In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

import warnings;
warnings.simplefilter('ignore')
```

In [2]:

```
df = pd.read_csv('payment_fraud.csv')
df.head()
```

Out[2]:

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	label
0	29	1	4.745402	paypal	28.204861	0
1	725	1	4.742303	storecredit	0.000000	0
2	845	1	4.921318	creditcard	0.000000	0
3	503	1	4.886641	creditcard	0.000000	0
4	2000	1	5.040929	creditcard	0.000000	0

In [3]:

```
train, test = train_test_split(df, test_size=0.3, random_state=1)
```

In [4]:

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

In [5]:

```
X
```

Out[5]:

	accountAgeDays	numItems	localTime	paymentMethodAgeDays	label
0	29	1	4.745402	28.204861	0
1	725	1	4.742303	0.000000	0
2	845	1	4.921318	0.000000	0
3	503	1	4.886641	0.000000	0
4	2000	1	5.040929	0.000000	0
...	...	...	...	...	...
39216	986	1	4.836982	0.000000	0
39217	1647	1	4.876771	377.930556	0
39218	1591	1	4.742303	0.000000	0
39219	237	1	4.921318	236.082639	0
39220	272	1	5.040929	0.000694	0

39221 rows × 5 columns

In [6]:

```
df['paymentMethod'].replace(['paypal','storecredit','creditcard'],[0,1,2],inplace=True)
```

In [7]:

```
y
```

Out[7]:

```
0        0
1        1
2        2
3        2
4        2
       ..
39216    2
39217    2
39218    2
39219    2
39220    0
Name: paymentMethod, Length: 39221, dtype: int64
```

In [8]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

In [9]:

```
x_train.head()
```

Out[9]:

	accountAgeDays	numItems	localTime	paymentMethodAgeDays	label
29661	341	1	4.876771	0.000694	0
721	590	2	4.524580	109.174306	0
26901	1684	1	5.040929	0.000000	0
19689	1719	1	5.034622	1.094444	0
28244	922	1	4.921349	406.868750	0

In [10]:

```
x_test.head()
```

Out[10]:

	accountAgeDays	numItems	localTime	paymentMethodAgeDays	label
6413	95	1	5.034622	0.149306	0
19681	2000	1	5.034622	116.172917	0
36157	29	1	5.040929	0.000000	0
27761	88	1	4.742303	0.000000	0
18120	42	1	4.895263	41.868750	0

In [11]:

```
y_train.head()
```

Out[11]:

```
29661    2
721      2
26901    2
19689    2
28244    2
Name: paymentMethod, dtype: int64
```

In [12]:

```
y_test.head()
```

Out[12]:

```
6413     2
19681    0
36157    2
27761    2
18120    2
Name: paymentMethod, dtype: int64
```

KNN Classifier

In [13]:

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier()
clf.fit(x_train,y_train)
```

Out[13]:

```
KNeighborsClassifier
KNeighborsClassifier()
```

In [14]:

```
y_pred=clf.predict(x_test)
```

In [15]:

```
print("Acutual Payment Method : ")
print(y_test.values)
```

```
Acutual Payment Method :
[2 0 2 ... 2 0 2]
```

In [16]:

```
print("\nPredicted Payment Method : ")
print(y_pred)
```

```
Predicted Payment Method :
[2 2 2 ... 0 2 2]
```

In [17]:

```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matri

print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred) * 100))
```

```
Accuracy score: 64.544914
```

In [18]:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[ 384   28 2411]
 [  71    3  481]
 [1102   79 7208]]
```

In [19]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.25      0.14      0.18      2823
           1       0.03      0.01      0.01       555
           2       0.71      0.86      0.78      8389

    accuracy                           0.65     11767
   macro avg       0.33      0.33      0.32     11767
weighted avg       0.57      0.65      0.60     11767
```

Decision Tree

In [20]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.metrics import accuracy_score
```

In [21]:

```
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(x_train,y_train)
```

Out[21]:

```
DecisionTreeClassifier(criterion='entropy')
```

In [22]:

```
y_pred=clf.predict(x_test)
```

In [23]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred) * 100))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy score: 56.607462
[[ 601  135 2087]
 [ 134   30  391]
 [1968  391 6030]]
```

In [24]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.22      0.21      0.22      2823
           1       0.05      0.05      0.05       555
           2       0.71      0.72      0.71      8389

    accuracy                           0.57     11767
   macro avg       0.33      0.33      0.33     11767
weighted avg       0.56      0.57      0.56     11767
```

Naive Bayes Classification

In [25]:

```
from sklearn.naive_bayes import GaussianNB
nb_classifier =GaussianNB()
nb_classifier.fit(x_train, y_train)
```

Out[25]:

GaussianNB

GaussianNB()

In [26]:

```
y_pred=nb_classifier.predict(x_test)
```

In [27]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred) * 100))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy score: 70.731707
[[   0   25 2798]
 [   0    1  554]
 [   0   67 8322]]
```

In [28]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2823
           1       0.01      0.00      0.00       555
           2       0.71      0.99      0.83      8389

    accuracy                           0.71     11767
   macro avg       0.24      0.33      0.28     11767
weighted avg       0.51      0.71      0.59     11767
```

Random Forest Classifier

In [29]:

```
from sklearn.ensemble import RandomForestClassifier
rmf = RandomForestClassifier(max_depth=3, random_state=0)
rmf_clf = rmf.fit(x_train, y_train)
```

In [30]:

```
#Predict on test data
y_pred=rmf.predict(x_test)
```

In [31]:

```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matri
print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred) * 100))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy score: 71.292598
[[   0    0 2823]
 [   0    0  555]
 [   0    0 8389]]
```

In [32]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      2823
           1       0.00      0.00      0.00       555
           2       0.71      1.00      0.83      8389

    accuracy                           0.71     11767
   macro avg       0.24      0.33      0.28     11767
weighted avg       0.51      0.71      0.59     11767
```

From the above results Random Forest Classifier got the high accuracy score