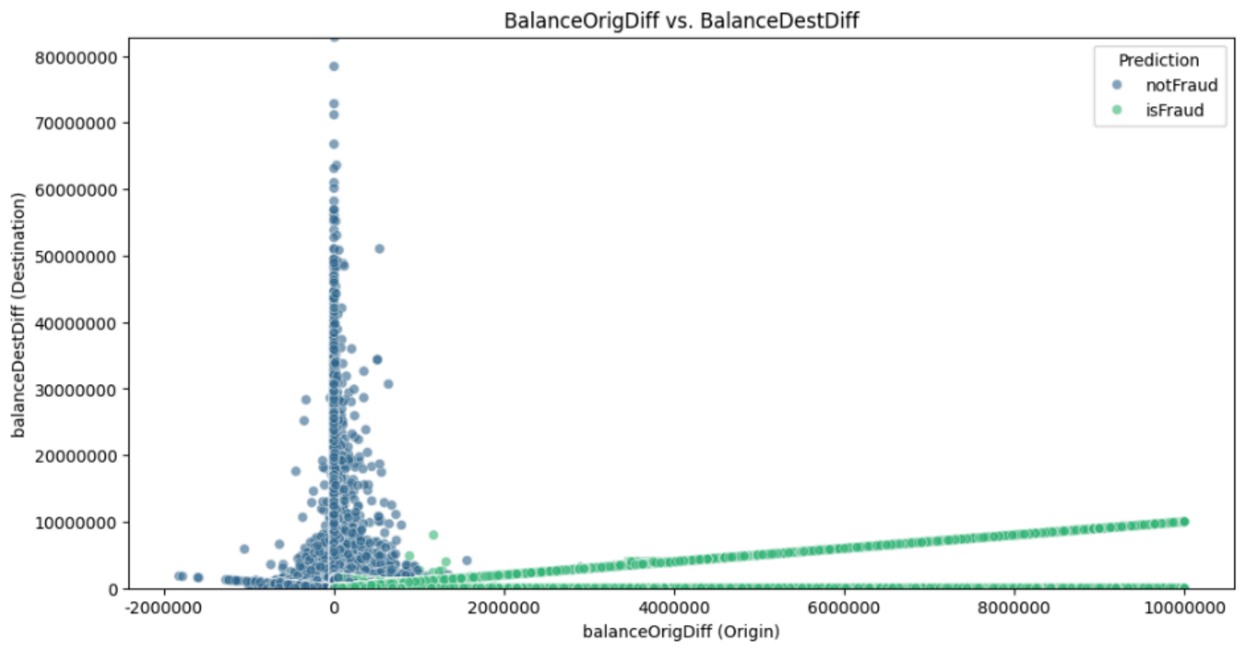

n_estimators vs Accuracy
Accuracy
1.04
1.02
1.00
0.98
0.96
9.6
9.8
10.0
10.2
10.4
n_estimators


max_depth vs Accuracy
Accuracy
1.04
1.02
1.00
0.98
0.96
50
52
54
56
58
60
max_depth

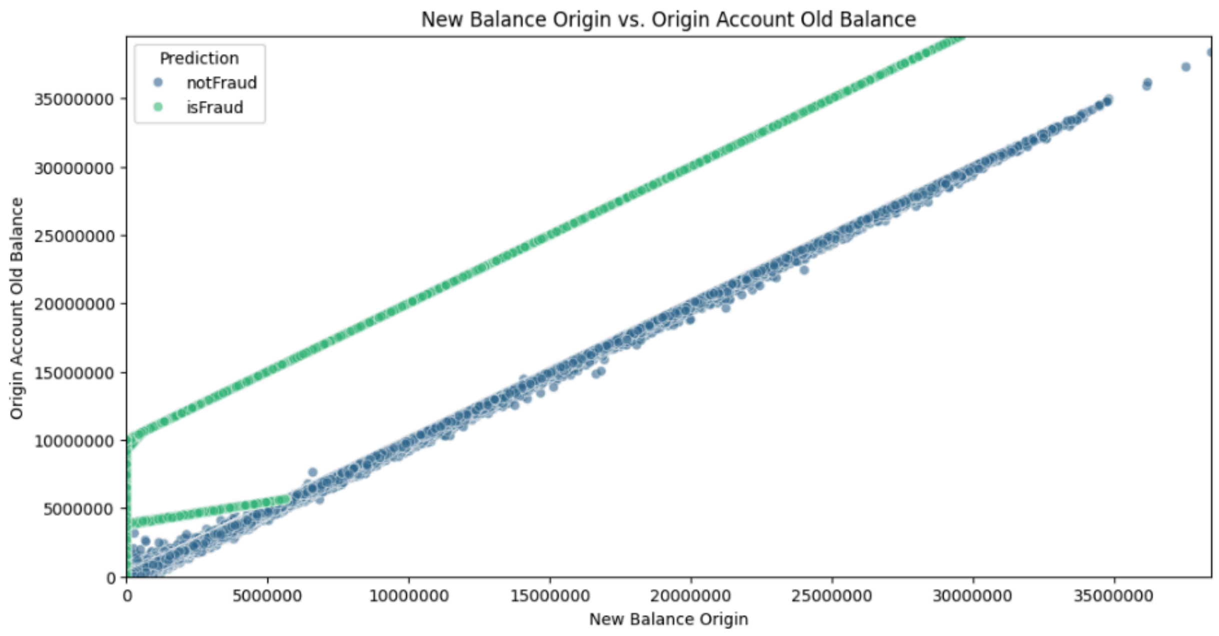




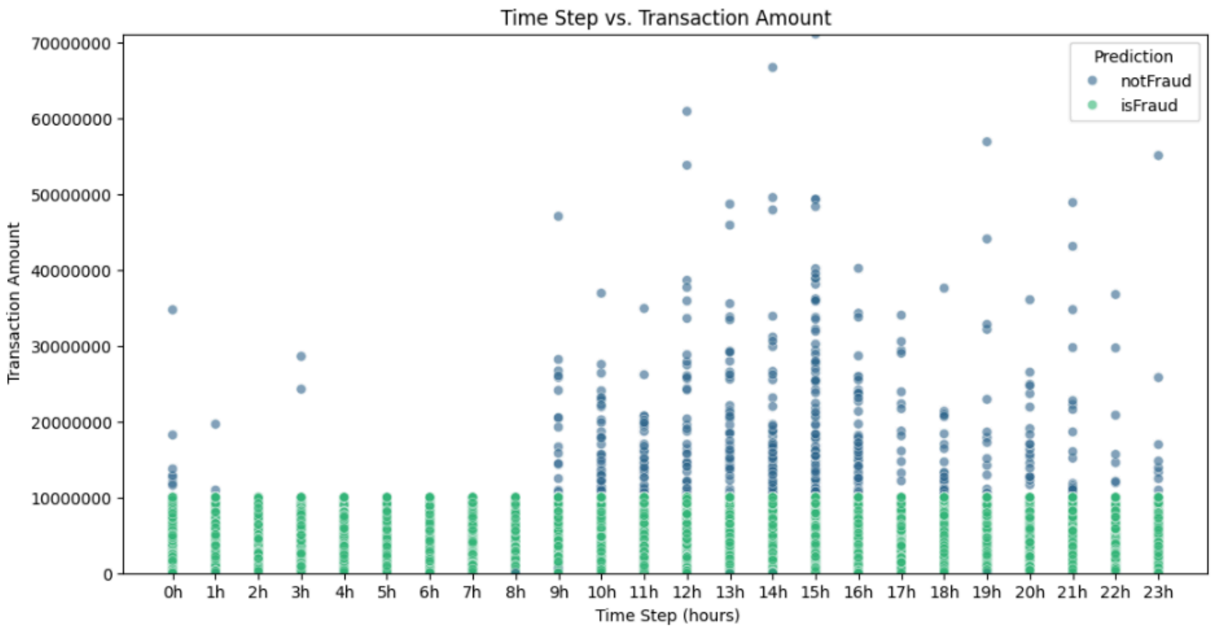
These are the most tuned hyperparameters from applying randomized cross validation on the random forest classifier in order to determine optimal hyperparameters to test the model with unknown data and improve its prediction model.



This relationship between difference balance destination and difference balance origin illustrates fraudulent transactions around 0 to 10,000,000 dollars on the x-axis.



The relationship between new balance origin and old balance origin demonstrates a linear relationship that seems consistent as bad actors are more likely to attempt fraudulent transactions around 10,000,000 to 35,000,000 old balance.



The relationship between step and amount demonstrates the specific hour during a 24 hour period where fraudulent transactions may occur from the model's prediction analysis of the data. Moreover, fraudulent activity appears to occur with small amounts because bad actors don't want to create suspicious activity so they would extract some amounts of money in order to avoid attention from the account owners during a 24 hour time period.

TEAM MEMBER CONTRIBUTION:

Rogelio worked on the machine learning algorithm, Random forest, in order to predict whether a transaction is considered fraud or not by visualizing the dataset using important features, confusion matrix, hyperparameters, and scatter plots to understand the impact fraudulent transactions can have in the modern world and looking at the dataset to get a better understanding at what bad actors really want from the crime.

Adolfo designed, implemented and tested the Decision Tree model, worked on the project's reports, and the presentation where he explained the decision tree model's performance.

Kento contributed to the project's documentation and presentation aspects. During the presentation, the Rule-Based Approach was explained, emphasizing its simplicity and transparency, along with its limitations in scalability and flexibility, to highlight its role as a foundational model in detecting fraudulent transactions.