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('CC_GENERAL.csv')
df.head()
```

Out[2]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INS
0	C10001	40.900749	0.818182	95.40	0.00	
1	C10002	3202.467416	0.909091	0.00	0.00	
2	C10003	2495.148862	1.000000	773.17	773.17	
3	C10004	1666.670542	0.636364	1499.00	1499.00	
4	C10005	817.714335	1.000000	16.00	16.00	

In [3]:

```
df.shape
```

Out[3]:

```
(8950, 18)
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   CUST_ID                           8950 non-null   object 
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64  
 12  PURCHASES_TRX                     8950 non-null   int64  
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64  
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

In [6]:

```
df['TENURE'].unique()
```

Out[6]:

```
array([12,  8, 11,  9, 10,  7,  6], dtype=int64)
```

In [7]:

```
print(df.groupby('TENURE').size())
```

```
TENURE
6      204
7      190
8      196
9      175
10     236
11     365
12    7584
dtype: int64
```

In [8]:

```
dataset = df.drop(['CUST_ID','CREDIT_LIMIT','MINIMUM_PAYMENTS'],axis = 1)
```

In [9]:

```
X = dataset.iloc[:,:-1]
```

In [10]:

```
X
```

Out[10]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLME
0	40.900749	0.818182	95.40	0.00	
1	3202.467416	0.909091	0.00	0.00	
2	2495.148862	1.000000	773.17	773.17	
3	1666.670542	0.636364	1499.00	1499.00	
4	817.714335	1.000000	16.00	16.00	
...	...	...	...	...	
8945	28.493517	1.000000	291.12	0.00	
8946	19.183215	1.000000	300.00	0.00	
8947	23.398673	0.833333	144.40	0.00	
8948	13.457564	0.833333	0.00	0.00	
8949	372.708075	0.666667	1093.25	1093.25	

8950 rows × 14 columns

In [11]:

```
y = dataset.iloc[:,-1:]
```

In [12]:

```
y
```

Out[12]:

	TENURE
0	12
1	12
2	12
3	12
4	12
...	...
8945	6
8946	6
8947	6
8948	6
8949	6

8950 rows × 1 columns

In [13]:

```
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 [14]:

```
X_train.head()
```

Out[14]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLME
1200	3322.314890	1.000000	676.00	601.0	
2304	933.150722	0.727273	0.00	0.0	
795	644.530629	1.000000	268.68	0.0	
3832	43.200086	1.000000	324.00	324.0	
4458	1714.896284	1.000000	2000.00	2000.0	

In [15]:

```
X_test.head()
```

Out[15]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLME
8143	1179.613243	1.000000	0.00	0.00	
2446	4779.786334	1.000000	1470.24	882.78	
1174	72.706076	0.272727	508.56	508.56	
8779	525.143055	0.571429	0.00	0.00	
927	634.825209	1.000000	1478.36	1343.45	

In [16]:

```
y_train.head()
```

Out[16]:

	TENURE
1200	12
2304	11
795	12
3832	12
4458	9

In [17]:

```
y_test.head()
```

Out[17]:

	TENURE
8143	12
2446	12
1174	12
8779	7
927	12

KNN Classifier

In [18]:

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

Out[18]:

```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [19]:

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

In [20]:

```
print("Acutual Tenure : ")
print(y_test.values)
```

```
Acutual Tenure :
[[12]
 [12]
 [12]
 ...
 [12]
 [12]
 [12]]
```

In [21]:

```
print("\nPredicted Tenure : ")
print(y_pred)
```

```
Predicted Tenure :
[12 12 12 ... 12 12 12]
```

In [22]:

```
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: 82.644320
```

In [23]:

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

```
[[   4    4    3    0    1    3   50]
 [   1    4    1    2    0    1   51]
 [   3    1    4    0    1    0   55]
 [   3    1    1    1    2    2   39]
 [   2    1    1    0    0    5   51]
 [   1    4    1    0    1    1   94]
 [  19   10   14   12   13   12 2205]]
```

In [24]:

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

```
              precision    recall  f1-score   support

           6       0.12      0.06      0.08        65
           7       0.16      0.07      0.09        60
           8       0.16      0.06      0.09        64
           9       0.07      0.02      0.03        49
          10       0.00      0.00      0.00        60
          11       0.04      0.01      0.02       102
          12       0.87      0.96      0.91      2285

    accuracy                           0.83      2685
   macro avg       0.20      0.17      0.18      2685
weighted avg       0.75      0.83      0.78      2685
```

Decision Tree

In [25]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [26]:

```
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
```

Out[26]:

▾ DecisionTreeClassifier

DecisionTreeClassifier()

In [27]:

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

In [28]:

```
print("Acutual Tenure : ")
print(y_test.values)
```

```
Acutual Tenure :
[[12]
 [12]
 [12]
 ...
 [12]
 [12]
 [12]]
```

In [29]:

```
print("\nPredicted Tenure : ")
print(y_pred)
```

```
Predicted Tenure :
[12 12 12 ... 12 12 12]
```

In [30]:

```
from sklearn.metrics import accuracy_score, confusion_matrix

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

```
[[  36    2    0    5    2    0   20]
 [   2   42    3    1    4    3    5]
 [   3    6   42    2    1    1    9]
 [   0    2    4   33    4    1    5]
 [   1    1    1    3   38    2   14]
 [   2    3    1    0    2   63   31]
 [  22   11   16    9   25   35 2167]]

Accuracy score: 90.167598
```

In [31]:

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

```
              precision    recall  f1-score   support

           6       0.55      0.55      0.55        65
           7       0.63      0.70      0.66        60
           8       0.63      0.66      0.64        64
           9       0.62      0.67      0.65        49
          10       0.50      0.63      0.56        60
          11       0.60      0.62      0.61       102
          12       0.96      0.95      0.96      2285

    accuracy                           0.90      2685
   macro avg       0.64      0.68      0.66      2685
weighted avg       0.91      0.90      0.90      2685
```

Naive Bayes Classification

In [32]:

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

Out[32]:

▾ GaussianNB

GaussianNB()

In [33]:

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

In [34]:

```
print("Acutual Tenure : ")
print(y_test.values)
```

```
Acutual Tenure :
[[12]
 [12]
 [12]
 ...
 [12]
 [12]
 [12]]
```

In [35]:

```
print("\nPredicted Tenure : ")
print(y_pred)
```

```
Predicted Tenure :
[ 7 11  7 ...  8  6 12]
```

In [36]:

```
from sklearn.metrics import accuracy_score, confusion_matrix

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

```
[[ 31  11   8   1   0   3  11]
 [ 15  14  18   2   0   2   9]
 [ 18  21  11   1   0   4   9]
 [ 15   6  13   1   0   6   8]
 [ 13  17  12   0   0   5  13]
 [ 11  25  34   4   0   6  22]
 [211 385 614  29   1 126 919]]

Accuracy score: 36.573557
```

In [37]:

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

```
              precision    recall  f1-score   support

           6       0.10      0.48      0.16        65
           7       0.03      0.23      0.05        60
           8       0.02      0.17      0.03        64
           9       0.03      0.02      0.02        49
          10       0.00      0.00      0.00        60
          11       0.04      0.06      0.05       102
          12       0.93      0.40      0.56      2285

    accuracy                           0.37      2685
   macro avg       0.16      0.19      0.13      2685
weighted avg       0.79      0.37      0.49      2685
```

Random Forest Classifier

In [38]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

rmf = RandomForestClassifier()
rmf_clf = rmf.fit(X_train, y_train)
```

In [39]:

```
y_pred=rmf.predict(X_test)
```

In [40]:

```
print("Acutual Tenure : ")
print(y_test.values)
```

```
Acutual Tenure :
[[12]
 [12]
 [12]
 ...
 [12]
 [12]
 [12]]
```

In [41]:

```
print("\nPredicted Tenure : ")
print(y_pred)
```

```
Predicted Tenure :
[12 12 12 ... 12 12 12]
```

In [42]:

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

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

```
[[  30    2    1    1    0    0   31]
 [   3   22    3    0    1    1   30]
 [   0    3   26    6    5    0   24]
 [   0    1    4   17    6    0   21]
 [   0    0    1    1   25    4   29]
 [   0    0    0    1    2   32   67]
 [   0    0    0    1    0    2 2282]]

Accuracy score: 90.651769
```

In [43]:

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

```
              precision    recall  f1-score   support

           6       0.91      0.46      0.61        65
           7       0.79      0.37      0.50        60
           8       0.74      0.41      0.53        64
           9       0.63      0.35      0.45        49
          10       0.64      0.42      0.51        60
          11       0.82      0.31      0.45       102
          12       0.92      1.00      0.96      2285

    accuracy                           0.91      2685
   macro avg       0.78      0.47      0.57      2685
weighted avg       0.90      0.91      0.89      2685
```

SVM algorithm

In [44]:

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

Out[44]:

SVC

SVC()

In [45]:

```
y_pred = svclassifier.predict(X_test)
```

In [46]:

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

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

```
[[   0    0    0    0    0    0   65]
 [   0    0    0    0    0    0   60]
 [   0    0    0    0    0    0   64]
 [   0    0    0    0    0    0   49]
 [   0    0    0    0    0    0   60]
 [   0    0    0    0    0    0  102]
 [   0    0    0    0    0    0 2285]]

Accuracy score: 85.102421
```

In [47]:

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

```
              precision    recall  f1-score   support

           6       0.00      0.00      0.00        65
           7       0.00      0.00      0.00        60
           8       0.00      0.00      0.00        64
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        60
          11       0.00      0.00      0.00       102
          12       0.85      1.00      0.92      2285

    accuracy                           0.85      2685
   macro avg       0.12      0.14      0.13      2685
weighted avg       0.72      0.85      0.78      2685
```

Comparing Classifiers

In [48]:

```
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
models.append(("RFC", RandomForestClassifier()))
models.append(("SVM",SVC()))

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  0.826443
1   DT  0.901304
2  GNB  0.365736
3  RFC  0.901676
4  SVM  0.851024
```

From the above results RandomForest Classifier got high accuracy score