Assignment 2 - ML - Navya Renduchintala - 2306AML112

Fraud is one of the major issues we come up majorly in banks, life insurance, health insurance, and many others. These major frauds are dependent on the person who is trying to sell you the fake product or service, if you are matured enough to decide what is wrong then you will never get into any fraud transactions. But one such fraud that has been increasing a lot these days is fraud in making payments.

payment_fraud.csv

Write a classification program and compare various classification algorithms using payment_fraud.csv dataset

In [1]:
```
import pandas as pd
payment_fraud = pd.read_csv("payment_fraud.csv")
payment_fraud
```

Out[1]:

	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
...	...	...	...	...	...	...
39216	986	1	4.836982	creditcard	0.000000	0
39217	1647	1	4.876771	creditcard	377.930556	0
39218	1591	1	4.742303	creditcard	0.000000	0
39219	237	1	4.921318	creditcard	236.082639	0
39220	272	1	5.040929	paypal	0.000694	0

39221 rows × 6 columns

In [2]:
```
print(payment_fraud.isnull().sum())
```

```
accountAgeDays          0
numItems                0
localTime               0
paymentMethod           0
paymentMethodAgeDays    0
label                   0
dtype: int64
```

In [3]:
```
payment_fraud_2 = payment_fraud.copy()
```

In [4]:
```
data_t = {"paymentMethod": {"creditcard": 1, "paypal": 2, "storecredit":3}}
payment_fraud_2.replace(data_t, inplace=True)
print(payment_fraud_2)
```

```
       accountAgeDays  numItems  localTime  paymentMethod  \
0                  29         1   4.745402              2
1                 725         1   4.742303              3
2                 845         1   4.921318              1
3                 503         1   4.886641              1
4                2000         1   5.040929              1
...               ...       ...        ...            ...
39216             986         1   4.836982              1
39217            1647         1   4.876771              1
39218            1591         1   4.742303              1
39219             237         1   4.921318              1
39220             272         1   5.040929              2

       paymentMethodAgeDays  label
0                 28.204861      0
1                  0.000000      0
2                  0.000000      0
3                  0.000000      0
4                  0.000000      0
...                     ...    ...
39216              0.000000      0
39217            377.930556      0
39218              0.000000      0
39219            236.082639      0
39220              0.000694      0

[39221 rows x 6 columns]
```

In [5]:
```
x=payment_fraud_2.iloc[:,0:5]
y=payment_fraud_2.iloc[:,5]
```

In [6]:
```
x
```

Out[6]:

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

39221 rows × 5 columns

In [7]:
```
y
```

Out[7]:
```
0        0
1        0
2        0
3        0
4        0
        ..
39216    0
39217    0
39218    0
39219    0
39220    0
Name: label, Length: 39221, dtype: int64
```

In [8]:
```
from sklearn.neighbors import KNeighborsClassifier

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
```

Out[9]:

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays
25308	316	1	4.876771	1	315.688194
18708	2000	1	4.742303	1	0.002778
4986	157	1	4.742303	2	0.000000
17601	402	1	4.742303	1	4.726389
15917	7	1	4.745402	2	0.000000
...	...	...	...	...	...
7025	886	1	4.921318	1	0.000000
10169	213	1	4.836982	2	25.050694
7373	2000	1	4.836982	2	0.000000
14729	24	2	4.748314	2	0.000000
166	2000	1	2.948940	1	0.000000

27454 rows × 5 columns

In [10]:
```
y_train
```

Out[10]:
```
25308    0
18708    0
4986     0
17601    0
15917    0
        ..
7025     0
10169    0
7373     0
14729    0
166      0
Name: label, Length: 27454, dtype: int64
```

In [11]:
```
x_test
```

Out[11]:

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays
27144	2000	1	4.962055	1	0.150000
37284	1804	1	4.505662	1	0.000000
13268	3	1	5.040929	1	2.246528
34529	2000	1	3.954522	2	1004.015278
17311	2000	1	4.921318	1	0.008333
...	...	...	...	...	...
16510	952	1	4.748314	1	0.000000
19392	2000	1	2.596228	2	1008.652083
4489	2000	1	5.040929	1	0.002778
26956	25	1	4.876771	2	10.221528
33083	842	1	4.965339	1	0.000000

11767 rows × 5 columns

In [12]:
```
y_test
```

Out[12]:
```
27144    0
37284    0
13268    0
34529    0
17311    0
        ..
16510    0
19392    0
4489     0
26956    0
33083    0
Name: label, Length: 11767, dtype: int64
```

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(y_test.values)
```

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

In [16]:
```
print(y_pred)
```

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

In [17]:
```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix

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

```
Accuracy score: 100.000000
```

Comparing Classification Algorithmns

In [18]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [19]:
```
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
models.append(('LR',LogisticRegression()))
models.append(('SVM',SVC()))
```

In [20]:
```
models
```

Out[20]:
```
[('KNN', KNeighborsClassifier()),
 ('DT', DecisionTreeClassifier()),
 ('GNB', GaussianNB()),
 ('LR', LogisticRegression()),
 ('SVM', SVC())]
```

In [21]:
```
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 = y, random_state=0)
```

In [22]:
```
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  1.000000
1   DT  1.000000
2  GNB  0.999898
3   LR  1.000000
4  SVM  0.985723
```

KNeighbors , Decision Tree and Logistic Regression Classification Algorithms are best suited for determining Payment Fraud based on accuracy_score metrics.