2306AML1107 MAHESH BABU M ASSIGNMENT --9

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
fraud=pd.read_csv('fraud.csv')
fraud.head()
```

Out[3]:

	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 [4]:
```
fraud.isnull().sum()
```

Out[4]:
```
accountAgeDays          0
numItems                0
localTime               0
paymentMethod           0
paymentMethodAgeDays    0
label                   0
dtype: int64
```

In [5]:
```
print(fraud.dtypes)
```

```
accountAgeDays            int64
numItems                  int64
localTime               float64
paymentMethod            object
paymentMethodAgeDays    float64
label                     int64
dtype: object
```

In [57]:
```
print(fraud.describe())
```

```
       accountAgeDays      numItems     localTime  paymentMethodAgeDays  \
count    39221.000000  39221.000000  39221.000000          39221.000000
mean       857.563984      1.084751      4.748232            122.641326
std        804.788212      0.566899      0.389360            283.569177
min          1.000000      1.000000      0.421214              0.000000
25%         72.000000      1.000000      4.742303              0.000000
50%        603.000000      1.000000      4.886641              0.012500
75%       1804.000000      1.000000      4.962055             87.510417
max       2000.000000     29.000000      5.040929           1999.580556

              label
count  39221.000000
mean       0.014278
std        0.118636
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
```

In [58]:
```
fraud1 = fraud.sort_values(['label','paymentMethodAgeDays'],ascending = [False,False])\
                        .reset_index(drop = True)
print(fraud1.shape)
fraud1.head(10)
```

```
(39221, 6)
```

Out[58]:

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	label
0	1	1	4.921349	creditcard	0.991667	1
1	1	1	4.886641	creditcard	0.950000	1
2	1	1	4.745402	creditcard	0.661806	1
3	1	1	4.886641	creditcard	0.659028	1
4	1	1	4.742303	paypal	0.618750	1
5	1	1	3.575983	paypal	0.509028	1
6	1	1	3.483951	creditcard	0.457639	1
7	1	1	4.524580	creditcard	0.370139	1
8	1	1	4.524580	creditcard	0.209722	1
9	1	3	4.921349	creditcard	0.090278	1

In [59]:
```
paymeth=fraud1.groupby('label')['paymentMethod'].count()
print(paymeth)
```

```
label
0    38661
1      560
Name: paymentMethod, dtype: int64
```

In [60]:
```
print(fraud1.groupby('label')['accountAgeDays'].count())
```

```
label
0    38661
1      560
Name: accountAgeDays, dtype: int64
```

In [61]:
```
#who are paid through the  acount AgeDays is 1 are found fruad
```

In [62]:
```
#data encoding using dict method

dict={'creditcard': 'c', 'paypal': 'p', 'storecredit': 's'}
fraud1['paymentMethod']=fraud1['paymentMethod'].map(dict)
print(fraud1)
```

```
       accountAgeDays  numItems  localTime paymentMethod  \
0                   1         1   4.921349             c
1                   1         1   4.886641             c
2                   1         1   4.745402             c
3                   1         1   4.886641             c
4                   1         1   4.742303             p
...               ...       ...        ...           ...
39216            1116         1   4.895263             c
39217            1607         1   4.886641             c
39218            2000         1   4.886641             c
39219             986         1   4.836982             c
39220            1591         1   4.742303             c

       paymentMethodAgeDays  label
0                  0.991667      1
1                  0.950000      1
2                  0.661806      1
3                  0.659028      1
4                  0.618750      1
...                     ...    ...
39216              0.000000      0
39217              0.000000      0
39218              0.000000      0
39219              0.000000      0
39220              0.000000      0

[39221 rows x 6 columns]
```

In [99]:
```
table = pd.pivot_table(fraud1, values='label', index=['paymentMethod', 'localTime'],
                       columns=['paymentMethodAgeDays'])
print (table)
```

```
paymentMethodAgeDays     0.000000     0.000694     0.001389     0.002083     \
paymentMethod localTime
c             0.421214           NaN          NaN     0.000000          NaN
              2.596228      0.205882     0.333333     0.200000     0.666667
              2.948940      0.129032     0.000000     0.400000     0.000000
              3.066058      0.128205     0.125000     0.142857     0.000000
              3.483951      0.060976     0.000000     0.142857     0.500000
...                              ...          ...          ...          ...
s             4.962055      0.000000     0.000000     0.000000     0.000000
              4.965339      0.000000     0.111111          NaN     0.000000
              5.017904      0.000000     0.166667     0.000000     0.000000
              5.034622      0.051282     0.000000     0.000000     0.000000
              5.040929      0.000000     0.000000     0.000000     0.000000

paymentMethodAgeDays     0.002778     0.003472     0.004167     0.004861     \
paymentMethod localTime
c             0.421214           NaN          NaN          NaN          NaN
              2.596228           0.0          NaN          0.0          0.0
              2.948940           0.5          NaN          NaN          1.0
              3.066058           0.0          0.0          NaN          0.0
              3.483951           0.0          0.0          0.0          0.0
...                              ...          ...          ...          ...
s             4.962055           NaN          NaN          0.0          NaN
              4.965339           0.0          NaN          0.0          NaN
              5.017904           0.0          NaN          0.0          0.0
              5.034622           NaN          NaN          NaN          NaN
              5.040929           NaN          0.0          NaN          0.0

paymentMethodAgeDays     0.005556     0.006250     ...  1983.398611  \
paymentMethod localTime                            ...
c             0.421214           NaN          NaN  ...          NaN
              2.596228           0.0          0.0  ...          0.0
              2.948940           0.0          0.0  ...          NaN
              3.066058           0.0          NaN  ...          NaN
              3.483951           NaN          NaN  ...          NaN
...                              ...          ...  ...          ...
s             4.962055           0.0          NaN  ...          NaN
              4.965339           NaN          NaN  ...          NaN
              5.017904           0.0          NaN  ...          NaN
              5.034622           NaN          NaN  ...          NaN
              5.040929           NaN          NaN  ...          NaN

paymentMethodAgeDays     1985.883333  1986.064583  1988.273611  1988.689583  \
paymentMethod localTime
c             0.421214           NaN          NaN          NaN          NaN
              2.596228           NaN          NaN          NaN          NaN
              2.948940           NaN          NaN          NaN          NaN
              3.066058           NaN          NaN          NaN          NaN
              3.483951           NaN          NaN          NaN          NaN
...                              ...          ...          ...          ...
s             4.962055           NaN          NaN          NaN          NaN
              4.965339           NaN          NaN          NaN          NaN
              5.017904           NaN          NaN          NaN          NaN
              5.034622           NaN          NaN          NaN          NaN
              5.040929           NaN          NaN          NaN          NaN

paymentMethodAgeDays     1988.692361  1988.699306  1991.989583  1994.804861  \
paymentMethod localTime
c             0.421214           NaN          NaN          NaN          NaN
              2.596228           NaN          NaN          NaN          NaN
              2.948940           NaN          NaN          NaN          NaN
              3.066058           NaN          NaN          NaN          NaN
              3.483951           NaN          NaN          NaN          NaN
...                              ...          ...          ...          ...
s             4.962055           NaN          NaN          NaN          NaN
              4.965339           NaN          NaN          NaN          NaN
              5.017904           NaN          NaN          NaN          NaN
              5.034622           NaN          NaN          NaN          NaN
              5.040929           NaN          NaN          NaN          NaN

paymentMethodAgeDays     1999.580556
paymentMethod localTime
c             0.421214           NaN
              2.596228           NaN
              2.948940           NaN
              3.066058           NaN
              3.483951           NaN
...                              ...
s             4.962055           NaN
              4.965339           NaN
              5.017904           NaN
              5.034622           NaN
              5.040929           NaN

[74 rows x 17094 columns]
```

In [100…
```
table.count()
```

Out[100]:
```
paymentMethodAgeDays
0.000000       72
0.000694       69
0.001389       62
0.002083       61
0.002778       52
               ..
1988.692361     1
1988.699306     1
1991.989583     1
1994.804861     1
1999.580556     1
Length: 17094, dtype: int64
```

In [101…
```
table.isnull().sum()
```

Out[101]:
```
paymentMethodAgeDays
0.000000        2
0.000694        5
0.001389       12
0.002083       13
0.002778       22
               ..
1988.692361    73
1988.699306    73
1991.989583    73
1994.804861    73
1999.580556    73
Length: 17094, dtype: int64
```

In [76]:
```
print(fraud1.groupby('label')['paymentMethod'].count())
```

```
label
0    38661
1      560
Name: paymentMethod, dtype: int64
```

In [77]:
```
fraud1.isnull().sum()
```

Out[77]:
```
accountAgeDays          0
numItems                0
localTime               0
paymentMethod           0
paymentMethodAgeDays    0
label                   0
dtype: int64
```

In [23]:
```
"""
'Outcome' is the column which we are
going to predict
fraud or not.
1 means the person is fruad  and 0 means notfruad.
"""
print(fraud1.groupby('label').size())
```

```
label
0    38661
1      560
dtype: int64
```

In [83]:
```
print(fraud1[fraud1.accountAgeDays

 >20].groupby('label')['paymentMethod'].count())
```

```
label
0    33349
Name: paymentMethod, dtype: int64
```

In [85]:
```
print(fraud1[fraud1.numItems

 ==1].groupby('label')['paymentMethod'].count())
```

```
label
0    36944
1      454
Name: paymentMethod, dtype: int64
```

In [90]:
```
fraud2 = fraud1['label'].groupby(fraud1['localTime'])

# printing the means value
print(fraud2.count())
```

```
localTime
0.421214       2
2.596228     213
2.948940     228
3.066058     264
3.483951     365
3.575983     443
3.954522     646
4.057414    1088
4.461622    1083
4.505662    1531
4.524580    2302
4.742303    1696
4.745402    2264
4.748314    2284
4.836982    2253
4.876771    2333
4.886641    2313
4.895263    2081
4.921318    2444
4.921349    2353
4.962055    2244
4.965339    2155
5.017904    2276
5.034622    2310
5.040929    2050
Name: label, dtype: int64
```

In [92]:
```
fraud3= fraud1['paymentMethod'].groupby(fraud1['label'])

# printing the means value
print(fraud3.count())
```

```
label
0    38661
1      560
Name: paymentMethod, dtype: int64
```

In [121…
```
dict={'c':1 ,  'p': '2' ,  's':'3' }
fraud1['paymentMethod']=fraud1['paymentMethod'].map(dict)
print(fraud1)
```

```
       accountAgeDays  numItems  localTime paymentMethod  \
0                   1         1   4.921349             1
1                   1         1   4.886641             1
2                   1         1   4.745402             1
3                   1         1   4.886641             1
4                   1         1   4.742303             2
...               ...       ...        ...           ...
39216            1116         1   4.895263             1
39217            1607         1   4.886641             1
39218            2000         1   4.886641             1
39219             986         1   4.836982             1
39220            1591         1   4.742303             1

       paymentMethodAgeDays  label
0                  0.991667      1
1                  0.950000      1
2                  0.661806      1
3                  0.659028      1
4                  0.618750      1
...                     ...    ...
39216              0.000000      0
39217              0.000000      0
39218              0.000000      0
39219              0.000000      0
39220              0.000000      0

[39221 rows x 6 columns]
```

In [125…
```
X = fraud1.drop(['label'],axis=1 )
Y= fraud1.label
```

In [126…
```
X
```

Out[126]:

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays
0	1	1	4.921349	1	0.991667
1	1	1	4.886641	1	0.950000
2	1	1	4.745402	1	0.661806
3	1	1	4.886641	1	0.659028
4	1	1	4.742303	2	0.618750
...	...	...	...	...	...
39216	1116	1	4.895263	1	0.000000
39217	1607	1	4.886641	1	0.000000
39218	2000	1	4.886641	1	0.000000
39219	986	1	4.836982	1	0.000000
39220	1591	1	4.742303	1	0.000000

39221 rows × 5 columns

In [127…
```
Y
```

Out[127]:
```
0        1
1        1
2        1
3        1
4        1
        ..
39216    0
39217    0
39218    0
39219    0
39220    0
Name: label, Length: 39221, dtype: int64
```

In [128…
```
import warnings;
warnings.simplefilter('ignore')
```

In [129…
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
```

In [130…
```
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
```

In [131…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, stratify = fraud1.label)
```

In [134…
```
names = []
scores = []
for name, model in models:
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    scores.append(accuracy_score(Y_test, Y_pred))
    names.append(name)
tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
```

```
  Name     Score
0  KNN  0.999898
1   DT  1.000000
2  GNB  0.999694
```

In [135…
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(Y_test, Y_pred))
```

```
[[9666    0]
 [   3  137]]
```

In [137…
```
print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9666
           1       1.00      0.98      0.99       140

    accuracy                           1.00      9806
   macro avg       1.00      0.99      0.99      9806
weighted avg       1.00      1.00      1.00      9806
```

In [138…
```
gnb = GaussianNB()

# Train our classifier
model = gnb.fit(X_train, Y_train)

# Make predictions
Y_pred = gnb.predict(X_test)
print(Y_pred)

# Evaluate accuracy
print(accuracy_score(Y_test, Y_pred))
#Classification accuracy: percentage of correct predictions
```

```
[0 0 0 ... 0 0 0]
0.9996940648582501
```

In [140…
```
from sklearn.ensemble import RandomForestClassifier
rmf = RandomForestClassifier(max_depth=3, random_state=0)
rmf_clf = rmf.fit(X_train, Y_train)
```

In [147…
```
from sklearn import preprocessing
```

In [149…
```
minmax=preprocessing.MinMaxScaler(feature_range=(0,1))
minmax.fit(X).transform(X)
```

Out[149]:
```
array([[0.00000000e+00, 0.00000000e+00, 9.74115289e-01, 0.00000000e+00,
        4.95937342e-04],
       [0.00000000e+00, 0.00000000e+00, 9.66602272e-01, 0.00000000e+00,
        4.75099639e-04],
       [0.00000000e+00, 0.00000000e+00, 9.36029178e-01, 0.00000000e+00,
        3.30972190e-04],
       ...,
       [1.00000000e+00, 0.00000000e+00, 9.66602272e-01, 0.00000000e+00,
        0.00000000e+00],
       [4.92746373e-01, 0.00000000e+00, 9.55852909e-01, 0.00000000e+00,
        0.00000000e+00],
       [7.95397699e-01, 0.00000000e+00, 9.35358357e-01, 0.00000000e+00,
        0.00000000e+00]])
```

In [153…
```
#Evaluation Metrics
from sklearn.model_selection import cross_val_score,cross_val_predict
```

In [168…
```
from sklearn.model_selection import cross_val_score
X_train_std = minmax.fit_transform(X_train)
X_test_std = minmax.transform(X_test)
```

In [158…
```
#Print cross validation score for Random Forest Calssifier
rmf_clf_acc = cross_val_score(rmf_clf, x_train_std, Y_train, cv=3, scoring="accuracy",  n_jobs=-1)
rmf_proba = cross_val_predict(rmf_clf, x_train_std, Y_train, cv=3, method='predict_proba')
rmf_clf_scores = rmf_proba[:, 1]
```

In [159…
```
#Predict on test data
Y_pred=rmf.predict(X_test)
```

In [162…
```
#Accuracy score on Test and Train
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix
print("\nAccuracy score: %f" %(accuracy_score(Y_test,Y_pred) * 100))
print("Recall score : %f" %(recall_score(Y_test, Y_pred) * 100))
print("ROC score : %f\n" %(roc_auc_score(Y_test, Y_pred) * 100))
print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy score: 100.000000
Recall score : 100.000000
ROC score : 100.000000

[[9666    0]
 [   0  140]]
```

In [170…
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

In [171…
```
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(X_train, y_train)
```

Out[171]:
```
▾          SVC
SVC(degree=8, kernel='poly')
```

In [172…
```
y_pred = svclassifier.predict(X_test)
```

In [173…
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
[[7726    0]
 [ 119    0]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      7726
           1       0.00      0.00      0.00       119

    accuracy                           0.98      7845
   macro avg       0.49      0.50      0.50      7845
weighted avg       0.97      0.98      0.98      7845

Accuracy: 0.9848311026131293
```

In [174…
```
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
```

Out[174]:
```
▾ SVC
SVC()
```

In [175…
```
y_pred = svclassifier.predict(X_test)
```

In [176…
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
[[7726    0]
 [ 119    0]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      7726
           1       0.00      0.00      0.00       119

    accuracy                           0.98      7845
   macro avg       0.49      0.50      0.50      7845
weighted avg       0.97      0.98      0.98      7845

Accuracy: 0.9848311026131293
```

In [177…
```
from sklearn.svm import SVC
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(X_train, y_train)
```

Out[177]:
```
▾          SVC
SVC(kernel='sigmoid')
```

In [178…
```
y_pred = svclassifier.predict(X_test)
```

In [179…
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
[[7726    0]
 [ 119    0]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      7726
           1       0.00      0.00      0.00       119

    accuracy                           0.98      7845
   macro avg       0.49      0.50      0.50      7845
weighted avg       0.97      0.98      0.98      7845

Accuracy: 0.9848311026131293
```

In [194…
```
#comparision  of classifiers
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
models.append(('rmf',RandomForestClassifier()))
```

In [192…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, stratify = fraud1.label)
```

In [193…
```
names = []
scores = []
for name, model in models:
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    scores.append(accuracy_score(Y_test, Y_pred))
    names.append(name)
tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
```

```
  Name     Score
0  KNN  0.999898
1   DT  1.000000
2  GNB  0.999694
3  rmf  1.000000
```

In [210…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, stratify = fraud1.label)
```

In [218…
```
from sklearn.svm import SVC
svcpoly = SVC(kernel='poly', degree=8) ,
svcrbf = SVC(kernel='rbf')
svcsigmoid = SVC(kernel='sigmoid')
```

In [219…
```
models = []
models.append(('svcpoly',SVC(degree=8, kernel='poly')))
models.append(('svcrbf',SVC()) )
models.append(('svcsibmoid',SVC(kernel='sigmoid')))
```

In [220…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, stratify = fraud1.label)
```

In [221…
```
names = []
scores = []
for name, model in models:
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)

    scores.append(accuracy_score(Y_test, Y_pred))
    names.append(name)
tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
```

```
         Name     Score
0     svcpoly  0.985723
1      svcrbf  0.985723
2  svcsibmoid  0.985723
```

In [ ]: