In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,roc_curve
```

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.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 [4]:
```
# 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 [5]:
```
# Removing outliers from Internet_Usage_Hours_per_Week column
out_cols=['Annual_Income','Loyalty_Points','Number_of_Children','Credit_Score','Monthly_Expenditure','Internet_
for i in out_cols:
    dataset=np.array(data[i]).tolist()
    dataset.sort()
    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[i]>ll) & (data[i]<ul)]
```

In [6]:
```
# setting customer_id as index
data.set_index('Customer_ID',inplace=True)
data.head()
```

Out[6]:

Customer_ID	Age	Annual_Income	Gender	Purchase_History	Product_Category	Customer_Satisfaction	Loyalty_Points	Marital_St
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 [7]:
```
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 [8]:
```
x=data.drop('Purchase_History',axis=1)
y=data['Purchase_History']
```

In [9]:
```
#splitting data into train and test
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=75)
```

In [10]:
```
#Feature Scaling
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [11]:
```
model_NB=GaussianNB()
model_NB.fit(x_train_scaled,y_train)
```

Out[11]:
```
▾ GaussianNB
GaussianNB()
```

In [12]:
```
train_pred=model_NB.predict(x_train_scaled)
```

In [13]:
```
test_pred=model_NB.predict(x_test_scaled)
```

In [16]:
```
train_accuracy = accuracy_score(y_train,train_pred)
print('Train accuracy: ',train_accuracy)
```

```
Train accuracy:  0.5973221422861711
```

In [17]:
```
test_accuracy = accuracy_score(y_test,test_pred)
print('Test accuracy: ',test_accuracy)
```

```
Test accuracy:  0.6046854823833241
```

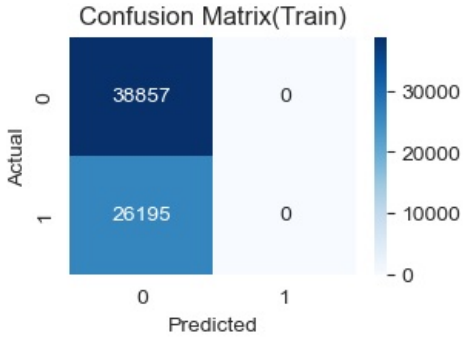
In [18]:
```
cm_train = confusion_matrix(y_train,train_pred)
cm_train
```

Out[18]:
```
array([[38857,     0],
       [26195,     0]], dtype=int64)
```

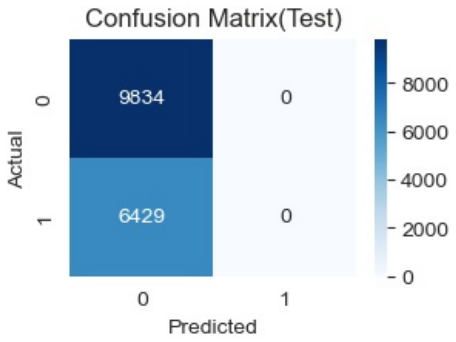
In [19]:
```
cm_test = confusion_matrix(y_test,test_pred)
cm_test
```

Out[19]:
```
array([[9834,    0],
       [6429,    0]], dtype=int64)
```

In [26]:
```
sns.set({'figure.figsize':(3,2)})
sns.heatmap(cm_train,fmt='d',annot=True,cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix(Train)')
plt.show()
```



In [21]:
```
sns.set({'figure.figsize':(3,2)})
sns.heatmap(cm_test,fmt='d',annot=True,cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix(Test)')
plt.show()
```



In [ ]: