Double-click (or enter) to edit

```
# Importing the modules
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/Mall_Customers.csv')
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
df.tail()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

```
# Display basic information about the DataFrame
print("DataFrame Info:")
print(df.info())
```

```
DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```

```
# Define features as the columns of your DataFrame (except the target variable)
features = df.drop('Age', axis=1)  # Assuming 'prices' is your target variable

# Now you can split your data into training and testing sets
X_train, X_test = train_test_split(features, test_size=0.2, random_state=42)
```

```
print(df.describe())
```

```
       CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   38.850000           60.560000               50.200000
std     57.879185   13.969007           26.264721               25.823522
min      1.000000   18.000000           15.000000                1.000000
25%     50.750000   28.750000           41.500000               34.750000
```

```
      50%    100.500000   36.000000           61.500000               50.000000
      75%    150.250000   49.000000           78.000000               73.000000
      max    200.000000   70.000000          137.000000               99.000000


# Define features as the columns of your DataFrame (except the target variable)
features = df.drop('Age', axis=1)  # Assuming 'prices' is your target variable

# Now you can split your data into training and testing sets
X_train, X_test = train_test_split(features, test_size=0.2, random_state=42)


Start coding or generate with AI.


# Selecting the features for clustering
features = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]



from sklearn.preprocessing import LabelEncoder

# Assuming 'gender' is the column with categorical data
le = LabelEncoder()
X_train['Gender'] = le.fit_transform(X_train['Gender'])
X_test['Gender'] = le.transform(X_test['Gender'])

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


scaled_features = X_train_scaled
scaled_features = np.vstack((X_train_scaled, X_test_scaled))
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)


# Plot the Elbow method graph
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')  # Within-cluster sum of squares
plt.show()
```

[4]