In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, roc_curve, auc
import seaborn as sns

"""For DT plotting"""
from io import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

In [ ]:

In [2]:

```
retail = pd.read_csv("Retail_Customer_Insights.csv")
retail.head()
```

Out[2]:

	Customer_ID	Age	Annual_Income	Gender	Purchase_History	Product_Category	Customer_Satisfaction	Loyalty_Points	Marital
0	CID770487	45	72633.53	Non-binary	0	Electronics	9.0	541.11	[
1	CID216739	38	61816.55	Non-binary	0	Books	6.0	497.41	
2	CID126225	47	57338.15	Non-binary	0	Grocery	3.0	634.90	
3	CID877572	58	83800.37	Female	0	Furniture	4.0	505.82	[
4	CID388389	37	64875.12	Male	0	Furniture	6.0	610.39	

In [3]:

```
retail.isnull().sum()
```

Out[3]:

```
Customer_ID                     0
Age                             0
Annual_Income                5000
Gender                          0
Purchase_History                0
Product_Category                0
Customer_Satisfaction        3000
Loyalty_Points               2000
Marital_Status                  0
Number_of_Children              0
Employment_Status               0
Credit_Score                    0
Owns_House                      0
Monthly_Expenditure          5000
Internet_Usage_Hours_per_Week   0
dtype: int64
```

In [4]:

```
retail.isna().sum()
```

Out[4]:

```
Customer_ID                     0
Age                             0
Annual_Income                5000
Gender                          0
Purchase_History                0
Product_Category                0
Customer_Satisfaction        3000
Loyalty_Points               2000
Marital_Status                  0
Number_of_Children              0
Employment_Status               0
Credit_Score                    0
Owns_House                      0
Monthly_Expenditure          5000
Internet_Usage_Hours_per_Week   0
dtype: int64
```

In [5]:

```
retail.fillna({'Annual_Income':retail['Annual_Income'].median(),
                        'Customer_Satisfaction':retail['Customer_Satisfaction'].median(),
                        'Loyalty_Points':retail['Loyalty_Points'].median(),
                        'Monthly_Expenditure':retail['Monthly_Expenditure'].median()},inplace=True)
```

In [6]:

```
retail.isna().sum()
```

Out[6]:

```
Customer_ID                    0
Age                            0
Annual_Income                  0
Gender                         0
Purchase_History               0
Product_Category               0
Customer_Satisfaction          0
Loyalty_Points                 0
Marital_Status                 0
Number_of_Children             0
Employment_Status              0
Credit_Score                   0
Owns_House                     0
Monthly_Expenditure            0
Internet_Usage_Hours_per_Week    0
dtype: int64
```

In [7]:

```
retail.isnull().sum()
```

Out[7]:

```
Customer_ID                    0
Age                            0
Annual_Income                  0
Gender                         0
Purchase_History               0
Product_Category               0
Customer_Satisfaction          0
Loyalty_Points                 0
Marital_Status                 0
Number_of_Children             0
Employment_Status              0
Credit_Score                   0
Owns_House                     0
Monthly_Expenditure            0
Internet_Usage_Hours_per_Week    0
dtype: int64
```

In [8]:

```
# Removing outliers from Age column
dataset=np.array(retail['Age']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=retail.loc[(retail['Age']>ll) & (retail['Age']<ul)]
```

In [9]:

```
# Removing outliers from Annual_Income column
dataset=np.array(data['Annual_Income']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Annual_Income']>ll) & (data['Annual_Income']<ul)]
```

In [10]:

```
# Removing outliers from Loyalty_Points column
dataset=np.array(data['Loyalty_Points']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Loyalty_Points']>ll) & (data['Loyalty_Points']<ul)]
```

In [11]:

```
# Removing outliers from Number_of_Children column
dataset=np.array(data['Number_of_Children']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Number_of_Children']>ll) & (data['Number_of_Children']<ul)]
```

In [12]:

```
# Removing outliers from Credit_Score column
dataset=np.array(data['Credit_Score']).tolist()
```

```
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(retail['Credit_Score']>ll) & (data['Credit_Score']<ul)]
```

In [13]:

```
# Removing outliers from Number_of_Children column
dataset=np.array(data['Monthly_Expenditure']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Monthly_Expenditure']>ll) & (data['Monthly_Expenditure']<ul)]
```

In [14]:

```
# Removing outliers from Internet_Usage_Hours_per_Week column
dataset=np.array(data['Internet_Usage_Hours_per_Week']).tolist()
dataset.sort()
median_= np.median(dataset)
q1=np.percentile(dataset,25)
q3=np.percentile(dataset,75)
iqr=q3-q1
ll=q1-1.5*iqr
ul=q3+1.5*iqr
data=data.loc[(data['Internet_Usage_Hours_per_Week']>ll) & (data['Internet_Usage_Hours_per_Week']<ul)]
```

In [15]:

```
# setting customer_id as index
data.set_index('Customer_ID',inplace=True)
data.head()
```

Out[15]:

	Age	Annual_Income	Gender	Purchase_History	Product_Category	Customer_Satisfaction	Loyalty_Points	Marital_St
Customer_ID								
CID770487	45	72633.53	Non-binary	0	Electronics	9.0	541.11	Divc
CID216739	38	61816.55	Non-binary	0	Books	6.0	497.41	Ma
CID126225	47	57338.15	Non-binary	0	Grocery	3.0	634.90	S
CID877572	58	83800.37	Female	0	Furniture	4.0	505.82	Divc
CID356787	37	57270.25	Prefer not to say	1	Grocery	3.0	458.98	Divc

In [16]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 81315 entries, CID770487 to CID966793
Data columns (total 14 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Age                            81315 non-null  int64  
 1   Annual_Income                  81315 non-null  float64
 2   Gender                         81315 non-null  object 
 3   Purchase_History               81315 non-null  int64  
 4   Product_Category               81315 non-null  object 
 5   Customer_Satisfaction          81315 non-null  float64
 6   Loyalty_Points                 81315 non-null  float64
 7   Marital_Status                 81315 non-null  object 
 8   Number_of_Children             81315 non-null  int64  
 9   Employment_Status              81315 non-null  object 
 10  Credit_Score                   81315 non-null  int64  
 11  Owns_House                     81315 non-null  bool   
 12  Monthly_Expenditure            81315 non-null  float64
 13  Internet_Usage_Hours_per_Week  81315 non-null  int64  
dtypes: bool(1), float64(4), int64(5), object(4)
memory usage: 8.8+ MB
```

In [17]:

```
cat_cols=['Gender','Product_Category','Marital_Status','Employment_Status','Owns_House']
for i in cat_cols:
    le=LabelEncoder()
    data[i]=le.fit_transform(data[i])
```

In [18]:
```
data.head()
```

Out[18]:

	Age	Annual_Income	Gender	Purchase_History	Product_Category	Customer_Satisfaction	Loyalty_Points	Marital_St
Customer_ID								
CID770487	45	72633.53	2	0	2	9.0	541.11	
CID216739	38	61816.55	2	0	0	6.0	497.41	
CID126225	47	57338.15	2	0	4	3.0	634.90	
CID877572	58	83800.37	0	0	3	4.0	505.82	
CID356787	37	57270.25	3	1	4	3.0	458.98	

In [19]:
```
data.shape
```

Out[19]:
```
(81315, 14)
```

In [20]:
```
x=data.drop('Purchase_History',axis=1)
y=data['Purchase_History']
```

In [21]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=74)
```

In [22]:
```
#Feature Scaling
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [23]:
```
model_dt_2 = DecisionTreeClassifier(max_depth=2)
model_dt_2.fit(x_train_scaled,y_train)
model_dt_2_tr_score = model_dt_2.score(x_train_scaled,y_train)
model_dt_2_te_score = model_dt_2.score(x_test_scaled,y_test)

print(f'Training Score: {model_dt_2_tr_score}')
print(f'Test Score: {model_dt_2_te_score}')
```

```
Training Score: 0.5984904384184959
Test Score: 0.6006271905552482
```

In [24]:
```
model_dt_4 = DecisionTreeClassifier(max_depth=4)
model_dt_4.fit(x_train_scaled,y_train)
model_dt_4_tr_score = model_dt_4.score(x_train_scaled,y_train)
model_dt_4_te_score = model_dt_4.score(x_test_scaled,y_test)

print(f'Training Score: {model_dt_4_tr_score}')
print(f'Test Score: {model_dt_4_te_score}')
```

```
Training Score: 0.5987210231814548
Test Score: 0.6003812334747587
```

In [25]:
```
model_dt_8 = DecisionTreeClassifier(max_depth=8)
model_dt_8.fit(x_train_scaled,y_train)
model_dt_8_tr_score = model_dt_8.score(x_train_scaled,y_train)
model_dt_8_te_score = model_dt_8.score(x_test_scaled,y_test)

print(f'Training Score: {model_dt_8_tr_score}')
print(f'Test Score: {model_dt_8_te_score}')
```

```
Training Score: 0.6021490499907766
Test Score: 0.5976142163192523
```

In [26]:
```
model_dt_16 = DecisionTreeClassifier(max_depth=16)
model_dt_16.fit(x_train_scaled,y_train)
model_dt_16_tr_score = model_dt_16.score(x_train_scaled,y_train)
model_dt_16_te_score = model_dt_16.score(x_test_scaled,y_test)

print(f'Training Score: {model_dt_16_tr_score}')
print(f'Test Score: {model_dt_16_te_score}')
```

```
Training Score: 0.6284049683330258
Test Score: 0.5891286970423661
```

In [27]:
```
model_dt_2 = DecisionTreeClassifier(max_depth=2,criterion='entropy')
model_dt_2.fit(x_train_scaled,y_train)
model_dt_2_tr_score = model_dt_2.score(x_train_scaled,y_train)
model_dt_2_te_score = model_dt_2.score(x_test_scaled,y_test)

print(f'Training Score: {model_dt_2_tr_score}')
print(f'Test Score: {model_dt_2_te_score}')
```

```
Training Score: 0.5984904384184959
Test Score: 0.6006271905552482
```

```
In [29]: model_dt_4 = DecisionTreeClassifier(max_depth=4,criterion='entropy')
         model_dt_4.fit(x_train_scaled,y_train)
         model_dt_4_tr_score = model_dt_4.score(x_train_scaled,y_train)
         model_dt_4_te_score = model_dt_4.score(x_test_scaled,y_test)

         print(f'Training Score: {model_dt_4_tr_score}')
         print(f'Test Score: {model_dt_4_te_score}')

         Training Score: 0.5986595339113324
         Test Score: 0.6002582549345139
```

```
In [31]: model_dt_8 = DecisionTreeClassifier(max_depth=8,criterion='entropy')
         model_dt_8.fit(x_train_scaled,y_train)
         model_dt_8_tr_score = model_dt_8.score(x_train_scaled,y_train)
         model_dt_8_te_score = model_dt_8.score(x_test_scaled,y_test)

         print(f'Training Score: {model_dt_8_tr_score}')
         print(f'Test Score: {model_dt_8_te_score}')

         Training Score: 0.6007194244604317
         Test Score: 0.5985980446412101
```

```
In [33]: tr_prediction = model_dt_2.predict(x_train_scaled)

         print(model_dt_2.score(x_train_scaled,y_train))

         0.5984904384184959
```

```
In [35]: te_prediction = model_dt_2.predict(x_test_scaled)

         print(model_dt_2.score(x_test_scaled,y_test))

         0.6006271905552482
```

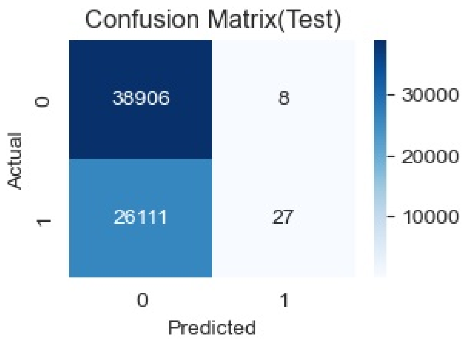
```
In [37]: cm_tr = confusion_matrix(y_train,tr_prediction)
         cm_tr

Out[37]: array([[38906,     8],
                [26111,    27]], dtype=int64)
```

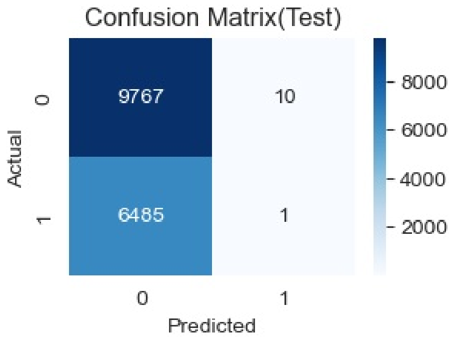
```
In [39]: cm_te = confusion_matrix(y_test,te_prediction)
         cm_te

Out[39]: array([[9767,   10],
                [6485,    1]], dtype=int64)
```

```
In [41]: sns.set({'figure.figsize':(3,2)})
         sns.heatmap(cm_tr,fmt='d',annot=True,cmap='Blues')
         plt.xlabel('Predicted')
         plt.ylabel('Actual')
         plt.title('Confusion Matrix(Test)')
         plt.show()
```



```
In [43]: sns.set({'figure.figsize':(3,2)})
         sns.heatmap(cm_te,fmt='d',annot=True,cmap='Blues')
         plt.xlabel('Predicted')
         plt.ylabel('Actual')
         plt.title('Confusion Matrix(Test)')
         plt.show()
```



In [45]:

```
from sklearn.ensemble import AdaBoostClassifier
base_classifier = DecisionTreeClassifier(max_depth = 2)
adaboost_classifier = AdaBoostClassifier(estimator = base_classifier,n_estimators = 40, random_state = 37)
adaboost_classifier.fit(x_train_scaled, y_train)
y_train_pred = adaboost_classifier.predict(x_train_scaled)
print(adaboost_classifier.score(x_train_scaled, y_train))
print(adaboost_classifier.score(x_test_scaled, y_test))
```

```
0.6007194244604317
0.5974297485088852
```

In [47]:

```
from sklearn.ensemble import AdaBoostClassifier
base_classifier = DecisionTreeClassifier(max_depth = 2)
adaboost_classifier = AdaBoostClassifier(estimator = base_classifier,n_estimators = 30, random_state = 37)
adaboost_classifier.fit(x_train_scaled, y_train)
y_train_pred = adaboost_classifier.predict(x_train_scaled)
print(adaboost_classifier.score(x_train_scaled, y_train))
print(adaboost_classifier.score(x_test_scaled, y_test))
```

```
0.6004273504273504
0.5985980446412101
```

In [49]:

```
from sklearn.ensemble import AdaBoostClassifier
base_classifier = DecisionTreeClassifier(max_depth = 1)
adaboost_classifier = AdaBoostClassifier(estimator = base_classifier,n_estimators = 30, random_state = 37)
adaboost_classifier.fit(x_train_scaled, y_train)
y_train_pred = adaboost_classifier.predict(x_train_scaled)
print(adaboost_classifier.score(x_train_scaled, y_train))
print(adaboost_classifier.score(x_test_scaled, y_test))
```

```
0.5985826723236796
0.6005042120150034
```

In [51]:

```
from sklearn.ensemble import AdaBoostClassifier
base_classifier = DecisionTreeClassifier(max_depth = 1)
adaboost_classifier = AdaBoostClassifier(estimator = base_classifier,n_estimators = 50, random_state = 37)
adaboost_classifier.fit(x_train_scaled, y_train)
y_train_pred = adaboost_classifier.predict(x_train_scaled)
print(adaboost_classifier.score(x_train_scaled, y_train))
print(adaboost_classifier.score(x_test_scaled, y_test))
```

```
0.5986595339113324
0.600442722744881
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js