Northeastern University
College of Professional Studies

May 18, 2023

ALY 6040: Data Mining
Online Payment Fraud Detection

Submitted to:
Dr Chinthaka Pathum Dinesh, Prof
Herath Gedara, Faculty Lecturer

Submitted by:
Abhilash Dikshit
Milan Prajapati
Minesh Patil
Murtaza Vora
Shamim Sherafati

Dataset Overview

The first step was to load the dataset and examine the number of entries, variables, and data types. The dataset contained over 6 million entries and 11 variables.
Hence, we split the dataset into train and test data with 80:20 split and we will be using test dataset further analysis.

```
display(df_raw)
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.00	0.00	1	0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.00	0.00	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776919290	0.00	339682.13	1	0
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881841831	0.00	0.00	1	0
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365125890	68488.84	6379898.11	1	0
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080388513	0.00	0.00	1	0
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873221189	6510099.11	7360101.63	1	0

6362620 rows × 11 columns

```
print('\033[1mOnline payment fraud detection:\n' + '='*32 + '\033[0m')
table = [['Type', 'Length', 'Shape'], [type(df_raw), len(df_raw), df_raw.shape]]
print(tabulate(table, headers='firstrow', tablefmt='fancy_grid'))
```

Online payment fraud detection:
================================

Type	Length	Shape
<class 'pandas.core.frame.DataFrame'>	6362620	(6362620, 11)

```
# Split the dataset into training and testing sets (80:20)
train, test = train_test_split(df_raw, test_size=0.2, random_state=42)

# Print the lengths of the training and testing sets
print("Length of training set:", len(train))
print("Length of testing set:", len(test))
```

```
Length of training set: 5090096
Length of testing set: 1272524
```

Column names:

- **Step:** represents a unit of time where 1 step equals 1 hour
- **Type:** type of online transaction
- **Amount:** the amount of the transaction
- **NameOrig:** customer starting the transaction
- **OldbalanceOrg:** balance before the transaction
- **NewbalanceOrig:** balance after the transaction
- **NameDest:** recipient of the transaction
- **OldbalanceDest:** initial balance of the recipient before the transaction
- **NewbalanceDest:** the new balance of the recipient after the transaction
- **IsFraud:** fraud transaction

EDA on test dataset due to large data:

====================================

	Datatype	Null_Count	Unique_Value
step	int64	0	697
type	object	0	5
amount	float64	0	1219164
nameOrig	object	0	1272160
oldbalanceOrg	float64	0	460453
newbalanceOrig	float64	0	548278
nameDest	object	0	777464
oldbalanceDest	float64	0	729323
newbalanceDest	float64	0	765658
isFraud	int64	0	2
isFlaggedFraud	int64	0	2

Dataset Description:

	step	amount	oldbalanceOrg	newbalanceOrig \
count	1.272524e+06	1.272524e+06	1.272524e+06	1.272524e+06
mean	2.434153e+02	1.802790e+05	8.358581e+05	8.573116e+05
std	1.423745e+02	6.127373e+05	2.893421e+06	2.929707e+06
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.560000e+02	1.336609e+04	0.000000e+00	0.000000e+00
50%	2.390000e+02	7.489837e+04	1.432206e+04	0.000000e+00
75%	3.350000e+02	2.090111e+05	1.073550e+05	1.446149e+05
max	7.420000e+02	6.933732e+07	4.489219e+07	3.894623e+07

	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	1.272524e+06	1.272524e+06	1.272524e+06	1.272524e+06
mean	1.105138e+06	1.229909e+06	1.273060e-03	2.357519e-06
std	3.428096e+06	3.704978e+06	3.565727e-02	1.535420e-03
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	1.327846e+05	2.152613e+05	0.000000e+00	0.000000e+00
75%	9.483279e+05	1.115401e+06	0.000000e+00	0.000000e+00
max	3.553805e+08	3.560159e+08	1.000000e+00	1.000000e+00

Statistical Measure:

The chart below displays each variable's mean, median, mode and quantiles as well as other common statistical measures.

Exploratory Data Analysis

In order to verify the overall extent of fraudulent activity, we are currently extracting a subset of the dataset from the "isFraud" column, specifically isolating instances where the values are either 0 or 1.

```
# To check the total fraud in the dataset
print('No Frauds', round(df['isFraud'].value_counts()[0]/len(df) * 100,2), '% of the dataset')
print('Frauds', round(df['isFraud'].value_counts()[1]/len(df) * 100,2), '% of the dataset')
```

```
No Frauds 99.87 % of the dataset
Frauds 0.13 % of the dataset
```

In this instance, we examined each payment category provided by the bank, along with their corresponding transaction volumes. This analysis will provide us with a comprehensive understanding of the frequency of payment channels utilized.

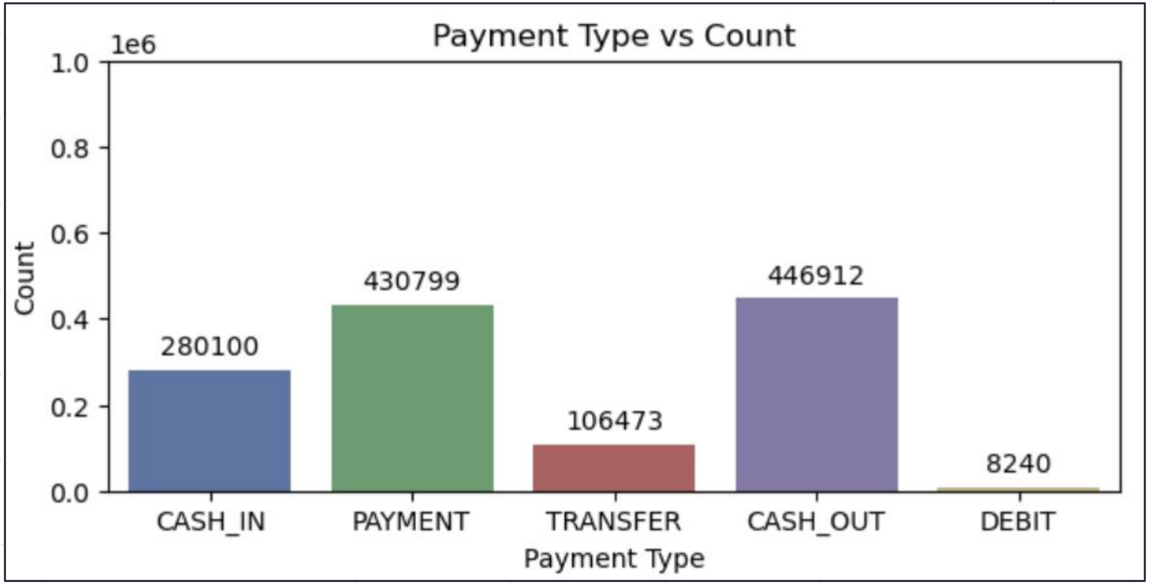


Fig: Count plot to show Payment Type vs Count:

Countplot For Frequency of Transaction Types For Fraud

This graph displays the distribution of payment types with a focus on fraudulent transactions. It provides insights into the prevalence and impact of fraudulent activities within different payment methods. There is an equal distribution of fraudulent transactions between cash and transfer.

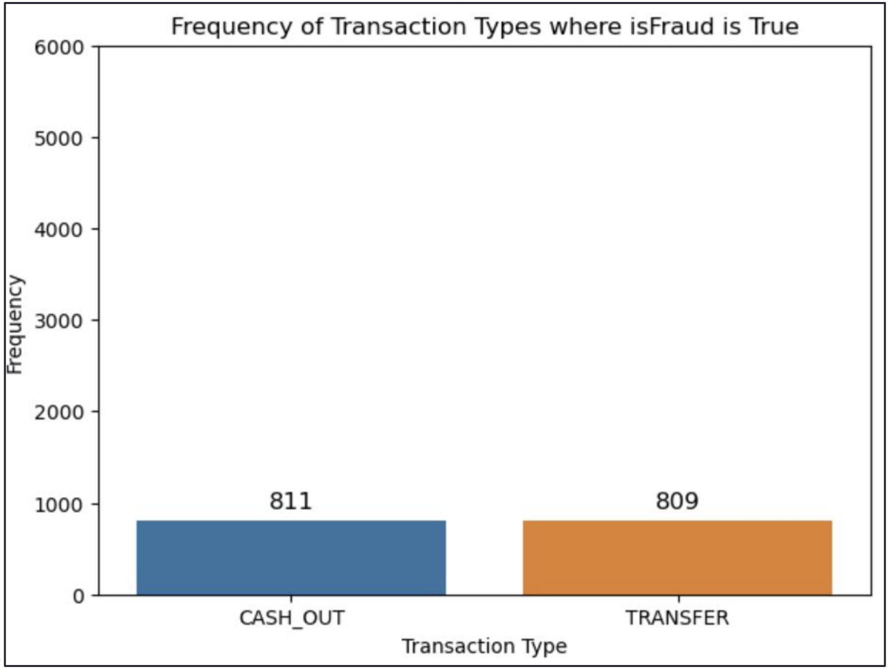


Fig: Countplot to show the Frequency of Transaction Types where Fraud happened

Box Plot For Each Numerical Variable

To enhance the visual representation, we categorized the columns into "numerical" and "categorical" types and generated a boxplot for each numerical variable to assess its skewness.

```
numerical = ['step',
 'amount',
 'oldbalanceOrg',
 'newbalanceOrig',
 'oldbalanceDest',
 'newbalanceDest']

categorical = ['type', 'nameOrig', 'nameDest', 'isFlaggedFraud']
```

```
# checking boxplots
def boxplots_custom(dataset, columns_list, rows, cols, suptitle):
    fig, axs = plt.subplots(rows, cols, sharey=True, figsize=(16,5))
    fig.suptitle(suptitle,y=1, size=25)
    axs = axs.flatten()
    for i, data in enumerate(columns_list):
        sns.boxplot(data=dataset[data], orient='h', ax=axs[i])
        axs[i].set_title(data + ', skewness is: '+str(round(dataset[data].skew(axis = 0, skipna = True),2)))

boxplots_custom(dataset=df, columns_list=numerical, rows=2, cols=3, suptitle='Boxplots for each variable')
plt.tight_layout()
```

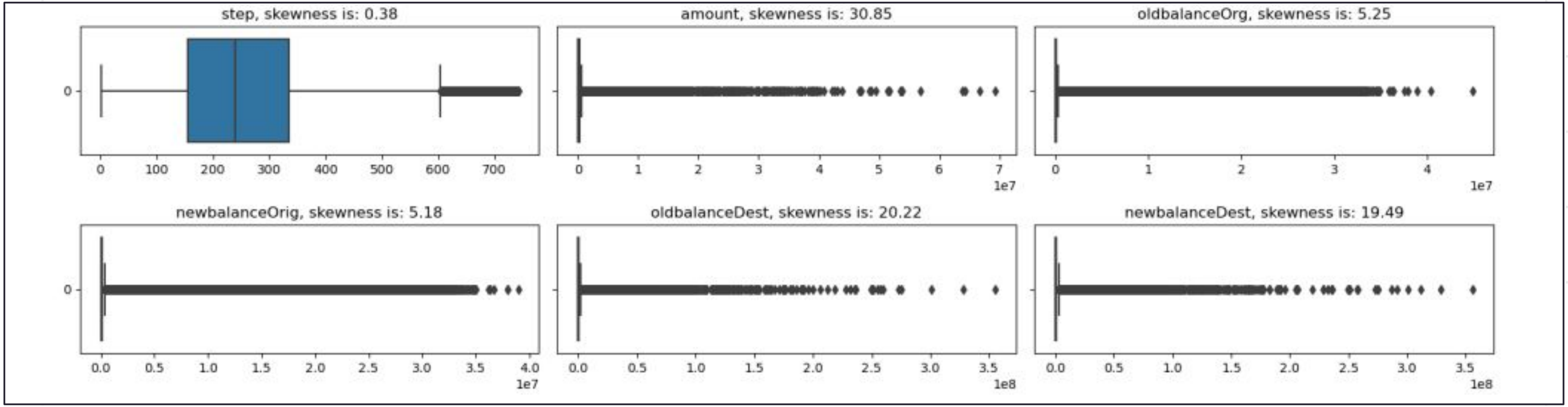


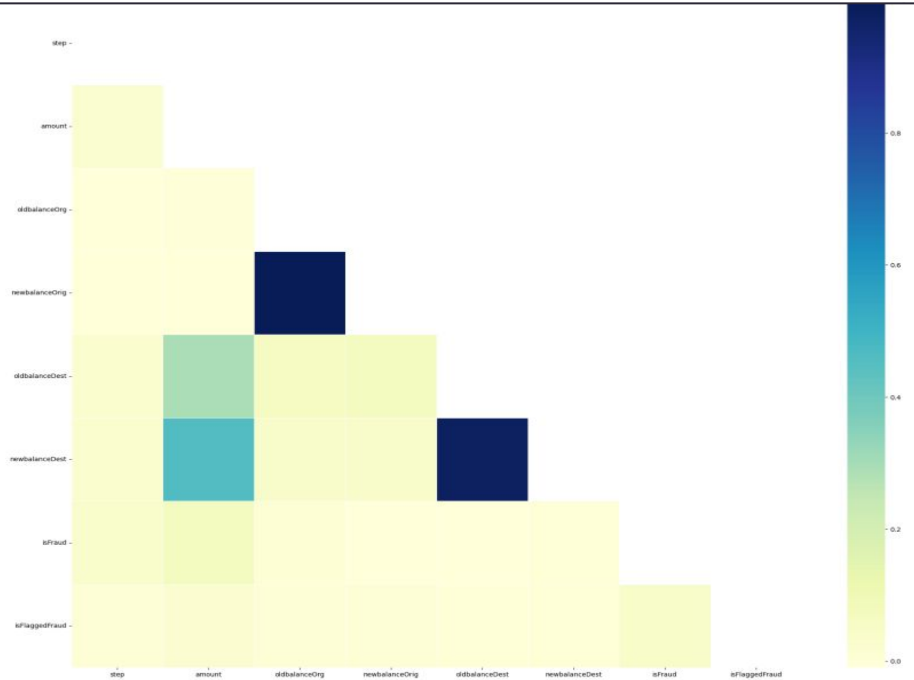
Fig: Boxplot for each numerical value

Correlation Plot:

A correlation matrix is a mathematical representation that provides valuable insights into the relationships between multiple variables within a dataset. It is commonly used in statistics and data analysis to explore the strength and direction of associations between pairs of variables. We can observe that there is a relatively low correlation between our variables. However, there is some degree of correlation between old balance, new balance, and amount.

As shown in the heatmap, we see there is an imbalance in our variables.

Note: We are considering our test dataset as the number of rows is > 6 million.



4.6 Correlation plot

Predictive algorithms: Data Scaling

As our dataset has approximately 6.3 million rows. It was important to scale the dataset as we had different ranges of values in the features. Robust scaling is a valuable preprocessing technique that provides a robust and resistant approach to feature scaling while mitigating the influence of outliers. Using robust statistical measures such as the median and interquartile range allows for more reliable and accurate data analysis and model building.

	0	1	2	3	4	5	6	7
0	-1.0	-0.909088	-0.999244	-0.099515	-1.791203e-06	-0.595314	-0.224912	-0.262478
1	-1.0	-0.909088	-0.999244	-0.099513	-8.956013e-07	-0.595313	-0.224912	-0.262478
2	-0.5	-0.909088	-0.999244	-0.099511	0.000000e+00	-0.595312	-0.224912	-0.262478
3	0.0	-0.909088	-0.999244	-0.099511	0.000000e+00	-0.595312	-0.224912	-0.262478
4	-1.0	-0.909087	-0.999243	-0.099508	8.956013e-07	-0.595311	-0.224912	-0.262478
...	...	...	...	...	...	...	...	...
6362615	0.0	1.097871	1.000724	4.019811	0.000000e+00	0.693388	-0.224912	1.434163
6362616	-0.5	1.097871	1.000724	4.019813	0.000000e+00	1.615955	-0.224912	-0.262478
6362617	0.0	1.097871	1.000724	4.019813	0.000000e+00	0.280299	1.555804	1.505169
6362618	-0.5	1.097872	1.000725	4.019815	0.000000e+00	1.615956	-0.224912	-0.262478
6362619	0.0	1.097872	1.000725	4.019815	0.000000e+00	-0.376457	1.555804	1.505170

6362620 rows × 8 columns

Principal Component Analysis

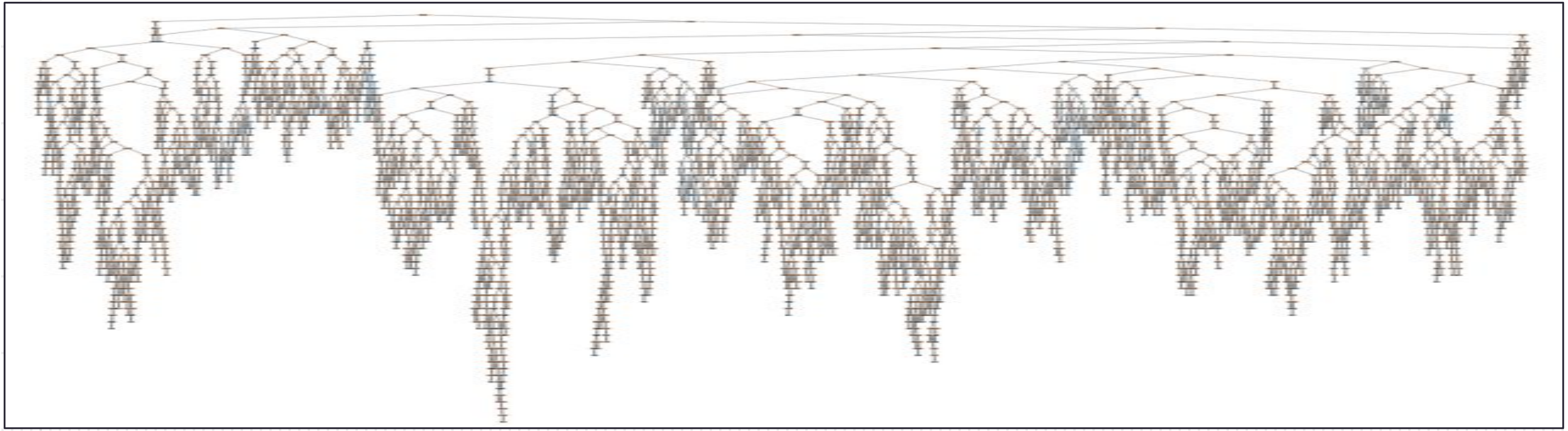
Principal Component Analysis is a powerful technique for dimensionality reduction, feature extraction, and data exploration. By transforming high-dimensional data into a lower-dimensional representation, PCA enables easier analysis and visualization while preserving the essential information. Thus, after running PCA the algorithm generated 5 new features instead of the 8 we had.

	0	1	2	3	4
0	-1.743298	0.868154	-0.526446	0.362759	0.493620
1	-1.743296	0.868155	-0.526445	0.362760	0.493620
2	-1.624446	0.712337	-0.809120	0.126664	0.222745
3	-1.505599	0.556518	-1.091795	-0.109432	-0.048129
4	-1.743291	0.868157	-0.526444	0.362760	0.493619
...	...	...	...	...	...
6362615	3.022748	0.462747	0.436663	1.880768	-1.279750
6362616	2.659810	1.468086	1.361611	1.519868	-1.696408
6362617	3.378394	-0.433167	-0.023285	2.296516	-0.500476
6362618	2.659812	1.468087	1.361611	1.519869	-1.696408
6362619	3.290339	-0.453034	-0.418475	2.436026	-0.176207

6362620 rows × 5 columns

Decision Tree

Decision trees are versatile and intuitive machine-learning algorithms used for both classification and regression tasks. They offer interpretability, handle nonlinear relationships, and provide feature importance rankings. However, decision trees are prone to overfitting and may not perform well on unseen data.

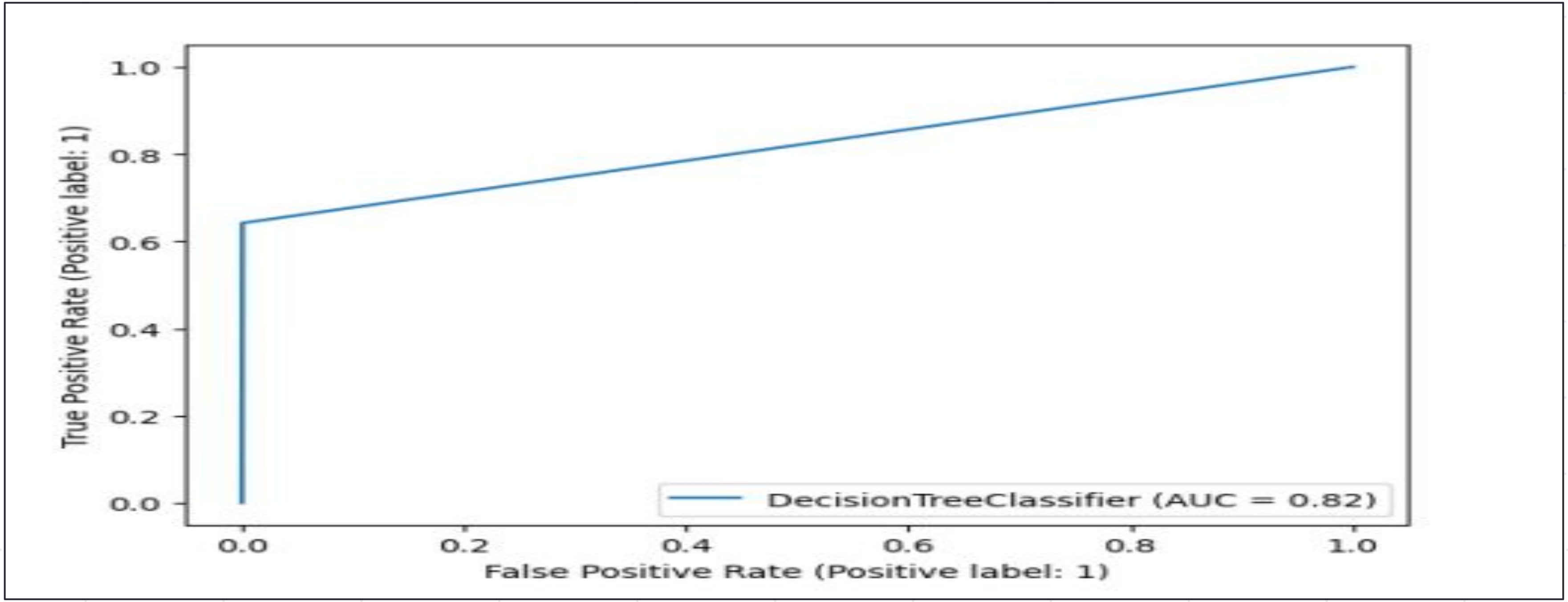


```
array([[1905426,     912],
       [    890,    1558]])
```

```
accuracy: 0.9990559444589389
```

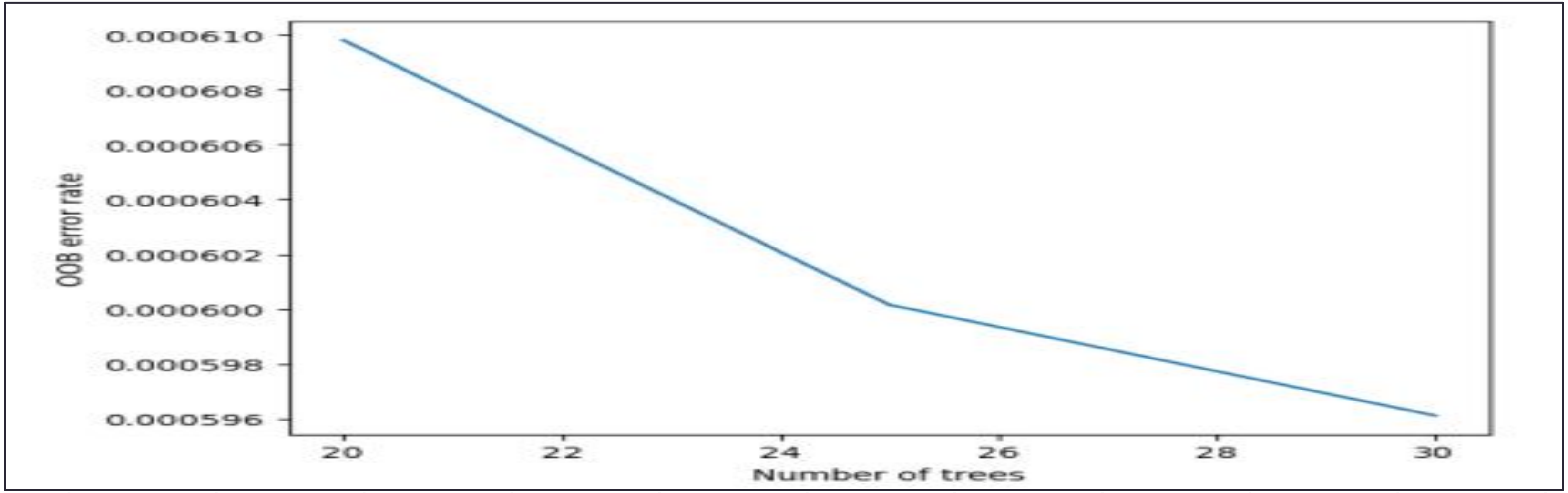
Receiver Operating Characteristic (ROC):

The Receiver Operating Characteristic (ROC) graph is a graphical representation that illustrates the performance of a binary classification model at various classification thresholds. It plots the true positive rate (TPR), also known as sensitivity or recall, on the y-axis against the false positive rate (FPR) on the x-axis. For the decision tree, our AUC is only 0.82.



Random Forest:

Random Forest is a powerful machine learning algorithm that combines the strength of decision trees and ensemble learning. It provides robust predictions, handles high dimensional data, and offers insights into feature importance. Random Forest has numerous applications across various domains and is a go-to choice for many data scientists and machine learning practitioners.

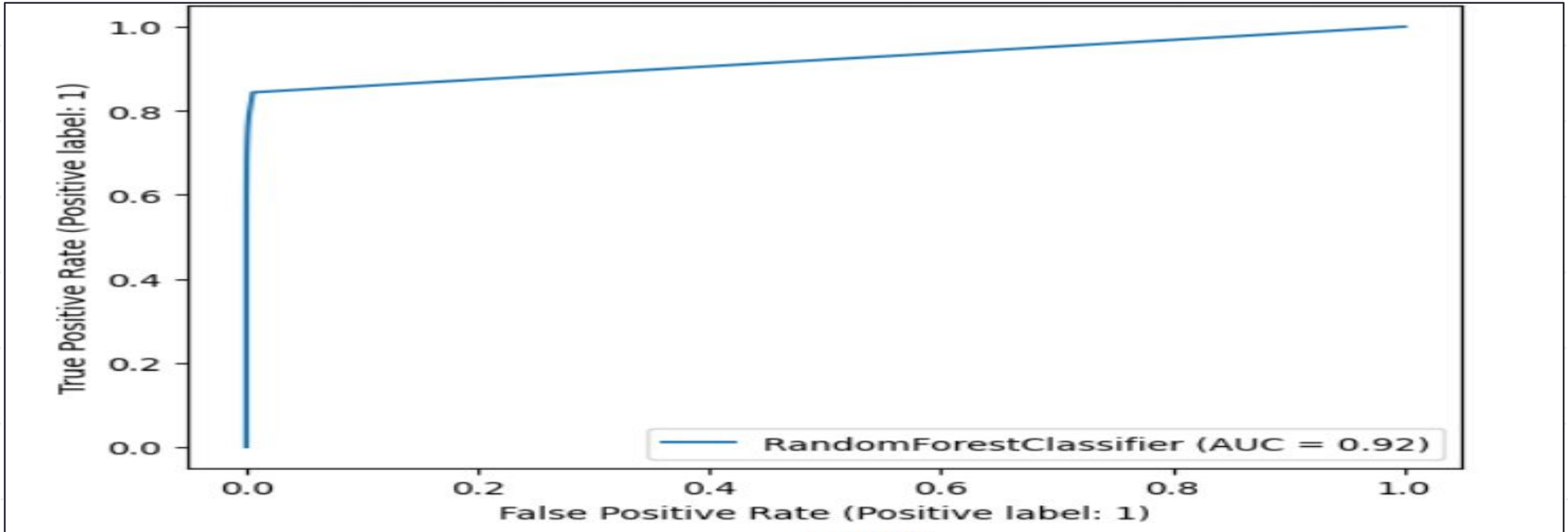



```
accuracy: 0.9994038095417715
```

```
array([[1906173,     165],
       [    973,    1475]])
```

Random Forest False Positive Rate:

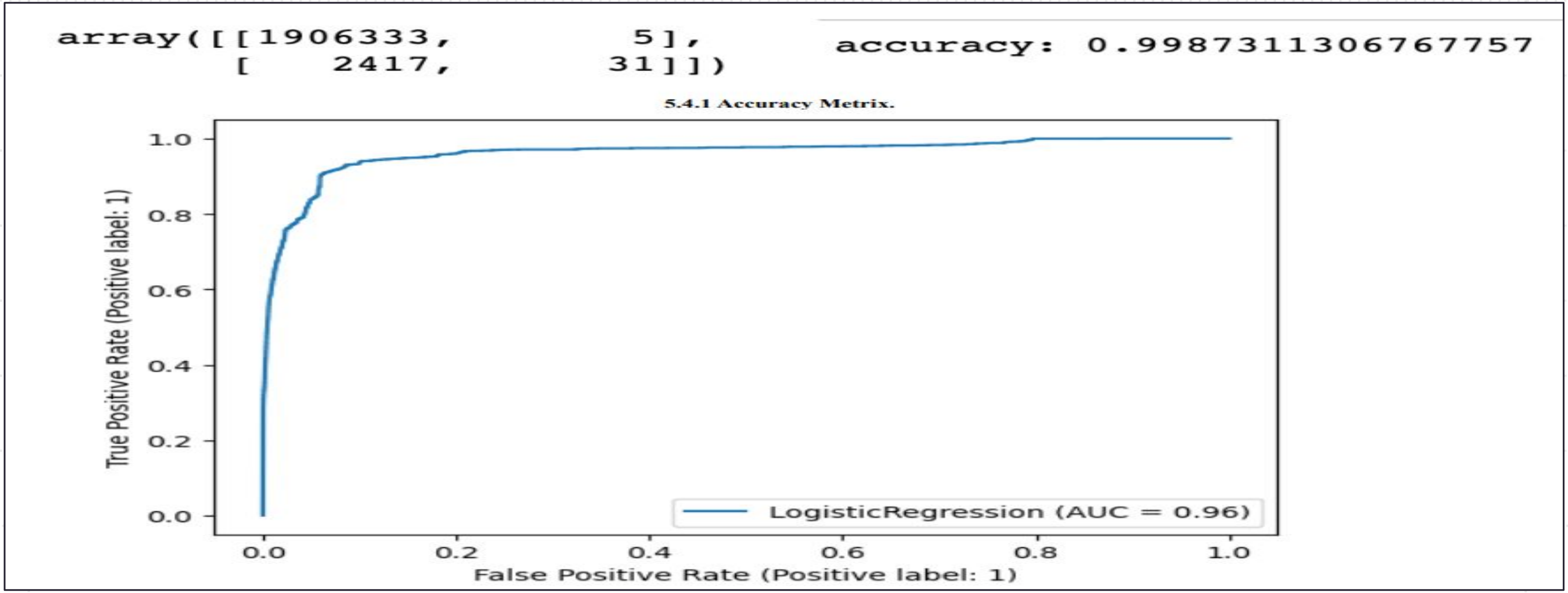
Upon running the algorithm, we obtained an exceptional accuracy rate of 99.9%. Nevertheless, there is a slight drawback associated with this outcome. We have come across a notable issue concerning false positives, specifically 165 instances, where our model mistakenly classifies negative values (fraudulent transactions) as positive values (valid transactions). It is imperative that we tackle this problem promptly and investigate strategies to further mitigate the occurrence of these errors.



5.3.3 ROC

Logistic Regression

Logistic regression is a widely used algorithm for binary classification tasks. It models the relationship between features and the probability of the positive class using the logistic function. Logistic regression provides interpretable coefficients, enables understanding of the impact of features, and offers flexibility in adjusting the decision boundary. It is a valuable tool in predictive modelling and understanding the factors influencing binary outcomes.



Support Vector Machine(SVM):

We preprocess the dataset by scaling the numeric features. We used the StandardScaler() function from scikit-learn to scale the numeric variables.

We Split the dataset into training and testing sets (70:30) using the train_test_split() function from scikit-learn.

Then we train the SVM model using the SVC () class and later we evaluate the performance of the model using appropriate evaluation metrics, such as accuracy, precision, recall, and F1 score.

```
train, test = train_test_split(df_raw, test_size=0.2, random_state=42)
df = test
print(df.columns)
```

```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

Select the columns of interest and create a new dataframe:

```
cols = ['step', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest', 'isFraud']

df_new = df[cols]
```

```
from sklearn.svm import SVC

clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	381286
1	0.99	0.32	0.48	472
accuracy			1.00	381758
macro avg	0.99	0.66	0.74	381758
weighted avg	1.00	1.00	1.00	381758

Fig: Model Performance

Conclusion

Our findings indicate that the algorithms used, including decision trees, random forests, and logistic regression, achieved exceptional accuracy rates of 99.9%. However, we have also encountered challenges such as a relatively high number of false positives and false negatives. It is crucial to address these issues to improve the overall performance of the models and minimize errors in fraud detection.

This project contributes to the ongoing efforts in developing robust fraud detection systems and provides valuable insights for individuals and organizations involved in online transactions. By understanding the underlying patterns and behaviours of fraudsters, we can work towards creating a safer and more secure online environment.

Approach	Accuracy Rate
Random Forest	99.94%
Decision Tree	99.90%
Logistic Regression	99.87%
SVM	100%

Thank You!

```python
def gratitude():
   print("Thank you.")
```