Project : Credit Card Fraud Detection

Problem Statement:

1. A credit card is a small thin plastic or fiber card that incorporates information about the person such as a picture or signature and the person's name on it to charge purchases and services to his linked account. Charges are debited regularly. Nowadays, card data is read by ATMs, swiping machines, store readers, banks and online transactions.
2. Each card has a unique card number which is very important. Its security mainly relies on the physical security of the card and also the privacy of the credit card number. There is a rapid growth in credit card transactions which has led to substantial growth in scam cases.
3. Credit card fraud is expanding heavily because fraud financial loss is increasing drastically. Multiple data mining and statistical techniques are used to catch fraud. Therefore the detection of fraud using efficient and secured methods are very important.

Tasks To Be Performed:

1. Load the dataset using the pandas module.
2. Perform missing value analysis on the dataset.
3. From the dataset, calculate the number of genuine transactions, number of fraud transactions and the percentage of fraud transactions.
4. Using the visualization module, visualize the genuine and fraudulent transactions using a bar graph.
5. Using the Standard Scaler module, normalize the amount column and store the new values in the NormalizedAmount column.
6. Split the dataset in train and test set and have a 70:30 split ratio for the model.
7. Now use a decision tree and random forest model for training on top of the train set.
8. Compare the predictions of both models using predict().
9. Compare the accuracy of both models using score().
10. Check the performance matrix of both models and compare which model is having the highest performance.

In [39]:

```
#1. Load the dataset using the pandas module.
import pandas as pd

df = pd.read_csv('creditcard.csv.crdownload')
df.head()
```

Out[39]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.12853
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.16717
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.32764
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.64737
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.20601

5 rows × 31 columns

In [40]:
```python
#2. Perform missing value analysis on the dataset.
df.isnull().sum()
```

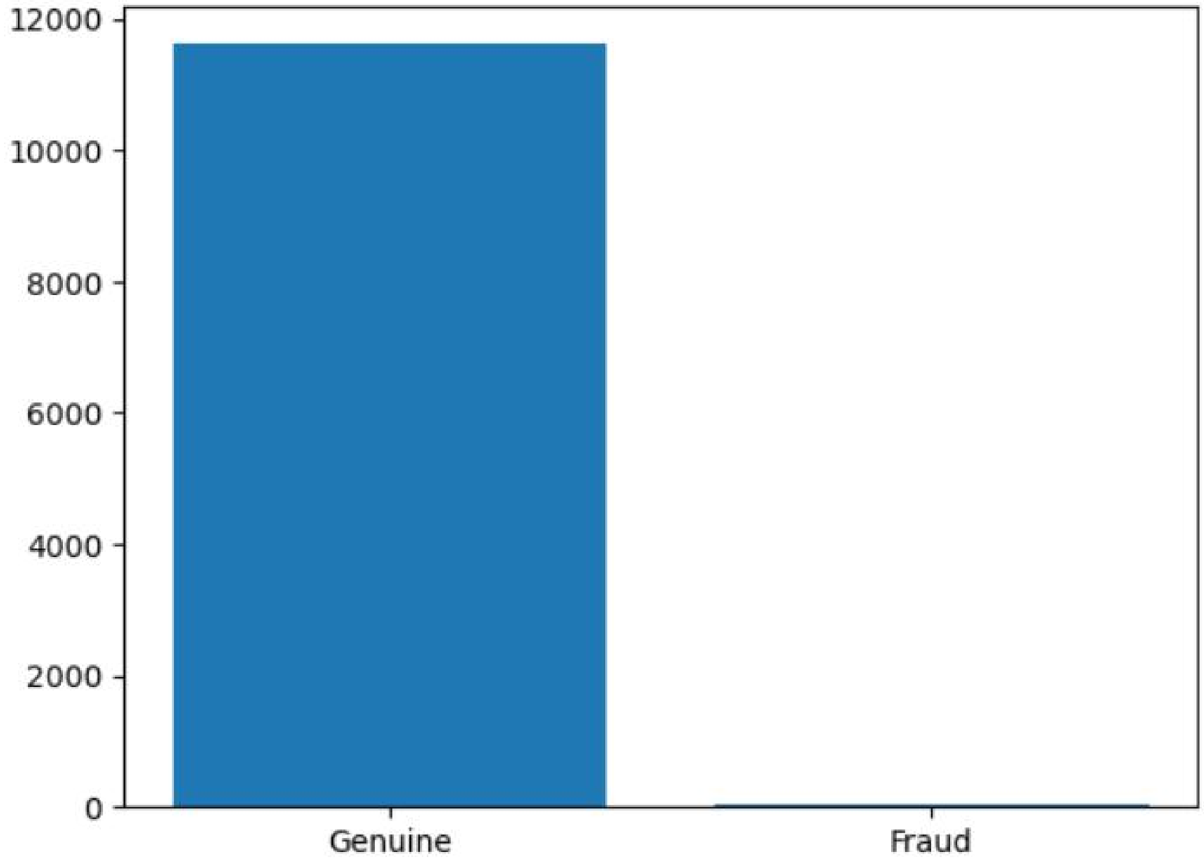
Out[40]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       1
V13       1
V14       1
V15       1
V16       1
V17       1
V18       1
V19       1
V20       1
V21       1
V22       1
V23       1
V24       1
V25       1
V26       1
V27       1
V28       1
Amount    1
Class     1
dtype: int64
```

In [41]:
```python
df.dropna(inplace=True)
```

```
In [42]: #3. From the dataset, calculate the number of genuine transactions, number of fraud transactions and the percentage of fraud trar
genuine = df[df['Class'] == 0]['Class'].count()
fraud = df[df['Class'] == 1]['Class'].count()
total = genuine + fraud
percentage_fraud = (fraud / total) * 100

print("Number of genuine transactions:", genuine)
print("Number of fraud transactions:", fraud)
print("Percentage of fraud transactions:", percentage_fraud)
```

```
Number of genuine transactions: 11615
Number of fraud transactions: 49
Percentage of fraud transactions: 0.4200960219478738
```

In [43]:
```python
#4. Using the visualization module, visualize the genuine and fraudulent transactions using a bar graph.
import matplotlib.pyplot as plt

genuine = df[df['Class'] == 0]['Class'].count()
fraud = df[df['Class'] == 1]['Class'].count()

plt.bar(['Genuine', 'Fraud'], [genuine, fraud])
plt.show()
```



In [44]:
```
#5. Using the Standard Scaler module, normalize the amount column and store the new values in the NormalizedAmount column.
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df['NormalizedAmount'] = scaler.fit_transform(df['Amount'].values.reshape(-1, 1))
df.drop(['Amount'], axis=1, inplace=True)
```

In [45]:
```
#6. Split the dataset in train and test set and have a 70:30 split ratio for the model.
from sklearn.model_selection import train_test_split

X = df.drop(['Class'], axis=1)
y = df['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
print("Shape of train_X: ", X_train.shape)
print("Shape of test_X: ", X_test.shape)
```

```
Shape of train_X:  (8164, 30)
Shape of test_X:  (3500, 30)
```

In [46]:
```python
#7. Now use a decision tree and random forest model for training on top of the train set.
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dtc = DecisionTreeClassifier()
rfc = RandomForestClassifier()

dtc.fit(X_train, y_train)
rfc.fit(X_train, y_train)
```

Out[46]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [47]:
```python
X_test.dropna(inplace=True)
y_test.dropna(inplace=True)
```

In [48]:
```python
print("Number of rows in X_test:", len(X_test))
print("Number of rows in y_test:", len(y_test))
```

```
Number of rows in X_test: 3500
Number of rows in y_test: 3500
```

In [49]:
```python
#8. Compare the predictions of both models using predict().
y_pred_dtc = dtc.predict(X_test)
y_pred_rfc = rfc.predict(X_test)
```

In [50]:
```python
#9. Compare the accuracy of both models using score().
print("Decision Tree Classifier Accuracy:", dtc.score(X_test, y_test))
print("Random Forest Classifier Accuracy:", rfc.score(X_test, y_test))
```

```
Decision Tree Classifier Accuracy: 0.9982857142857143
Random Forest Classifier Accuracy: 0.9988571428571429
```

In [51]:
```python
y_pred_dtc[:20]
```

Out[51]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0.])
```

In [52]:
```python
y_pred_rfc[:20]
```

Out[52]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0.])
```

In [53]:
```python
#10. Check the performance matrix of both models and compare which model is having the highest performance.
from sklearn.metrics import confusion_matrix

y_pred_dtc = dtc.predict(X_test)
y_pred_rfc = rfc.predict(X_test)

cm_dtc = confusion_matrix(y_test, y_pred_dtc)
cm_rfc = confusion_matrix(y_test, y_pred_rfc)

print("Decision Tree Classifier Confusion Matrix:")
print(cm_dtc)
print("Random Forest Classifier Confusion Matrix:")
print(cm_rfc)
```

```
Decision Tree Classifier Confusion Matrix:
[[3482    0]
 [   6   12]]
Random Forest Classifier Confusion Matrix:
[[3482    0]
 [   4   14]]
```

```
In [54]: from sklearn.metrics import accuracy_score, precision_score, confusion_matrix, recall_score, f1_score

         def metrics(actuals, predictions):
             print("Accuracy: {:.5f}".format(accuracy_score(actuals, predictions)))
             print("Precision: {:.5f}".format(precision_score(actuals, predictions)))
             print("Recall: {:.5f}".format(recall_score(actuals, predictions)))
             print("F1-score: {:.5f}".format(f1_score(actuals, predictions)))
```

```
In [55]: print("Decision Tree Classifier Metrics:")
         metrics(y_test, y_pred_dtc)

         print("Random Forest Classifier Metrics:")
         metrics(y_test, y_pred_rfc)
```

```
Decision Tree Classifier Metrics:
Accuracy: 0.99829
Precision: 1.00000
Recall: 0.66667
F1-score: 0.80000
Random Forest Classifier Metrics:
Accuracy: 0.99886
Precision: 1.00000
Recall: 0.77778
F1-score: 0.87500
```

Note: The Random Forest Classifier has a higher performance than the Decision Tree Classifier. The Random Forest Classifier has an accuracy of 99.88%,while the Decision Tree Classifier has an accuracy of 99.82%