Predicting Customer spending category

In this assignment we are going to classify the customer according to the amount he spends. We will study the customer spending patterns and will categorize them into categories based on within how much they spend on the website. Then we will train a classification model that will classify which customer belongs to which category.

In [24]:
```
#loading the required packages

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
# Load the datasets
online_sales = pd.read_csv('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Online
customer_data = pd.read_excel('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Cus
discount_coupons = pd.read_csv('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Di
marketing_spend= pd.read_csv('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Mark
tax_amount = pd.read_excel('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Tax_am
```

In [3]:
```
# Merging datasets using 'CustomerID' and 'Product Category as the common key
merged_data = pd.merge(online_sales, customer_data, on='CustomerID', how='left')
merged_data = pd.merge(merged_data, discount_coupons, on='Product_Category', how='left')
merged_data = pd.merge(merged_data, tax_amount, on='Product_Category', how='left')
```

In [4]:
```
# Convert 'Date' column in the marketing_spend DataFrame to datetime data type
marketing_spend['Date'] = pd.to_datetime(marketing_spend['Date'])
merged_data['Transaction_Date'] = pd.to_datetime(merged_data['Transaction_Date'], format='%Y%m%d')

# Merge datasets using 'Transaction_Date' as the common key
merged_data = pd.merge(merged_data, marketing_spend, left_on='Transaction_Date', right_on='Date', how='left')


# Dropping the duplicate 'Date' column if needed
merged_data.drop(columns=['Date'], inplace=True)
```

In [5]:
```
# Calculating Invoice Value for each transaction in the merged dataset
merged_data['Invoice_Value'] = (
    (merged_data['Quantity'] * merged_data['Avg_Price']) *
    (1 - merged_data['Discount_pct']) *
    (1 + merged_data['GST']) +
    merged_data['Delivery_Charges']
).abs()
```

In [6]:
```
#examining the merged dataset
merged_data.head()
```

Out[6]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_name	Product_Category	Quantity	Avg_Price	Delivery_Charges	Co
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	
1	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	
2	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	
3	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	
4	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	

5 rows × 21 columns

In [7]:
```
#correcting the format
merged_data['Transaction_Date'] = pd.to_datetime(merged_data['Transaction_Date'])
merged_data['Transaction_Date']
```

Out[7]:
```
0         2019-01-01
1         2019-01-01
2         2019-01-01
3         2019-01-01
4         2019-01-01
             ...
630683    2019-12-31
630684    2019-12-31
630685    2019-12-31
630686    2019-12-31
630687    2019-12-31
Name: Transaction_Date, Length: 630688, dtype: datetime64[ns]
```

In [11]:
```
# Formatting the dataset
customer_data = merged_data.groupby('CustomerID').agg({
    'Transaction_ID': 'count',
    'Transaction_Date': 'max',
    'Quantity': 'sum',
    'Avg_Price': 'mean',
    'Delivery_Charges': 'sum',
    'Gender': 'first',
    'Location': 'first',

    'Tenure_Months': 'first',
    'Offline_Spend': 'sum',
    'Online_Spend': 'sum',
    'Invoice_Value': 'sum',

}).reset_index()
customer_data.round()
```

Out[11]:

	CustomerID	Transaction_ID	Transaction_Date	Quantity	Avg_Price	Delivery_Charges	Gender	Location	Tenure_Months	Offline_Spend
0	12346	24	2019-09-15	36	13.0	1800.0	F	New York	31	72000
1	12347	709	2019-11-02	4082	63.0	7910.0	M	New York	20	1418000
2	12348	276	2019-10-19	2508	15.0	2366.0	M	California	39	894000
3	12350	204	2019-12-14	252	77.0	1535.0	M	California	25	816000
4	12356	432	2019-09-15	672	35.0	7650.0	F	Chicago	31	1296000
...	...	...	...	...	...	...	...	...	...	...
1463	18259	73	2019-04-05	541	11.0	3268.0	F	California	5	182500
1464	18260	469	2019-10-05	1549	28.0	7387.0	M	New York	43	1473000
1465	18269	96	2019-06-20	120	11.0	618.0	M	Chicago	25	252000
1466	18277	12	2019-10-23	24	149.0	72.0	F	Chicago	47	42000
1467	18283	1213	2019-10-10	1816	56.0	8547.0	F	New Jersey	36	3572500

1468 rows × 12 columns

In [13]:
```
# Distributing the customers according to Gaussian norms
quantiles = [0, 0.33, 0.66, 1.0]
labels=['Low-value','Mid-value','High-value']
customer_data['Invoice_Value_Category'] = pd.qcut(customer_data['Invoice_Value'], q=quantiles, labels=labels)

print(customer_data[['Invoice_Value', 'Invoice_Value_Category']])
```

```
      Invoice_Value Invoice_Value_Category
0      6.336910e+03              Low-value
1      3.612323e+06             High-value
2      3.728585e+05              Mid-value
3      3.423656e+05              Mid-value
4      3.558345e+05              Mid-value
...             ...                    ...
1463   1.277934e+05              Low-value
1464   5.820105e+05              Mid-value
1465   2.664770e+04              Low-value
1466   7.466640e+04              Low-value
1467   1.571264e+06             High-value

[1468 rows x 2 columns]
```

In [16]:
```
#examining the final dataset
customer_data.head()
```

Out[16]:

	CustomerID	Transaction_ID	Transaction_Date	Quantity	Avg_Price	Delivery_Charges	Gender	Location	Tenure_Months	Offline_Spend	Onl
0	12346	24	2019-09-15	36	12.745000	1800.00	F	New York	31	72000	
1	12347	709	2019-11-02	4082	63.231072	7910.18	M	New York	20	1418000	1
2	12348	276	2019-10-19	2508	14.631304	2365.80	M	California	39	894000	
3	12350	204	2019-12-14	252	77.200000	1534.56	M	California	25	816000	
4	12356	432	2019-09-15	672	34.578611	7649.88	F	Chicago	31	1296000	1

In [17]:
```
# Creating an instance of LabelEncoder
label_encoder = LabelEncoder()

# List of categorical columns to encode
categorical_columns = ['Gender', 'Location']

# Apply Label Encoding to each categorical column
for col in categorical_columns:
    customer_data[col] = label_encoder.fit_transform(customer_data[col])
```

In [19]:
```
# Selecting Features and Target Variable
features = ['CustomerID', 'Quantity', 'Avg_Price', 'Delivery_Charges', 'Gender', 'Location', 'Tenure_Months',
X = customer_data[features]
y = customer_data['Invoice_Value_Category']
```

In [20]:
```
#Splitting the Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [21]:
```
# Creating and training a Multinomial Naive Bayes Classifier
clf = MultinomialNB()
clf.fit(X_train, y_train)
```

Out[21]:
```
▾ MultinomialNB
MultinomialNB()
```

In [22]:
```
# Making predictions on the test set
y_pred = clf.predict(X_test)
y_pred
```

Out[22]:
```
array(['Mid-value', 'High-value', 'Low-value', 'Mid-value', 'High-value',
       'High-value', 'Mid-value', 'Mid-value', 'Mid-value', 'Low-value',
       'Low-value', 'Low-value', 'Mid-value', 'High-value', 'Mid-value',
       'Low-value', 'Mid-value', 'High-value', 'Mid-value', 'High-value',
       'Low-value', 'High-value', 'Mid-value', 'High-value', 'Mid-value',
       'High-value', 'High-value', 'Mid-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'Low-value', 'Low-value', 'High-value', 'Mid-value',
       'High-value', 'High-value', 'Mid-value', 'Mid-value', 'Low-value',
       'Low-value', 'Mid-value', 'Mid-value', 'Low-value', 'Low-value',
       'Mid-value', 'High-value', 'Mid-value', 'Low-value', 'Mid-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'High-value',
       'High-value', 'Mid-value', 'Mid-value', 'Mid-value', 'High-value',
       'High-value', 'Mid-value', 'Low-value', 'High-value', 'High-value',
       'Low-value', 'High-value', 'Low-value', 'Low-value', 'High-value',
       'High-value', 'High-value', 'High-value', 'High-value',
       'Low-value', 'High-value', 'Low-value', 'High-value', 'Low-value',
       'Mid-value', 'High-value', 'Low-value', 'Low-value', 'High-value',
       'High-value', 'Mid-value', 'High-value', 'Mid-value', 'Low-value',
       'Low-value', 'Low-value', 'High-value', 'High-value', 'High-value',
       'High-value', 'Mid-value', 'Mid-value', 'Mid-value', 'High-value',
       'Low-value', 'Mid-value', 'High-value', 'Mid-value', 'Low-value',
       'Low-value', 'High-value', 'Low-value', 'High-value', 'High-value',
       'Low-value', 'High-value', 'High-value', 'High-value', 'Low-value',
       'Mid-value', 'High-value', 'Mid-value', 'Mid-value', 'High-value',
       'High-value', 'High-value', 'Low-value', 'High-value', 'Mid-value',
       'High-value', 'High-value', 'Low-value', 'Mid-value', 'High-value',
       'High-value', 'Low-value', 'High-value', 'Low-value', 'Mid-value',
       'Mid-value', 'Low-value', 'High-value', 'High-value', 'High-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'Mid-value',
       'High-value', 'Low-value', 'Mid-value', 'Mid-value', 'High-value',
       'High-value', 'Mid-value', 'Low-value', 'Mid-value', 'High-value',
       'High-value', 'Mid-value', 'High-value', 'Mid-value', 'High-value',
       'Mid-value', 'Mid-value', 'High-value', 'High-value', 'Low-value',
       'Low-value', 'Mid-value', 'Mid-value', 'High-value', 'High-value',
       'Low-value', 'Mid-value', 'Mid-value', 'High-value', 'Low-value',
       'High-value', 'High-value', 'Mid-value', 'Low-value', 'Mid-value',
       'Low-value', 'Mid-value', 'High-value', 'Mid-value', 'High-value',
       'Mid-value', 'Mid-value', 'Low-value', 'Mid-value', 'High-value',
       'High-value', 'High-value', 'High-value', 'Low-value', 'Low-value',
       'Low-value', 'Low-value', 'Mid-value', 'High-value', 'Low-value',
       'Mid-value', 'Mid-value', 'Low-value', 'High-value', 'Low-value',
       'High-value', 'High-value', 'High-value', 'High-value',
       'Mid-value', 'High-value', 'Low-value', 'High-value', 'Low-value',
       'Mid-value', 'High-value', 'Low-value', 'High-value', 'Low-value',
       'Mid-value', 'Low-value', 'High-value', 'Mid-value', 'Mid-value',
       'High-value', 'Mid-value', 'Mid-value', 'High-value', 'Mid-value',
       'Mid-value', 'Low-value', 'High-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'Low-value', 'Low-value', 'High-value', 'Mid-value',
       'Low-value', 'Mid-value', 'Mid-value', 'Low-value', 'Mid-value',
       'High-value', 'High-value', 'Mid-value', 'Mid-value', 'Mid-value',
       'High-value', 'Low-value', 'High-value', 'High-value', 'Mid-value',
       'High-value', 'Mid-value', 'Mid-value', 'Mid-value', 'Low-value',
       'High-value', 'Low-value', 'High-value', 'Low-value', 'Mid-value',
       'Low-value', 'Mid-value', 'Mid-value', 'High-value', 'High-value',
       'Low-value', 'High-value', 'Mid-value', 'High-value', 'High-value',
       'High-value', 'Low-value', 'High-value', 'Mid-value', 'Mid-value',
       'Low-value', 'High-value', 'Mid-value', 'High-value', 'Mid-value',
       'High-value', 'Low-value', 'High-value', 'Low-value', 'High-value',
       'Mid-value', 'High-value', 'High-value', 'Mid-value', 'High-value',
       'High-value', 'Mid-value', 'High-value', 'High-value', 'Mid-value',
       'Mid-value', 'High-value', 'Mid-value', 'Low-value', 'High-value',
       'High-value', 'Mid-value', 'High-value', 'Low-value', 'Mid-value',
       'Low-value', 'High-value', 'Low-value', 'High-value', 'Mid-value',
       'Mid-value', 'Mid-value', 'Low-value', 'High-value', 'Low-value',
       'High-value', 'Mid-value', 'Low-value', 'Low-value', 'Mid-value',
       'Low-value', 'Low-value', 'High-value', 'Mid-value', 'Low-value',
       'Low-value', 'Mid-value', 'Low-value', 'Mid-value', 'Low-value',
       'Low-value', 'Mid-value', 'Low-value', 'High-value', 'Mid-value',
       'Low-value', 'High-value', 'Low-value', 'Low-value', 'High-value',
       'Mid-value', 'High-value', 'Mid-value', 'Mid-value', 'High-value',
       'Mid-value', 'Mid-value', 'Low-value', 'Mid-value', 'Low-value',
       'Mid-value', 'High-value', 'Low-value', 'Low-value', 'High-value',
       'High-value', 'Low-value', 'High-value', 'Mid-value', 'Mid-value',
       'Low-value', 'Mid-value', 'Mid-value', 'High-value', 'Low-value',
       'Mid-value', 'Mid-value', 'High-value', 'High-value', 'High-value',
       'Mid-value', 'Mid-value', 'High-value', 'High-value', 'High-value',
       'High-value', 'Mid-value', 'High-value', 'Mid-value', 'High-value',
       'Mid-value', 'Low-value', 'High-value', 'High-value', 'Mid-value',
       'High-value', 'High-value', 'Low-value', 'Low-value', 'High-value',
       'Mid-value', 'High-value', 'Mid-value', 'Low-value', 'High-value',
       'High-value', 'High-value', 'Low-value', 'Low-value', 'Low-value',
       'High-value', 'Low-value', 'Low-value', 'High-value', 'High-value',
       'Mid-value', 'High-value', 'High-value', 'High-value', 'Mid-value',
       'High-value', 'High-value', 'High-value', 'High-value',
       'High-value', 'High-value', 'Low-value', 'High-value',
       'High-value', 'Low-value', 'Low-value', 'High-value', 'Mid-value',
       'Mid-value', 'Low-value', 'High-value', 'Low-value', 'Low-value',
       'High-value', 'High-value', 'High-value', 'Mid-value', 'Low-value',
       'Mid-value', 'Low-value', 'High-value', 'Mid-value', 'Mid-value'],
      dtype='<U10')
```

In [23]:
```
#  Model Performance Evaluation
accuracy = accuracy_score(y_test, y_pred)
classification_report_result = classification_report(y_test, y_pred)

# Printing the evaluation metrics
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report_result)
```

```
Accuracy: 0.6213151927437641

Classification Report:
               precision    recall  f1-score   support

  High-value       0.61      0.74      0.67       149
   Low-value       0.83      0.69      0.75       140
   Mid-value       0.46      0.44      0.45       152

    accuracy                           0.62       441
   macro avg       0.64      0.62      0.63       441
weighted avg       0.63      0.62      0.62       441
```

Remark: The accuracy of the model is 64% and does a decent job in classifying the customers into categories.

In [25]:
```
#  Training the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

Out[25]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

In [26]:
```
#Making the Predictions
y_pred = clf.predict(X_test)
y_pred
```

Out[26]:
```
array(['High-value', 'Mid-value', 'Low-value', 'High-value', 'High-value',
       'Mid-value', 'Mid-value', 'Mid-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'Low-value', 'Low-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'Mid-value', 'High-value', 'Mid-value', 'High-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'Mid-value',
       'High-value', 'High-value', 'Mid-value', 'Mid-value', 'Low-value',
       'High-value', 'Low-value', 'Low-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'High-value', 'Mid-value', 'Low-value', 'Low-value',
       'Low-value', 'Mid-value', 'Low-value', 'Low-value', 'Low-value',
       'Low-value', 'Mid-value', 'Mid-value', 'Low-value', 'High-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'High-value',
       'High-value', 'Mid-value', 'Mid-value', 'Low-value', 'Mid-value',
       'Mid-value', 'High-value', 'Low-value', 'High-value', 'High-value',
       'Low-value', 'Mid-value', 'Low-value', 'Low-value', 'High-value',
       'High-value', 'Mid-value', 'High-value', 'High-value', 'Low-value',
       'High-value', 'Low-value', 'High-value', 'Low-value', 'Mid-value',
       'High-value', 'Low-value', 'Low-value', 'High-value', 'High-value',
       'Mid-value', 'Mid-value', 'Mid-value', 'Low-value', 'Low-value',
       'Mid-value', 'Mid-value', 'Mid-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'Low-value', 'High-value', 'Mid-value', 'Low-value',
       'Low-value', 'Mid-value', 'Low-value', 'Low-value', 'Mid-value',
       'Mid-value', 'Low-value', 'High-value', 'High-value', 'Low-value',
       'Mid-value', 'High-value', 'High-value', 'Low-value', 'High-value',
       'Mid-value', 'Low-value', 'Mid-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'Low-value', 'Mid-value', 'High-value', 'High-value',
       'Mid-value', 'Low-value', 'High-value', 'Mid-value', 'High-value',
       'Low-value', 'High-value', 'Mid-value', 'Low-value', 'Mid-value',
       'Low-value', 'High-value', 'High-value', 'High-value', 'Low-value',
       'High-value', 'High-value', 'Mid-value', 'Mid-value', 'High-value',
       'Low-value', 'High-value', 'Low-value', 'Mid-value', 'High-value',
       'Mid-value', 'Low-value', 'Low-value', 'Mid-value', 'High-value',
       'Mid-value', 'Mid-value', 'High-value', 'High-value', 'High-value',
       'Mid-value', 'High-value', 'High-value', 'Low-value', 'Low-value',
       'Mid-value', 'Mid-value', 'High-value', 'High-value', 'Low-value',
       'High-value', 'Low-value', 'High-value', 'Low-value', 'Mid-value',
       'High-value', 'Mid-value', 'Low-value', 'High-value', 'Mid-value',
       'High-value', 'Mid-value', 'Low-value', 'High-value', 'Low-value',
       'Low-value', 'Low-value', 'Low-value', 'High-value', 'Mid-value',
       'High-value', 'Mid-value', 'Low-value', 'Low-value', 'Low-value',
       'Low-value', 'Mid-value', 'High-value', 'Low-value', 'Mid-value',
       'Mid-value', 'Low-value', 'High-value', 'Mid-value', 'Low-value',
       'High-value', 'High-value', 'High-value', 'Mid-value',
       'High-value', 'Low-value', 'High-value', 'Low-value', 'Low-value',
       'Mid-value', 'Low-value', 'High-value', 'Low-value', 'Mid-value',
       'Low-value', 'Mid-value', 'Low-value', 'High-value', 'High-value',
       'Mid-value', 'Mid-value', 'High-value', 'Mid-value', 'Mid-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'High-value',
       'Low-value', 'Low-value', 'High-value', 'Mid-value', 'Low-value',
       'Low-value', 'Mid-value', 'Low-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'High-value', 'Mid-value', 'High-value', 'Mid-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'Mid-value',
       'Low-value', 'Mid-value', 'Mid-value', 'Low-value', 'High-value',
       'Low-value', 'Mid-value', 'High-value', 'High-value', 'Mid-value',
       'High-value', 'Mid-value', 'High-value', 'High-value', 'Low-value',
       'High-value', 'High-value', 'High-value', 'High-value',
       'High-value', 'Low-value', 'Mid-value', 'Low-value', 'High-value',
       'Low-value', 'Mid-value', 'Mid-value', 'High-value', 'Low-value',
       'Mid-value', 'Mid-value', 'Mid-value', 'Low-value', 'High-value',
       'Low-value', 'High-value', 'High-value', 'Low-value', 'Mid-value',
       'Mid-value', 'Low-value', 'High-value', 'High-value', 'Mid-value',
       'High-value', 'Mid-value', 'Mid-value', 'Low-value', 'Low-value',
       'High-value', 'Mid-value', 'High-value', 'Mid-value', 'Mid-value',
       'Low-value', 'Mid-value', 'Low-value', 'Mid-value', 'Mid-value',
       'Low-value', 'Low-value', 'Low-value', 'High-value', 'Low-value',
       'High-value', 'Low-value', 'Low-value', 'Mid-value', 'High-value',
       'Mid-value', 'Low-value', 'Mid-value', 'Mid-value', 'Low-value',
       'Low-value', 'Mid-value', 'Low-value', 'Mid-value', 'Low-value',
       'Low-value', 'Low-value', 'Low-value', 'High-value', 'High-value',
       'Low-value', 'Mid-value', 'Low-value', 'Low-value', 'High-value',
       'High-value', 'High-value', 'High-value', 'Low-value',
       'High-value', 'High-value', 'Mid-value', 'Low-value', 'Low-value',
       'Low-value', 'High-value', 'Mid-value', 'Low-value', 'High-value',
       'High-value', 'Mid-value', 'Low-value', 'Mid-value', 'Mid-value',
       'Low-value', 'Mid-value', 'Mid-value', 'Mid-value', 'High-value',
       'Low-value', 'Low-value', 'Low-value', 'Mid-value', 'High-value',
       'High-value', 'High-value', 'Low-value', 'High-value',
       'High-value', 'Mid-value', 'High-value', 'Mid-value', 'Mid-value',
       'Low-value', 'High-value', 'Mid-value', 'Low-value', 'Mid-value',
       'High-value', 'Low-value', 'Mid-value', 'High-value', 'Mid-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'Mid-value',
       'Low-value', 'High-value', 'High-value', 'Mid-value', 'Mid-value',
       'Mid-value', 'Low-value', 'Mid-value', 'Low-value', 'Mid-value',
       'Mid-value', 'High-value', 'Mid-value', 'High-value', 'Mid-value',
       'High-value', 'High-value', 'High-value', 'High-value',
       'High-value', 'High-value', 'High-value', 'High-value',
       'Low-value', 'High-value', 'High-value', 'Low-value', 'Low-value',
       'High-value', 'Mid-value', 'High-value', 'Low-value', 'Mid-value',
       'Low-value', 'Low-value', 'High-value', 'High-value', 'High-value',
       'Low-value', 'Low-value', 'High-value', 'Low-value', 'Mid-value',
       'Low-value', 'Mid-value'], dtype=object)
```

In [27]:
```
#  Model Performance Evaluation
accuracy = accuracy_score(y_test, y_pred)
classification_report_result = classification_report(y_test, y_pred)

# Printing the evaluation metrics
print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report_result)
```

```
Accuracy: 1.0

Classification Report:
               precision    recall  f1-score   support

  High-value       1.00      1.00      1.00       149
   Low-value       1.00      1.00      1.00       140
   Mid-value       1.00      1.00      1.00       152

    accuracy                           1.00       441
   macro avg       1.00      1.00      1.00       441
weighted avg       1.00      1.00      1.00       441
```

Remark: The Decision Tree model is 100% accurate and we can say that it can classify customers into Spending categories perfectly.