Predictive Modeling Against Financial Fraud

Rehan Khan

Agenda

Introduction

Problem

Goal

Modeling Steps

Conclusion

Introduction

For our analysis, we utilized one of PaySim's publicly available datasets on Kaggle. By leveraging this synthetic data, our goal is to gain a deeper understanding of features associated with fraudulent activities through data analysis, ultimately enabling us to make accurate predictions on unseen data. We further improved prediction performance by implementing various machine learning algorithms and rigorously evaluating each model.

Problem

Our data contains fraudulent transactions.

The transactions that were fraudulent were not all accurately flagged by the current model.

Goal

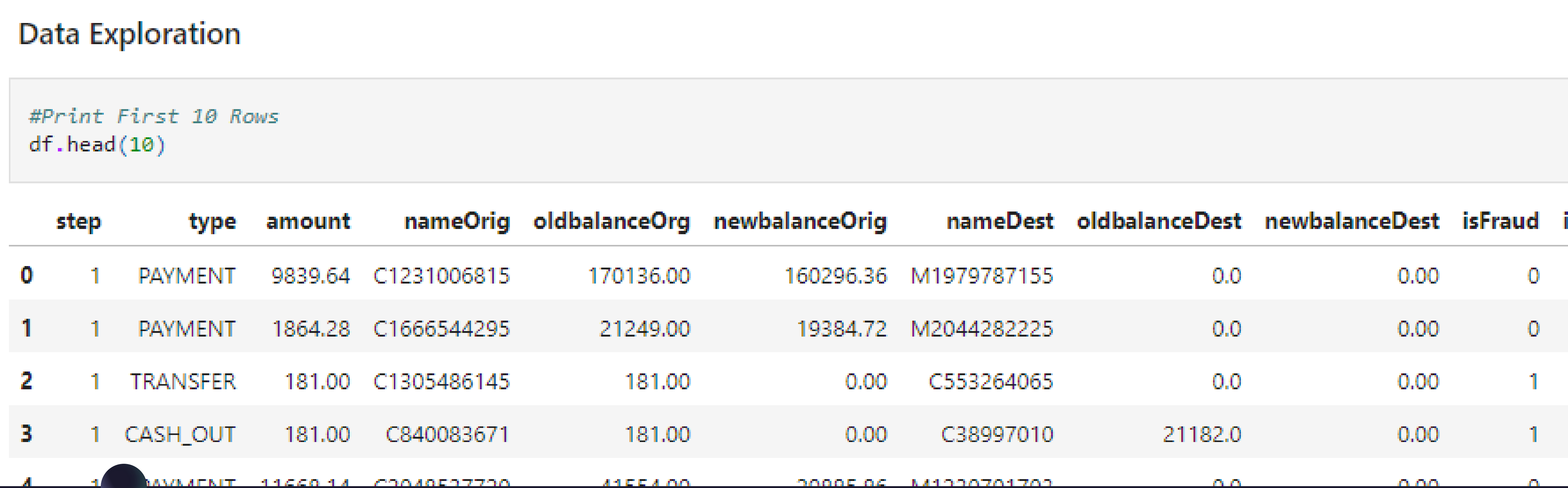
Create an improved fraud detection model that accurately predicts fraudulent transactions by a percentage of 90 or greater.

Steps of Machine Learning Model Creation

I.Data Wrangling	Cleaning, organizing and transforming raw data.
II. Exploratory Data Analysis	Step that identifies patterns in data and understanding outliers and features of dataset
III. Data Pre-processing	Changing raw data into a clean, usable format to prepare for final step of creating the machine learning model
IV. Modeling	Final step of the process where we apply different algorithms to processed data to see which model is most effective in predicting our target of a fraudulent transaction

Data Wrangling

Column Name	Data Type	Content
step	integer	unit of time in the real world. I step is I hour of time. Total steps 744 (30 days simulation)
type	string/categorical	transaction type
amount	float	transaction amount
nameOrig	string	customer initiating transaction
oldbalanceOrg	float	initial balance before the transaction
newbalanceOrig	float	new balance after transaction
nameDest	string	customer who is recipient
oldbalanceDest	float	initial balance of recipient
isFraud	boolean	new balance of recipient after transaction
isFlaggedFraud	boolean	if transaction is flagged as fraudulent
newbalanceDest	float	new balance after recipient transaction



Data Exploration

```
#Print First 10 Rows
df.head(10)
```

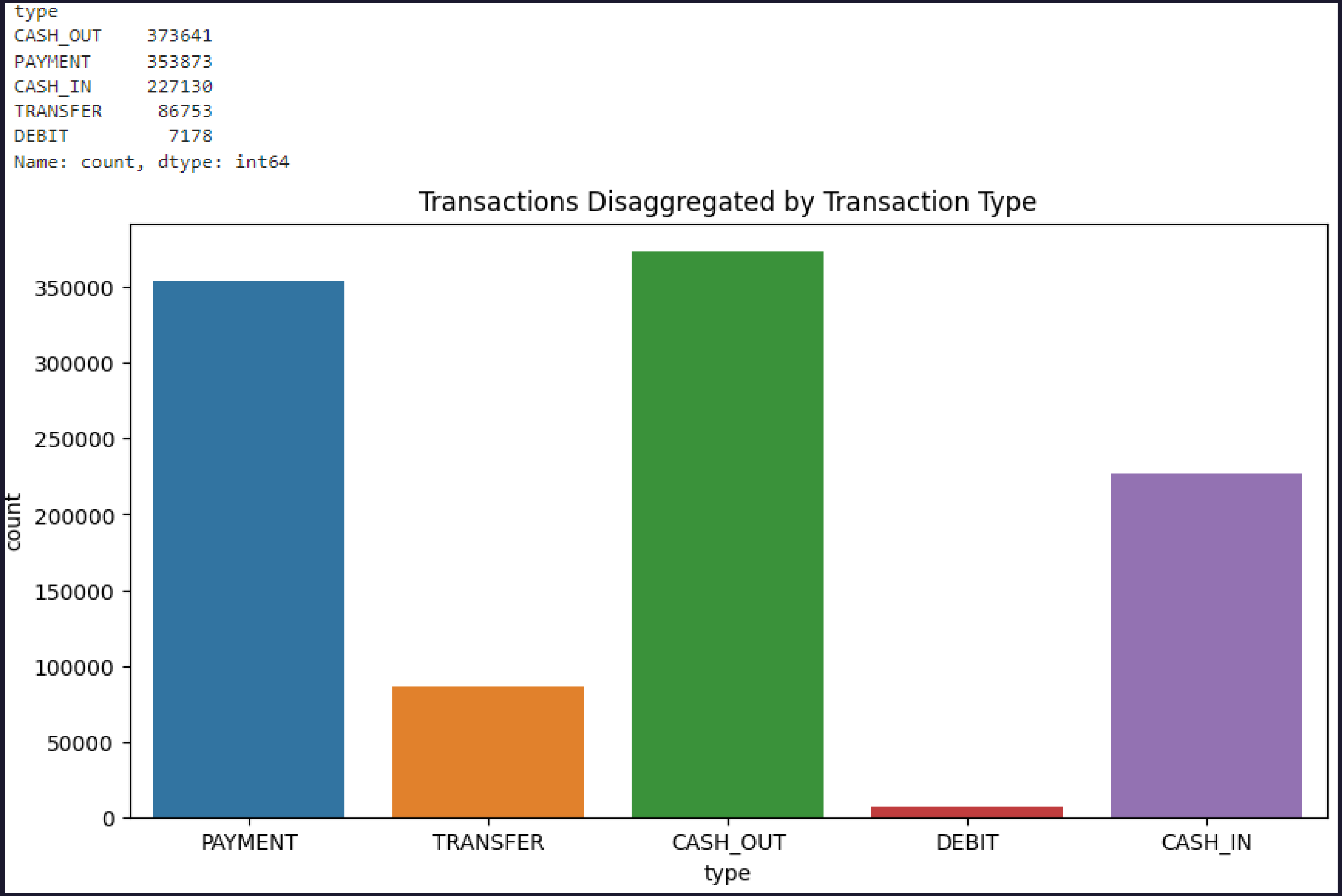
	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0	0.00	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0	0.00	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0	0.00	1
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0	0.00	1

Data Wrangling

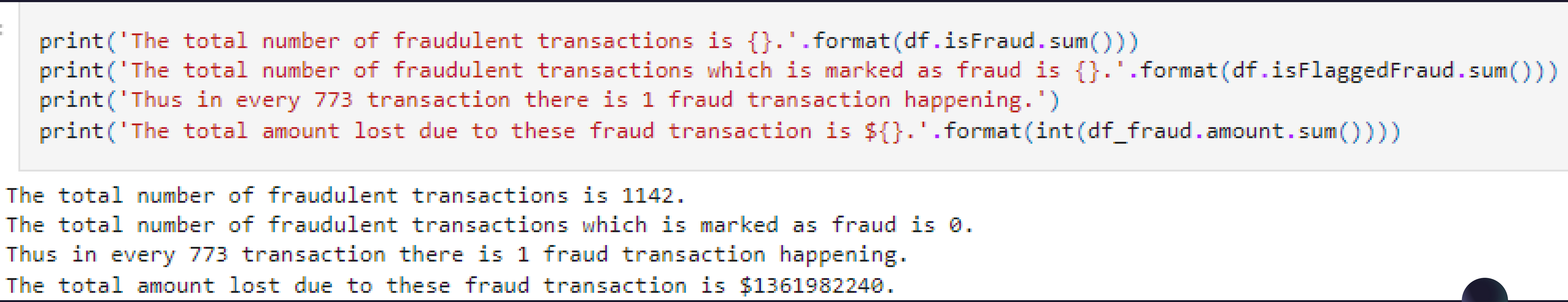
- To facilitate analysis, we imported the file into Jupyter Notebooks as a DataFrame object and processed it using the Python programming language. The file was loaded into a DataFrame structure, consisting of 6,362,620 rows and 11 columns.

Exploratory Data Analysis

Figure on the right shows the types of transactions that were found in the dataset and their frequency.



```
type
CASH_OUT    373641
PAYMENT     353873
CASH_IN     227130
TRANSFER     86753
DEBIT         7178
Name: count, dtype: int64
```



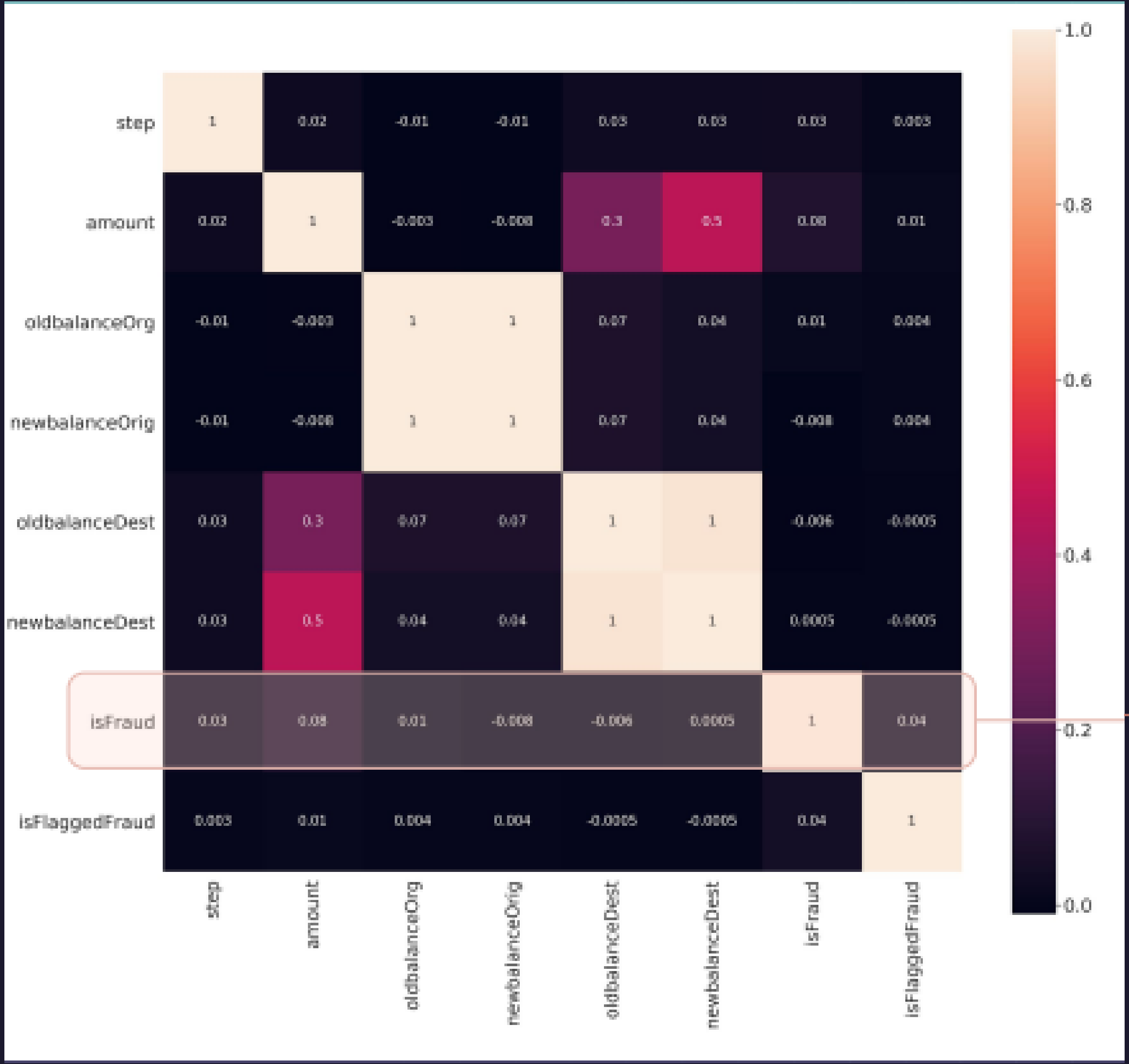
```
print('The total number of fraudulent transactions is {}.'.format(df.isFraud.sum()))
print('The total number of fraudulent transactions which is marked as fraud is {}.'.format(df.isFlaggedFraud.sum()))
print('Thus in every 773 transaction there is 1 fraud transaction happening.')
print('The total amount lost due to these fraud transaction is ${}.'.format(int(df_fraud.amount.sum())))
```

```
The total number of fraudulent transactions is 1142.
The total number of fraudulent transactions which is marked as fraud is 0.
Thus in every 773 transaction there is 1 fraud transaction happening.
The total amount lost due to these fraud transaction is $1361982240.
```

Exploratory Data Analysis

- Figure above shows how many fraudulent transactions we had, the frequency of those fraudulent transactions, and the financial loss associated with fraudulent transactions.

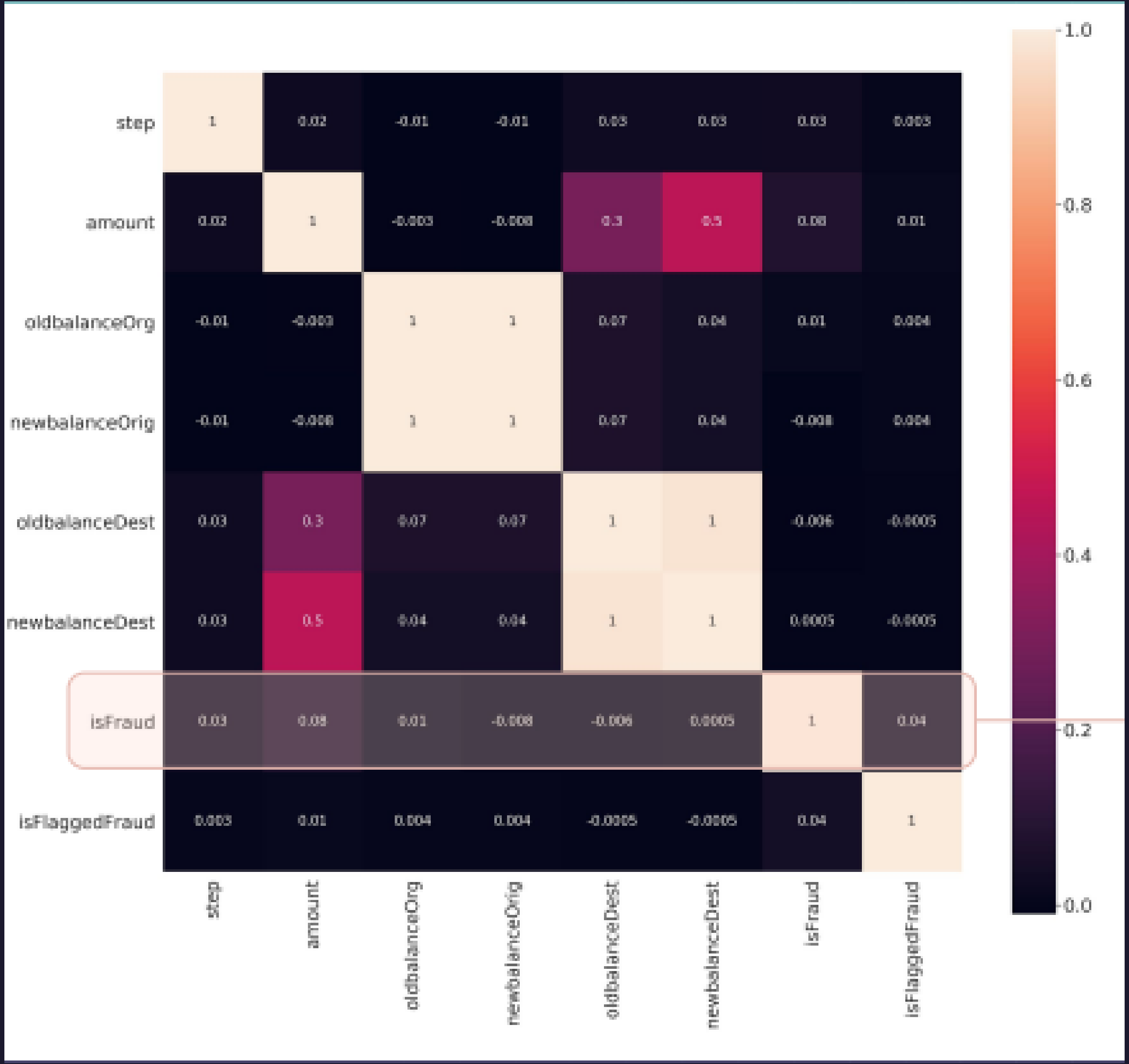
Correlation Coefficient Analysis of Fraudulent Transactions



	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
step	1	0.02	-0.01	-0.01	0.03	0.03	0.03	0.003
amount	0.02	1	-0.003	-0.008	0.3	0.5	0.08	0.01
oldbalanceOrg	-0.01	-0.003	1	1	0.07	0.04	0.01	0.004
newbalanceOrig	-0.01	-0.008	1	1	0.07	0.04	-0.008	0.004
oldbalanceDest	0.03	0.3	0.07	0.07	1	1	-0.006	-0.0005
newbalanceDest	0.03	0.5	0.04	0.04	1	1	0.0005	-0.0005
isFraud	0.03	0.08	0.01	-0.008	-0.006	0.0005	1	0.04
isFlaggedFraud	0.003	0.01	0.004	0.004	-0.0005	-0.0005	0.04	1

Column Name	r
amount	0.077
isFlaggedFraud	0.044
step	0.032
oldbalanceOrg	0.010
newbalanceDest	0.001
oldbalanceDest	-0.06
newbalanceOrig	-0.008

Correlation Coefficient Analysis of Fraudulent Transactions

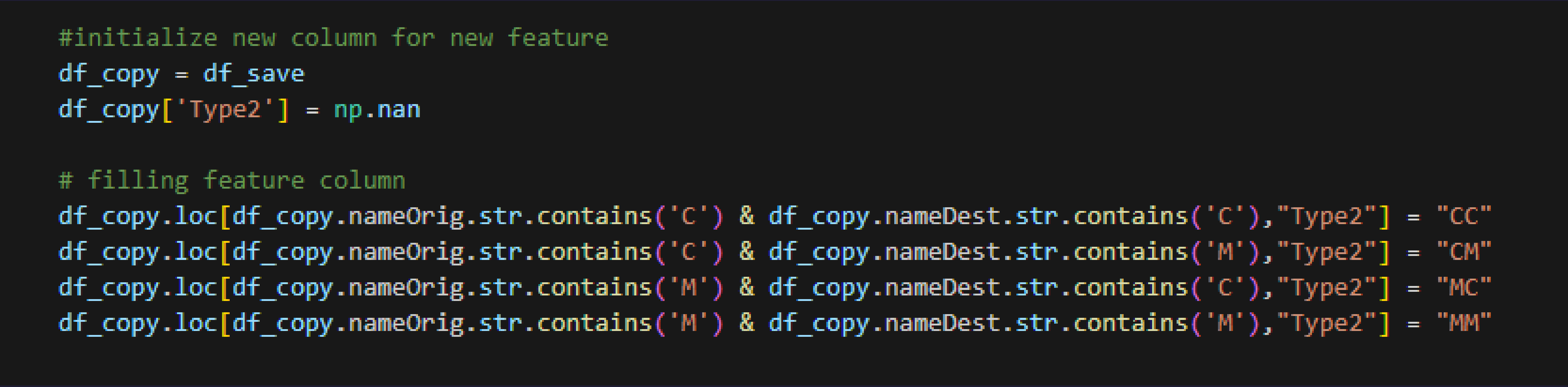


	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
step	1	0.02	-0.01	-0.01	0.03	0.03	0.03	0.003
amount	0.02	1	-0.003	-0.008	0.3	0.5	0.08	0.01
oldbalanceOrg	-0.01	-0.003	1	1	0.07	0.04	0.01	0.004
newbalanceOrig	-0.01	-0.008	1	1	0.07	0.04	-0.008	0.004
oldbalanceDest	0.03	0.3	0.07	0.07	1	1	-0.006	-0.0005
newbalanceDest	0.03	0.5	0.04	0.04	1	1	0.0005	-0.0005
isFraud	0.03	0.08	0.01	-0.008	-0.006	0.0005	1	0.04
isFlaggedFraud	0.003	0.01	0.004	0.004	-0.0005	-0.0005	0.04	1

Despite the importance of feature selection, using correlation as a metric did not allow us to narrow down our focus to a few features. None of the columns exhibited a satisfactory correlation score with the target variable 'isFraud' (all measured values were below |0.1|)

Data Preprocessing

For this step, I combined categorical values converting them to a new “type” to include information to process into the final machine learning algorithms I will use to test and predict our data. I did this using a numpy function to initialize a new “Type2” column while using the “.loc” function.



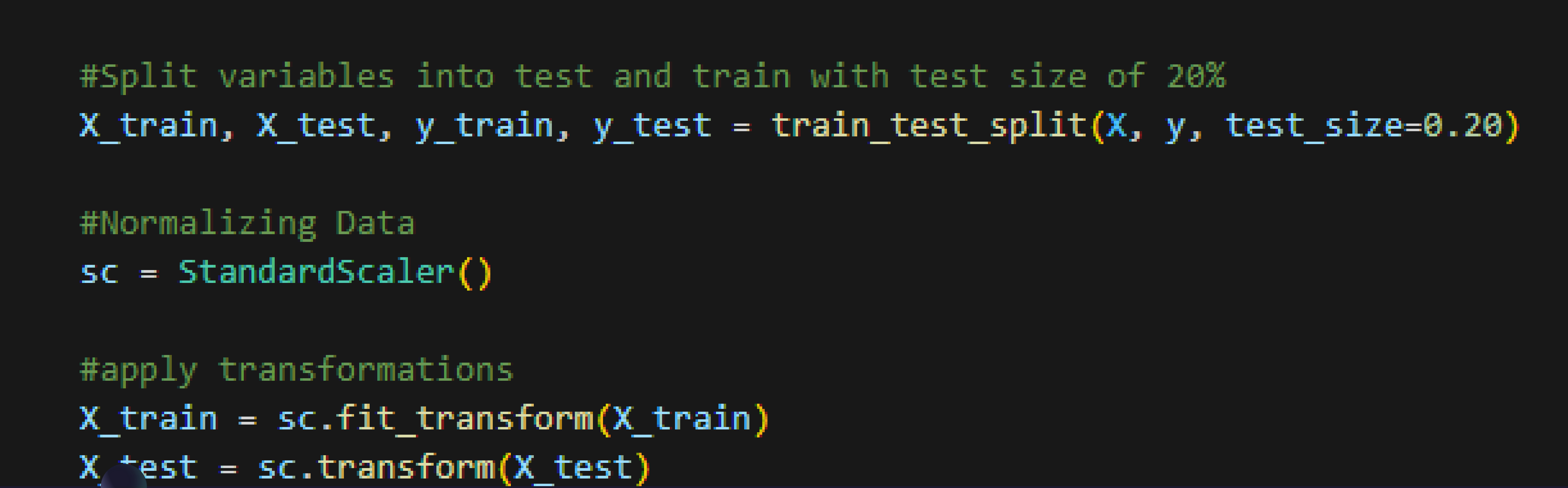
```
#initialize new column for new feature
df_copy = df_save
df_copy['Type2'] = np.nan

# filling feature column
df_copy.loc[df_copy.nameOrig.str.contains('C') & df_copy.nameDest.str.contains('C'),"Type2"] = "CC"
df_copy.loc[df_copy.nameOrig.str.contains('C') & df_copy.nameDest.str.contains('M'),"Type2"] = "CM"
df_copy.loc[df_copy.nameOrig.str.contains('M') & df_copy.nameDest.str.contains('C'),"Type2"] = "MC"
df_copy.loc[df_copy.nameOrig.str.contains('M') & df_copy.nameDest.str.contains('M'),"Type2"] = "MM"
```

Data Preprocessing

Since not all features are providing information relevant to detect fraud, it is necessary to select only the best indicators for our target and build a model based on these features only.

Modeling

In this step, we established both a training and testing set, laying the groundwork for constructing our initial and fundamental model as a baseline. This baseline serves as a reference point, offering valuable insights into potential enhancements for our subsequent models. To establish this baseline, we initiated the process by standardizing the data, ensuring a uniform input regularization.



```
#Split variables into test and train with test size of 20%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

#Normalizing Data
sc = StandardScaler()

#apply transformations
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Modeling

Data was split with 20% going towards testing and 80% going towards training the model.

Models Used

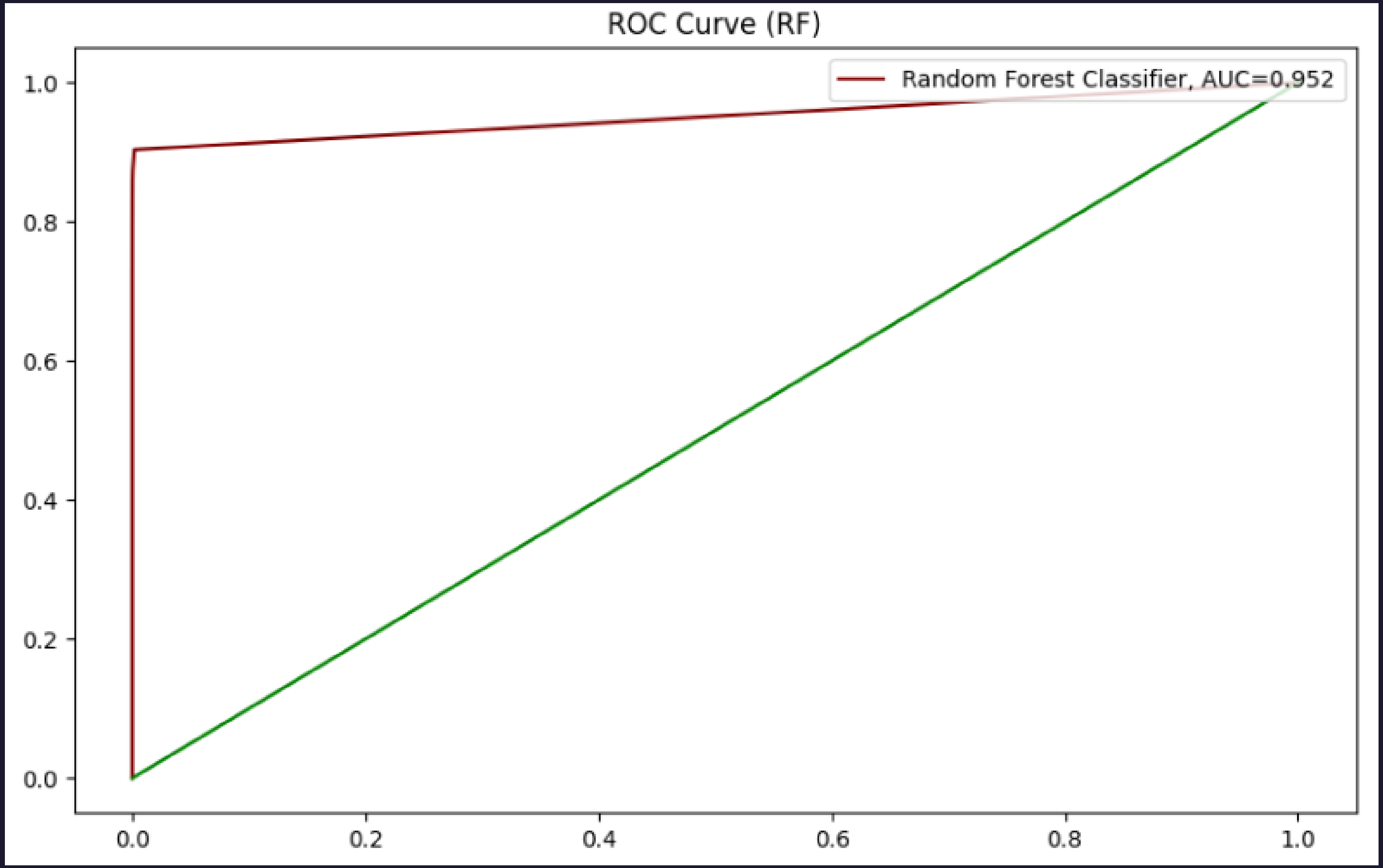
Random Forest

Decision Tree

Logistic Regression

Model Evaluation

Random Forest was our best performing model.



Conclusion

In summary, we crafted advanced predictive classifier models to enhance the efficacy of our financial fraud detection system. The outcome was a notable improvement in detecting fraudulent operations, achieving higher recall scores across all three distinct algorithms compared to the original 'isFlaggedFraud' or baseline model. Here are our key findings:

1. We found that the Random Forest Algorithm Performed the best out of the three algorithms with an AUC score of 0.952.

2. Decision Tree algorithm performed the worst with an AUC score of 0.901.

Thank You