CREDIT CARD FRAUD DETECTION

CSE 587

Phase 2

Team

Sri Harshitha Palla	50469733	spalla3
Venkata Subrahmanyam	50465653	vupputur

Problem Statement

With the increase in online transactions and e-commerce platforms, credit card fraud has become more common than before. The main objective of this project is to accurately detect fraudulent credit card transactions by using of various Machine Learning Algorithms. We will be using the Credit Card Transactions Fraud Detection Dataset from Kaggle[1] to achieve this.

A. DISCUSS THE BACKGROUND OF THE PROBLEM LEADING TO YOUR OBJECTIVES. WHY IS IT A SIGNIFICANT PROBLEM?

Credit card fraud can result in significant financial losses for both cardholders and financial institutions. Fraudulent transactions can be very large, and often go unnoticed until the victim receives their credit card statement. By detecting and preventing fraud, financial institutions can save themselves and their customers from financial losses. Overall, credit card fraud detection is significant because it helps financial institutions to protect themselves and their customers from financial losses, maintain their reputation, comply with laws and regulations, and provide a better customer experience.

B. EXPLAIN THE POTENTIAL OF YOUR PROJECT TO CONTRIBUTE TO YOUR PROBLEM DOMAIN. DISCUSS WHY THIS CONTRIBUTION IS CRUCIAL?

Financial institutions process large volumes of credit card transactions every day, and data science techniques can help them to quickly and accurately analyse this data to detect and prevent fraud. This project uses machine learning algorithms that can identify patterns and anomalies in credit card transactions that may be indicative of fraudulent activity. These techniques can help financial institutions to identify fraudulent transactions more accurately and efficiently than traditional manual methods.

Feature Encoding

One hot encoding

Since we have a lot of categorical data in our dataset, we will begin with converting them into numerical data by one hot encoding[2]. This is done to make the processing easier when applying machine learning algorithms to the dataset for fraud detection.

One hot encoding works by representing each category as a binary vector, where only one element is 1 and all others are 0s. This way, each category is represented as a unique numerical vector, which can be used as input for machine learning algorithms.

Here, we are applying One hot encoding to columns – category, gender, day

```
In [74]: # one hot encoding for - category, gender, day, age

ohe_category = pd.get_dummies(data.category, prefix='category', drop_first=True)

ohe_gender = pd.get_dummies(data.gender, prefix='gender', drop_first=True)

ohe_day_of_week = pd.get_dummies(data.day, prefix='day', drop_first=True)
```

The columns now are:

```
In [76]: data_new.columns

Out[76]: Index(['cc_num', 'merchant', 'category', 'amt', 'gender', 'street', 'city',
                'state', 'zip', 'lat', 'long', 'city_pop', 'job', 'trans_num',
                'unix_time', 'merch_lat', 'merch_long', 'is_fraud', 'label', 'hour',
                'day', 'year-month', 'age', 'category_food_dining',
                'category_gas_transport', 'category_grocery_net',
                'category_grocery_pos', 'category_health_fitness', 'category_home',
                'category_kids_pets', 'category_misc_net', 'category_misc_pos',
                'category_personal_care', 'category_shopping_net',
                'category_shopping_pos', 'category_travel', 'gender_m', 'day_Mon',
                'day_Sat', 'day_Sun', 'day_Thu', 'day_Tue', 'day_Wed'],
               dtype='object')
```

Since columns like merchant, street, city, state, job have a lot of unique values, it is difficult to apply one hot encoding. Hence, we are going to drop them.

In [78]:
```
data_new.drop(['merchant','street','city','state','job', 'category','gender','day',],
              axis=1, inplace=True)
```

In [79]:
```
data_new.columns
```

Out[79]:
```
Index(['cc_num', 'amt', 'zip', 'lat', 'long', 'city_pop', 'trans_num',
       'unix_time', 'merch_lat', 'merch_long', 'is_fraud', 'label', 'hour',
       'year-month', 'age', 'category_food_dining', 'category_gas_transport',
       'category_grocery_net', 'category_grocery_pos',
       'category_health_fitness', 'category_home', 'category_kids_pets',
       'category_misc_net', 'category_misc_pos', 'category_personal_care',
       'category_shopping_net', 'category_shopping_pos', 'category_travel',
       'gender_m', 'day_Mon', 'day_Sat', 'day_Sun', 'day_Thu', 'day_Tue',
       'day_Wed'],
      dtype='object')
```

1. Logistic Regression

Algorithm [3]

Collecting numerical data

```
In [81]: data_lr = data_new.select_dtypes(include='number')
```

```
In [84]: data_lr.columns
```

```
Out[84]: Index(['cc_num', 'amt', 'zip', 'lat', 'long', 'city_pop', 'unix_time',
                'merch_lat', 'merch_long', 'is_fraud', 'hour', 'age',
                'category_food_dining', 'category_gas_transport',
                'category_grocery_net', 'category_grocery_pos',
                'category_health_fitness', 'category_home', 'category_kids_pets',
                'category_misc_net', 'category_misc_pos', 'category_personal_care',
                'category_shopping_net', 'category_shopping_pos', 'category_travel',
                'gender_m', 'day_Mon', 'day_Sat', 'day_Sun', 'day_Thu', 'day_Tue',
                'day_Wed'],
               dtype='object')
```

Removing less significant columns

```
In [85]: data_lr.drop(['zip', 'lat', 'long', 'city_pop', 'unix_time', 'merch_lat','merch_long'],axis=1, inplace=True)
```

```
In [86]: data_lr
```

Out[86]:

	cc_num	amt	is_fraud	hour	age	category_food_dining	category_gas_transport	category_grocery_net	category_grocery_pos	category_health_f
0	2.703190e+15	4.97	0	0	31	0	0	0	0	
1	6.304230e+11	107.23	0	0	41	0	0	0	1	
2	3.885950e+13	220.11	0	0	57	0	0	0	0	
3	3.534090e+15	45.00	0	0	52	0	1	0	0	
4	3.755340e+14	41.96	0	0	33	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
1604289	3.056060e+13	43.77	0	23	55	0	0	0	0	
1604290	3.556610e+15	111.84	0	23	21	0	0	0	0	
1604291	6.011720e+15	86.88	0	23	39	0	0	0	0	
1604292	4.079770e+12	7.99	0	23	55	0	0	0	0	
1604293	4.170690e+15	38.13	0	23	28	0	0	0	0	

1604223 rows × 25 columns

Feature extraction

```
In [230]: #feature selection

          target = data_lr.is_fraud.values
          features = data_lr.drop(['is_fraud'],axis=1).values
```

We use feature scaling to transform the features of our dataset so that they have similar scales or ranges. StandardScaler[4] is a popular method of feature scaling that transforms the data to have a mean of 0 and a standard deviation of 1.

In [238]:
```
#feature scaling

from sklearn.preprocessing import StandardScaler

features = StandardScaler().fit_transform(features)
```

Test, Train, Split

In [336]:
```
#train, test, split


stratified_kf = StratifiedKFold(n_splits=3)

stratified_kf.get_n_splits(features,target)

for i, j in stratified_kf.split(features,target):
    features_train, features_test = features[i], features[j]
    target_train, target_test = target[i], target[j]
```

Code reference[15]

Here,

- StratifiedKFold[5] is a specific type of cross-validation technique that is used when dealing with imbalanced datasets.
- We have a highly imbalanced dataset with too few fraud transactions and too many non-fraud transactions.
- This imbalance creates a bias to the model building process.
- In StratifiedKFold, the dataset is divided into k folds, with each fold having roughly the same proportion of instances from each class as the entire dataset.
- This ensures that the class distribution is preserved in each fold, which can help prevent bias in the evaluation of the model.

Prediction

In [337]:
```
#Prediction

lr = LogisticRegression(random_state=22)

model = lr.fit(features_train, target_train)

target_train_pred = model.predict(features_train)
target_test_pred = model.predict(features_test)
```

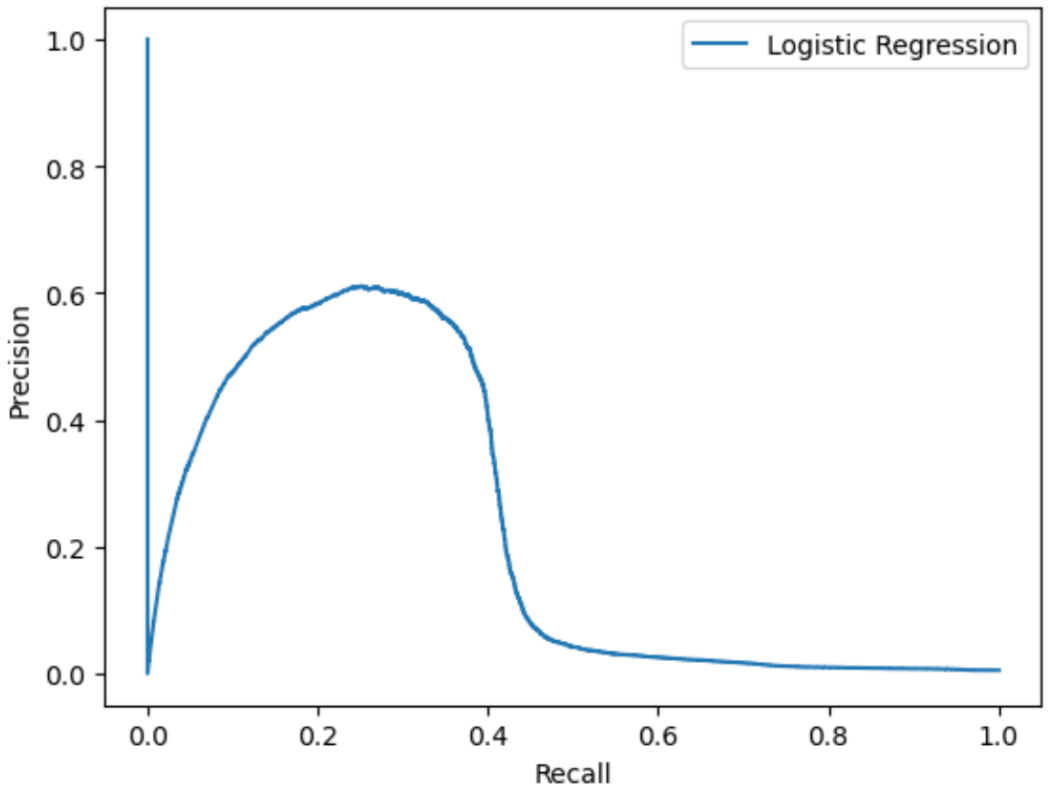
In [338]:
```
print('target_train_pred: ',target_train_pred)
print('target_test_pred: ', target_test_pred)
```

```
target_train_pred:  [0 0 0 ... 0 0 0]
target_test_pred:  [0 0 0 ... 0 0 0]
```

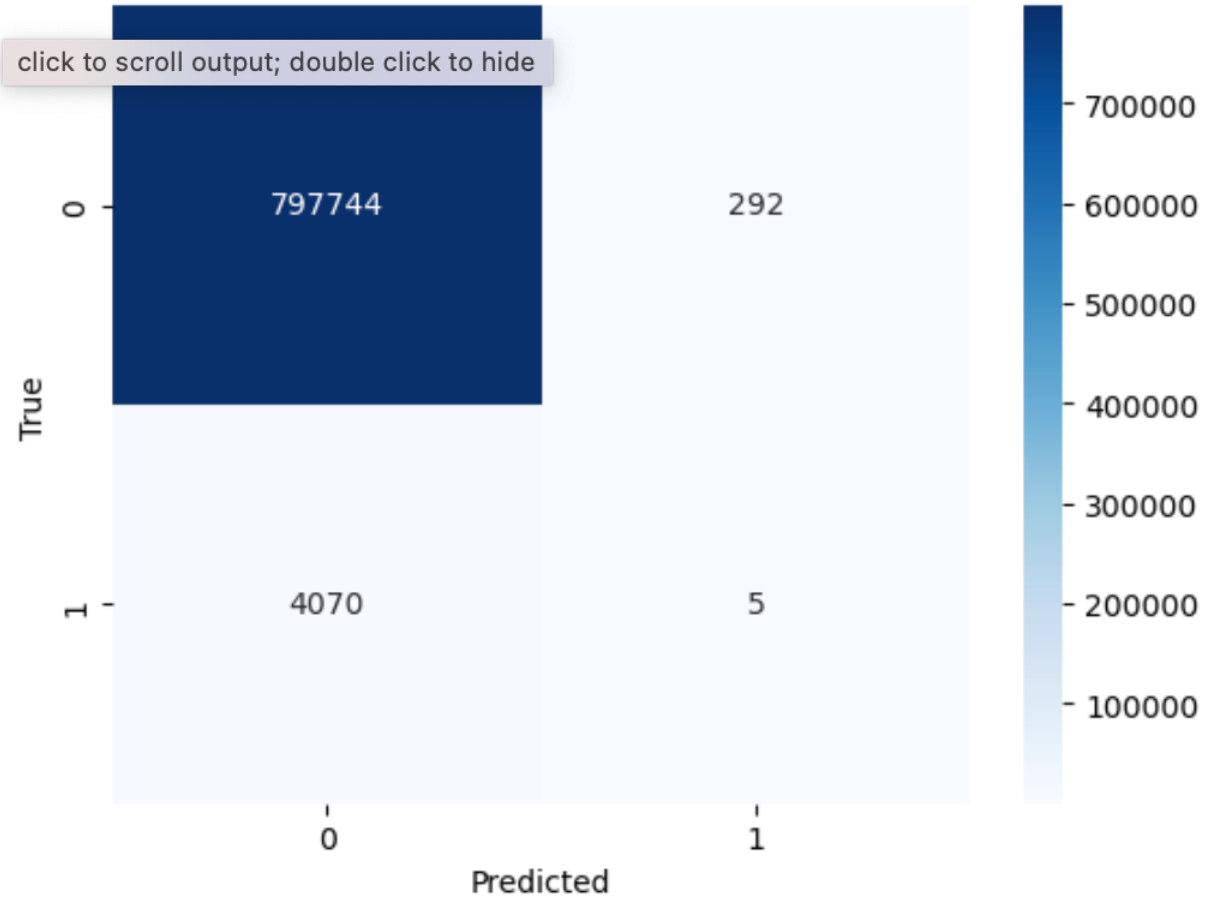
Visualisation

Precision-Recall curve [6]

Precision-recall curve can be useful for evaluating the performance of a model on imbalanced datasets, where the positive class is much less frequent than the negative class.



Confusion matrix [7]



Explanation & Analysis

Why Logistic Regression?

Logistic regression is a commonly used algorithm in machine learning for binary classification problems, where the goal is to predict one of two possible outcomes. In the case of credit card fraud detection, the goal is to predict whether a credit card transaction is fraudulent or not, which is a binary classification problem.

Model Evaluation: [8]

```
In [339]: #evaluating the model

          #Accuracy, Preciision, Recall, F1 score,
          accuracy = accuracy_score(target_test,target_test_pred)
          precision = precision_score(target_test, target_test_pred)
          recall = recall_score(target_test,target_test_pred)
          f1 = f1_score(target_test, target_test_pred)


          df = pd.DataFrame([[accuracy, precision, recall, f1]],
                                    columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
          df
```

Observations:

	Accuracy	Precision	Recall	F1 Score
0	0.994549	0.037383	0.002944	0.005459

In Logistic regression for fraud detection the effectiveness of the model is as follows,

- An accuracy of 0.994549
- Precision of 0.037383
- Recall of 0.002944
- F1 score of 0.005459

2. Decision Tree

Algorithm [9,10]

Test, Train, Split

```
In [342]: #test train split

features_train, features_test, target_train, target_test = train_test_split(features,target,
                                                                            test_size=0.2, random_state=26)
```

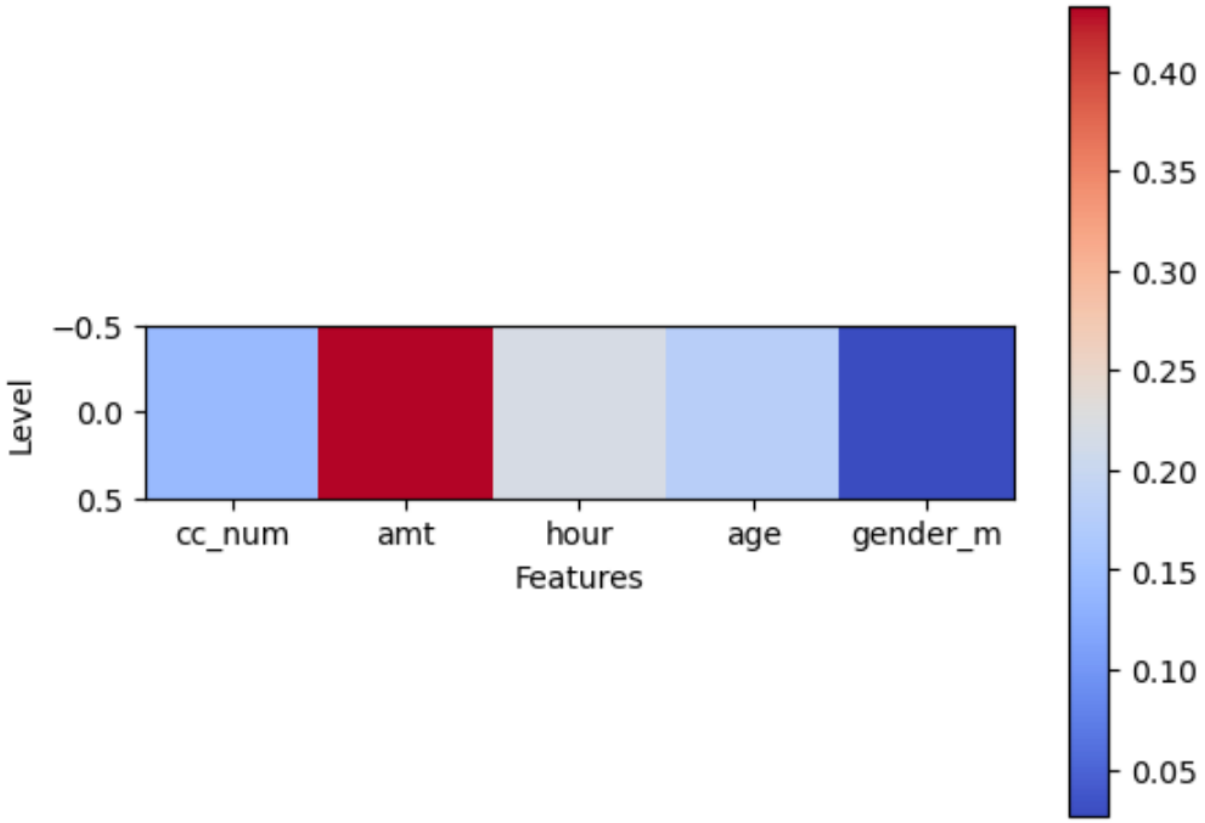
Prediction

```
In [343]: #prediction

dt = DecisionTreeClassifier(max_depth=12)
model = dt.fit(features_train,target_train)
target_test_pred = model.predict(features_test)
print(target_test_pred)
```

```
[0 0 0 ... 0 0 0]
```

Visualisation

Decision Tree heatmap [11]



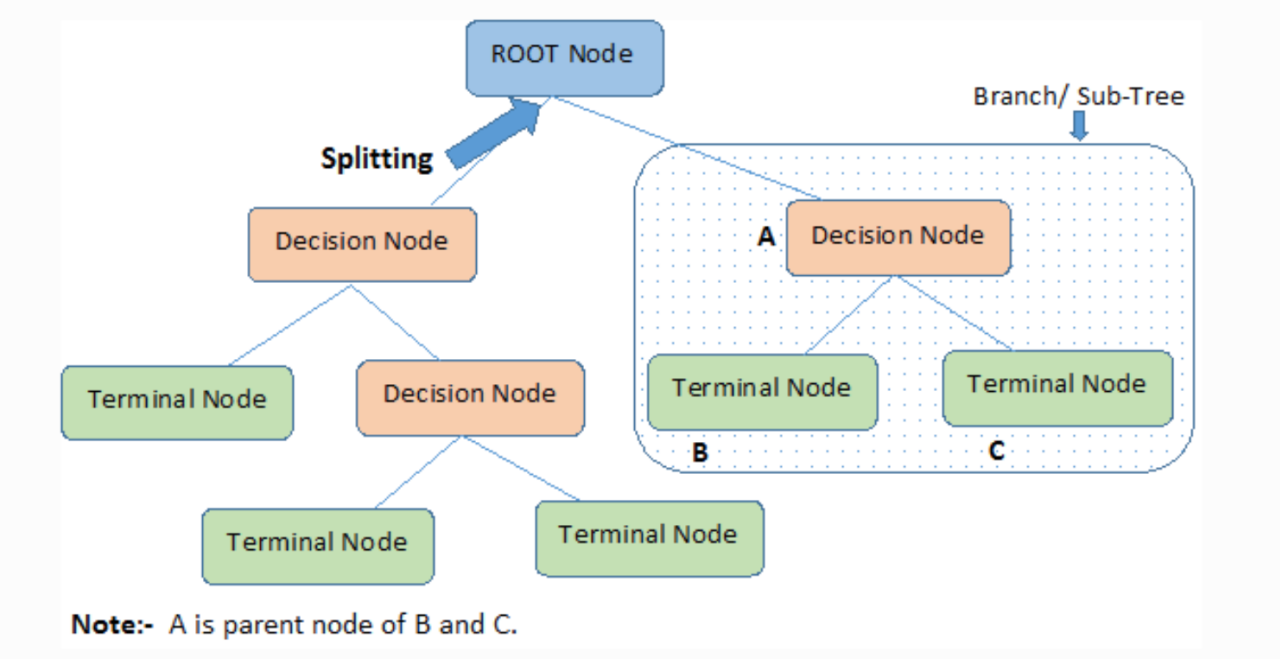
This heatmap can be used to visualize the importance of each feature in the decision tree. Each x axis feature in the heatmap corresponds to a feature, and each column corresponds to a level of the decision tree. The colour of each cell in the heatmap indicates the importance of the corresponding feature at the corresponding level.

Most importance – amt
Least importance - gender

Explanation & Analysis

What is Decision Tree Algorithm? [12,13]

The decision tree algorithm is a popular machine learning algorithm that is used for classification tasks. It works by recursively splitting the data into subsets based on the value of a certain attribute, and then making decisions based on the resulting subsets. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.



Why Decision Tree Algorithm?

- Decision tree algorithm handles large, complex datasets like ours
- It can effectively capture non-linear relationships between input features and the target variable
- By constructing a decision tree based on the features of our input, the algorithm can effectively classify transactions as either fraudulent or legitimate

Model Evaluation [14]

```
In [344]: #evaluation

          #Accuracy, Preciision, Recall, F1 score,
          accuracy = accuracy_score(target_test,target_test_pred)
          precision = precision_score(target_test, target_test_pred)
          recall = recall_score(target_test,target_test_pred)
          f1 = f1_score(target_test, target_test_pred)


          df = pd.DataFrame([[accuracy, precision, recall, f1]],
                                    columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
          df
```

Observations

	Accuracy	Precision	Recall	F1 Score
0	0.998632	0.918239	0.803178	0.856863

In Decision Tree Algorithm for fraud detection the effectiveness of the model can be observed from,

- An accuracy of 0.998632
- Precision of 0.918239
- Recall of 0.803178
- F1 score of 0.856863

As we can see the precision from Decision Tree Algorithm is much better than Logistic regression. So this model fits better for our dataset.

3. Random Forest Algorithm

Test, train, split

```
In [349]: # test train split

features_train, features_test, target_train, target_test = train_test_split(features,target,
                                                                             test_size=0.2, random_state=34)
```

Model [16]

```
In [350]: #model

rf = RandomForestClassifier(n_estimators=70,
                            random_state=34,
                            n_jobs=-1)
model = rf.fit(features_train,target_train)
```

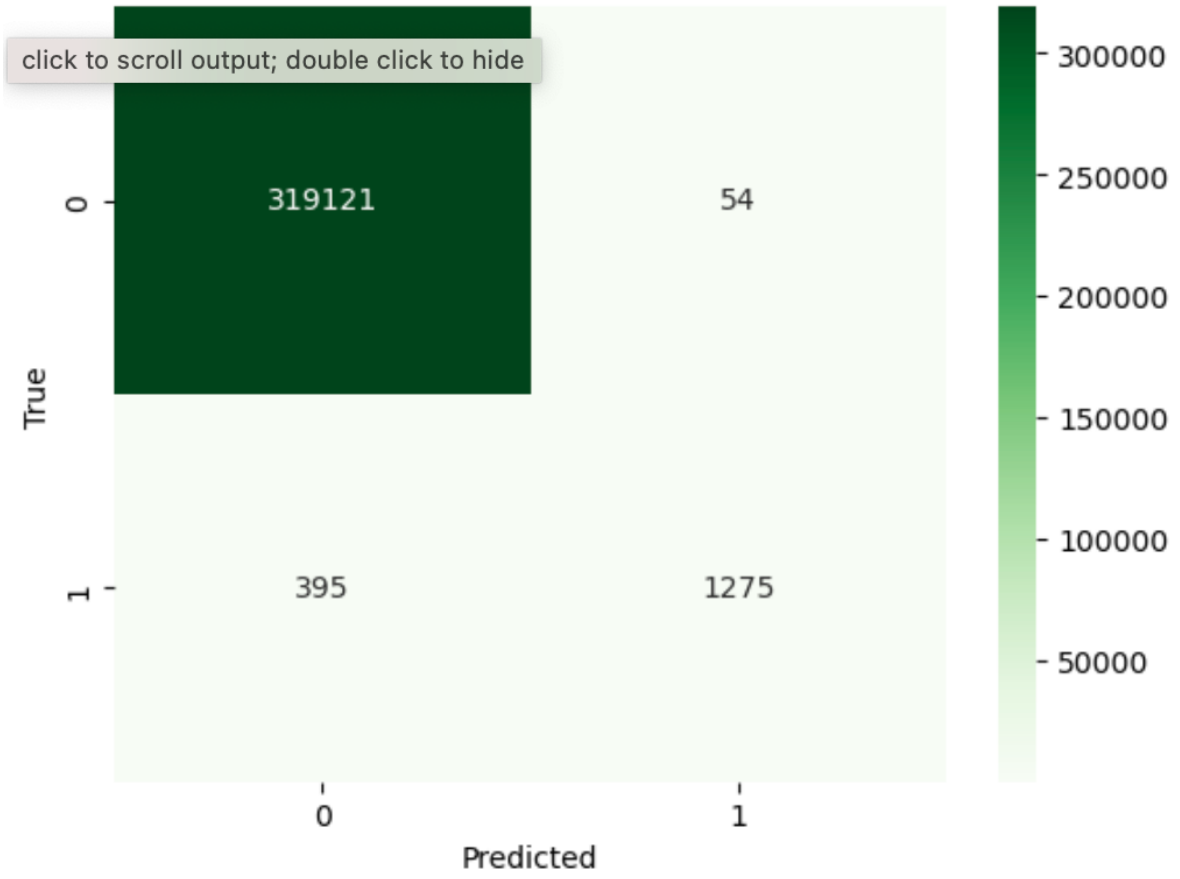
Prediction

```
In [352]: #prediction

target_test_pred = model.predict(features_test)
print(target_test_pred)
```

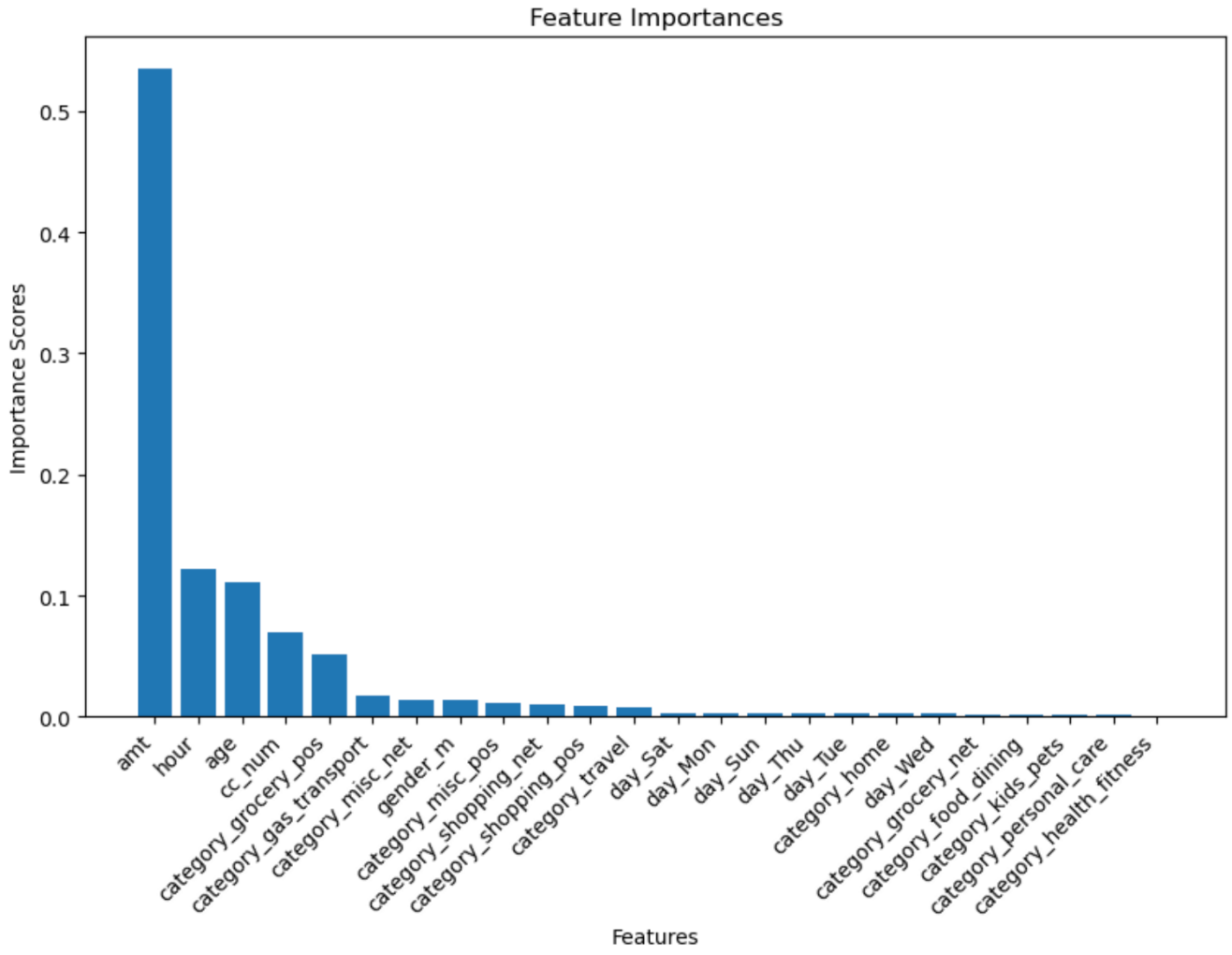
```
[0 0 0 ... 0 0 0]
```

Visualisation

Confusion Matrix



Feature Importance Plot [17]



This figure illustrates the relative importance of each feature of our model.
Therefore, we can conclude that:

- Amount is the most important feature of our dataset
- Followed by hour
- Then comes age
- In terms of category, grocery is an important feature
- Followed by gas
- In terms of day of the week, Saturday has the most importance
- Followed by Monday

Explanation & Analysis

What is Random Forest?

Random forest[18], like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model's prediction (see figure below). For classification tasks, the output of the random forest is the class selected by most trees.

Why Random Forest?

- It is a popular machine learning algorithm for credit card fraud detection because it is an ensemble method that combines multiple decision trees to improve the accuracy and reduce the risk of overfitting
- It is a non-parametric model, which means it can capture complex relationships and interactions between features without making assumptions about the underlying data distribution. This is important for credit card fraud detection because fraudulent transactions can take many different forms and exhibit unusual patterns that may not be captured by a linear model
- Resistance to overfitting, which can be a problem when dealing with imbalanced datasets like ours

Model Evaluation

In [353]:
```
#evaluating the model

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                          columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Observations

	Accuracy	Precision	Recall	F1 Score
0	0.998604	0.962617	0.755501	0.846575

In Random Forest for fraud detection the effectiveness of the model can be observed from,

- An accuracy of 0.998604
- Precision of 0.962617
- Recall of 0.755501
- F1 score of 0.846575

As observed, the precision from Random Forest is much better than Logistic regression and Decision Tree. Accuracy of Random Forest and Decision tree is almost similar. So this model fits better for our dataset than Logistic Regression.

4. Naïve Bayes

Algorithm

Test – train – split

```
In [356]: # test train split

features_train, features_test, target_train, target_test = train_test_split(features,target,
                                                                             test_size=0.2,random_state=27,
                                                                             stratify=target)
```

Model [19]

```
In [357]: #model

nb = GaussianNB()
nb.fit(features_train,target_train)
```

```
Out[357]: GaussianNB()
```

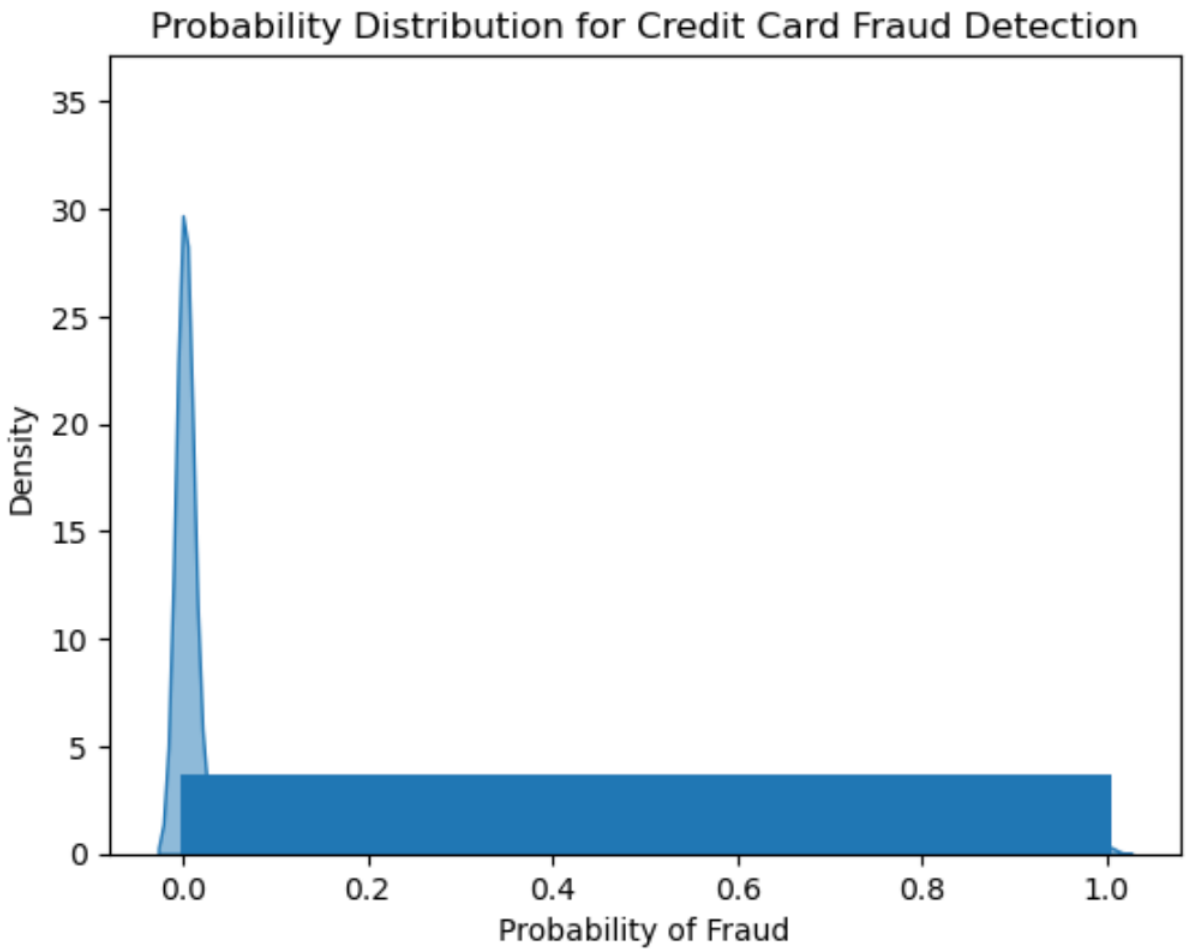
Prediction

```
In [358]: #prediction

target_test_pred = nb.predict(features_test)
target_pred_prob = nb.predict_proba(features_test)
print('target_train_pred: ',target_train_pred)
print('target_test_pred: ', target_test_pred)
```

```
target_train_pred:  [0 0 0 ... 0 0 0]
target_test_pred:  [0 0 0 ... 0 0 0]
```

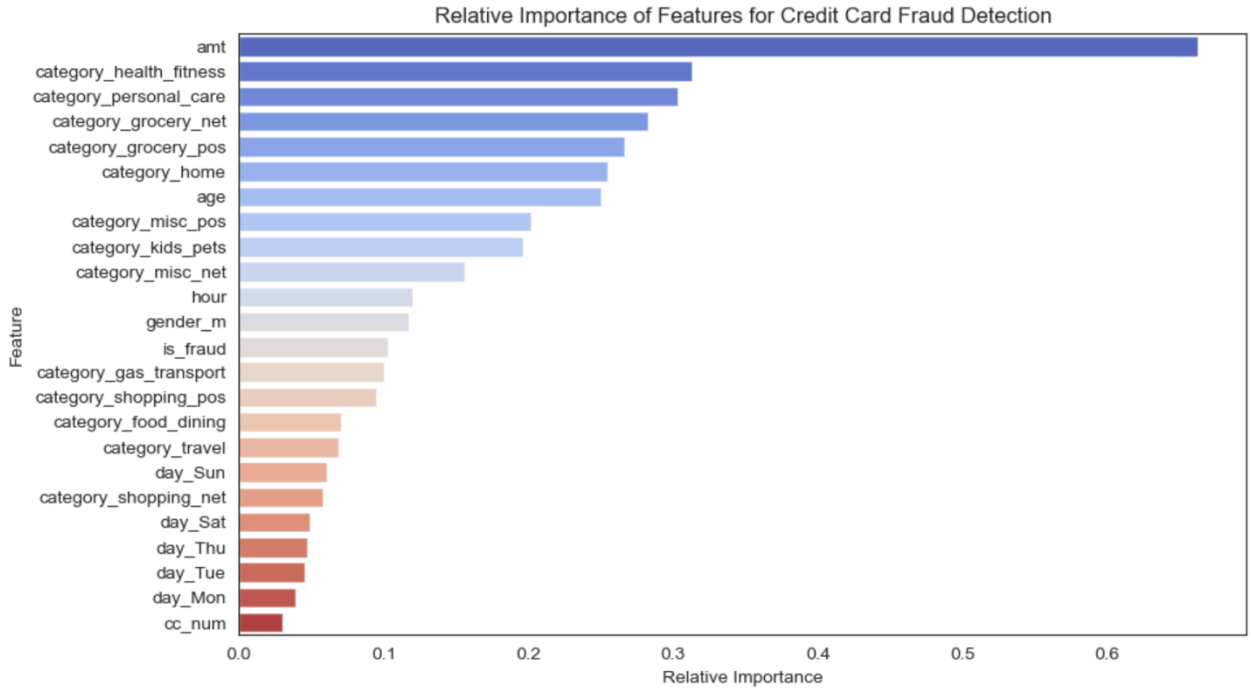
Visualisation

Probability Distribution plot



- This probability distribution plot that uses a KDE plot[20] and a rug plot to show the distribution of probabilities
- The KDE plot estimates the probability density function of the probabilities
- The rug plot[21] shows each individual probability estimate as a tick mark on the x-axis.

Relative importance graph [22],[23]



According to this graph:

- Amt has the most importance
- Heath and fitness category has the most importance among all categories
- Age has more importance than gender
- Hour oof transaction has more importance than gender
- Sunday has more importance than other days
- Cc_num has the least importance

Explanation & Analysis

Why Naïve Bayes?

- It is a simple, effective and efficient algorithm for our problem statement
- It is well-suited for to work with high-dimensional data, such as our credit card transactions dataset, and can handle both categorical and continuous features
- Easy way to detect rare events, such as fraudulent transactions, because it makes predictions based on probabilities
- This means that it can identify transactions that have a high probability of being fraudulent, even if there are only a few instances of fraud in the dataset like ours
- It is also relatively computationally inexpensive, which is important when dealing with large datasets

Model Evaluation

In [359]:
```
# evalluation metrics

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                           columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Observation

	Accuracy	Precision	Recall	F1 Score
0	0.913503	0.039295	0.683436	0.074316

For this model,

- Accuracy obtained is 0.913503
- Precision obtained is 0.039295
- Recall obtained is 0.683436
- F1 score obtained is 0.074316

5. K Means

Algorithm

Test – train – split

In [366]:
```
#train test split

features_train, features_test, target_train, target_test = train_test_split(features,target, test_size=0.2)
features_train = normalize(features_train)
features_test = normalize(features_test)
```

Model [24]

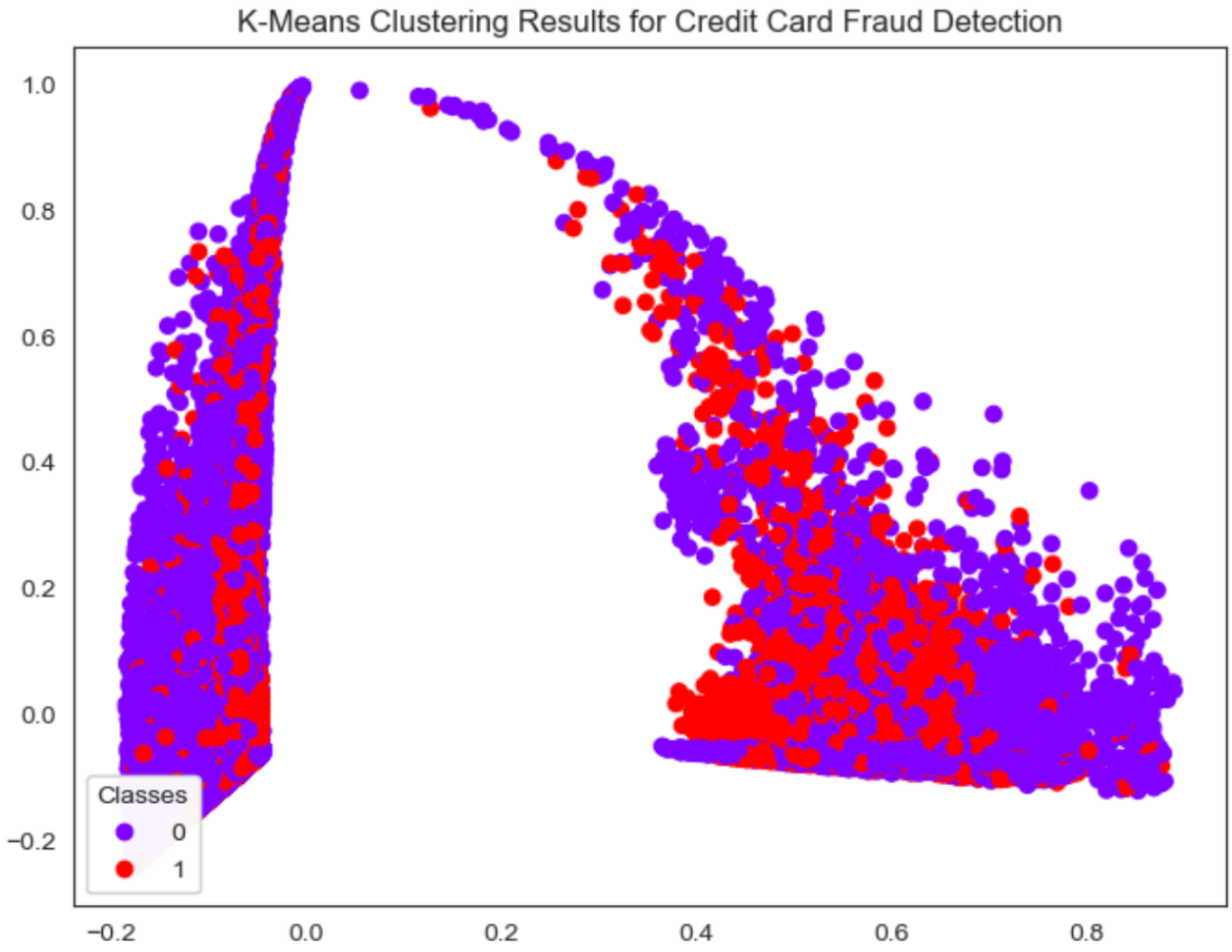
In [367]:
```
#model

km = KMeans(n_clusters=2,random_state=0,algorithm="elkan",max_iter=10000)
km.fit(features_train)
```

Out[367]:
```
KMeans(algorithm='elkan', max_iter=10000, n_clusters=2, random_state=0)
```

Prediction

In [369]:
```
#prediction

traget_train_pred = km.predict(features_train)

#confusion matrix between the predicted cluster labels and the true target labels for the training set

tn, fp, fn, tp=confusion_matrix(target_train, traget_train_pred).ravel()

''' if the number of correctly classified samples is less than the number of incorrectly classified samples
the predicted cluster labels are opposite to the true target labels '''

flg = 0
if tn+tp < fn+fp:
    flg = 1

target_test_pred = km.predict(features_test)

if flg:
    target_test_pred = 1 - target_test_pred

tn, fp, fn, tp=confusion_matrix(target_test, target_test_pred).ravel()
```

Visualisation



- This plot [26] shows a scatter plot of the first two principal components of the dataset
- The two axes correspond to the two components that capture the maximum amount of variance in the data.
- The plot suggests that the k-means algorithm is not able to completely separate the two classes.
- We can see some non-fraudulent transactions (purple) that are clustered together with fraudulent transactions (red).
- This could be due to the highly imbalanced nature of the dataset, where fraud transactions are much less frequent than non-fraudulent transactions.
- In such cases, clustering algorithms may not perform well as they tend to be biased towards the majority class.
- However, we can still see that most of the fraudulent transactions are clustered together in a dense region (bottom right), which indicates that the k-means algorithm is able to identify at least some of the fraudulent transactions.

Explanation & Analysis

What is K-Means?

Clustering models aim to group data into distinct “clusters” or groups. K-means [25] algorithm identifies *k* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The *‘means’* in the K-means refers to averaging of the data and finding the centroid.

Why K-Means?

K-means clustering can be used for credit card fraud detection as it can identify patterns and groups in the data that may indicate fraudulent activity. It can help to separate the normal transactions from the fraudulent transactions based on their similarities or differences in their features. But, K-means clustering may not always perform well in detecting credit card fraud as it assumes that the clusters are spherical and have equal variance.

Model Evaluation

In [370]:
```
# evaluation

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                            columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Observation

	Accuracy	Precision	Recall	F1 Score
0	0.6082	0.005595	0.424939	0.011045

For this model,

- Accuracy obtained is 0.6082
- Precision obtained is 0.005595
- Recall obtained is 0.424939
- F1 score obtained is 0.011045

Here, we can observe a low accuracy and precision. This might be because:

- K-means clustering may not always perform well in detecting credit card fraud as it assumes that the clusters are spherical and have equal variance
- Since K-Means is an unsupervised learning algorithm, it does not take into account the class labels during training
- It just tries to find the best partition of the data points into distinct clusters
- This may lead to improper capturing of underlying structure of the data that is specific to the fraudulent transactions
- K-means is highly sensitive to initial cluster centers
- This may lead to high number of false positives and false negatives in the classification task
- Also highly unbalanced dataset like our might influence the algorithm and the results of prediction

References

[1] https://www.kaggle.com/datasets/kartik2112/fraud-detection
[2] https://stackabuse.com/one-hot-encoding-in-python-with-pandas-and-scikit-learn/
[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[4] https://www.digitalocean.com/community/tutorials/standardscaler-function-in-python
[5] https://stackoverflow.com/questions/65600245/scikit-learn-stratifiedkfold-implementation
[6] https://www.programcreek.com/python/example/89259/sklearn.metrics.precision_recall_curve
[7] https://www.w3schools.com/python/python_ml_confusion_matrix.asp
[8] https://coderzcolumn.com/tutorials/machine-learning/model-evaluation-scoring-metrics-scikit-learn-sklearn#:~:text=Scikitlearn%20has%20a%20function%20named%20%27accuracy_score%20%28%29%27%20that,their%20score%20%28%29%20methods%20to%20evaluate%20model%20performance
[9]https://scikit-learn.org/stable/modules/tree.html
[10]https://machinelearningknowledge.ai/decision-tree-classifier-in-python-sklearn-with-example/
[11]https://stackoverflow.com/questions/56302647/how-to-plot-a-heatmap-and-find-best-hyperparameter-for-decision-tree-after-grids
[12]https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
[13]https://www.ibm.com/topics/decision-trees
[14]https://gist.github.com/pb111/af439e4affb1dd94879579cfd6793770
[15] https://www.analyseup.com/python-machine-learning/stratified-kfold.html
[16] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[17] https://stackoverflow.com/questions/44511636/plot-feature-importance-with-feature-names
[18] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[19] https://www.analyticsvidhya.com/blog/2021/11/implementation-of-gaussian-naive-bayes-in-python-sklearn/
[20]https://seaborn.pydata.org/generated/seaborn.kdeplot.html#:~:text=A%20kernel%20density%20estimate%20(KDE,in%20one%20or%20more%20dimensions
[21] https://seaborn.pydata.org/generated/seaborn.rugplot.html
[22] https://seaborn.pydata.org/generated/seaborn.barplot.html
[23] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[24] https://www.kaggle.com/code/isaikumar/credit-card-fraud-detection-using-k-means-and-knn
[25] https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
[26] https://www.askpython.com/python/examples/plot-k-means-clusters-python