ex5-k-means-hierarchical-clustering

August 12, 2024

[1]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

[2]:
```
df = pd.read_csv("datasets/Mall_Customers.csv")
# loads the csv file into a pandas dataframe
df
```

[2]:
```
     CustomerID   Genre  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
..          …     …  …                 …                     …
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

[200 rows x 5 columns]
```

[3]:
```
df.head()
```

[3]:
```
   CustomerID   Genre  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
```

[4]:
```
df.shape
```

[4]: (200, 5)

[5]:
```python
df.info()  # with the help of it we get brief information about our dataset
```

```
<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   Genre                   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
```

[6]:
```python
# one way to access the annual income and spending score column
df.iloc[:, [3, 4]]
```

[6]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
..	…	…
195	120	79
196	126	28
197	126	74
198	137	18
199	137	83

[200 rows x 2 columns]

[7]:
```python
x = df.loc[:, "Annual Income (k$)":"Spending Score (1-100)"].values
```

[8]:
```python
x
```

[8]:
```
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
```

```
[ 19,  14],
[ 19,  99],
[ 20,  15],
[ 20,  77],
[ 20,  13],
[ 20,  79],
[ 21,  35],
[ 21,  66],
[ 23,  29],
[ 23,  98],
[ 24,  35],
[ 24,  73],
[ 25,   5],
[ 25,  73],
[ 28,  14],
[ 28,  82],
[ 28,  32],
[ 28,  61],
[ 29,  31],
[ 29,  87],
[ 30,   4],
[ 30,  73],
[ 33,   4],
[ 33,  92],
[ 33,  14],
[ 33,  81],
[ 34,  17],
[ 34,  73],
[ 37,  26],
[ 37,  75],
[ 38,  35],
[ 38,  92],
[ 39,  36],
[ 39,  61],
[ 39,  28],
[ 39,  65],
[ 40,  55],
[ 40,  47],
[ 40,  42],
[ 40,  42],
[ 42,  52],
[ 42,  60],
[ 43,  54],
[ 43,  60],
[ 43,  45],
[ 43,  41],
[ 44,  50],
```

```
[ 44,  46],
[ 46,  51],
[ 46,  46],
[ 46,  56],
[ 46,  55],
[ 47,  52],
[ 47,  59],
[ 48,  51],
[ 48,  59],
[ 48,  50],
[ 48,  48],
[ 48,  59],
[ 48,  47],
[ 49,  55],
[ 49,  42],
[ 50,  49],
[ 50,  56],
[ 54,  47],
[ 54,  54],
[ 54,  53],
[ 54,  48],
[ 54,  52],
[ 54,  42],
[ 54,  51],
[ 54,  55],
[ 54,  41],
[ 54,  44],
[ 54,  57],
[ 54,  46],
[ 57,  58],
[ 57,  55],
[ 58,  60],
[ 58,  46],
[ 59,  55],
[ 59,  41],
[ 60,  49],
[ 60,  40],
[ 60,  42],
[ 60,  52],
[ 60,  47],
[ 60,  50],
[ 61,  42],
[ 61,  49],
[ 62,  41],
[ 62,  48],
[ 62,  59],
[ 62,  55],
```

```
[ 62,  56],
[ 62,  42],
[ 63,  50],
[ 63,  46],
[ 63,  43],
[ 63,  48],
[ 63,  52],
[ 63,  54],
[ 64,  42],
[ 64,  46],
[ 65,  48],
[ 65,  50],
[ 65,  43],
[ 65,  59],
[ 67,  43],
[ 67,  57],
[ 67,  56],
[ 67,  40],
[ 69,  58],
[ 69,  91],
[ 70,  29],
[ 70,  77],
[ 71,  35],
[ 71,  95],
[ 71,  11],
[ 71,  75],
[ 71,   9],
[ 71,  75],
[ 72,  34],
[ 72,  71],
[ 73,   5],
[ 73,  88],
[ 73,   7],
[ 73,  73],
[ 74,  10],
[ 74,  72],
[ 75,   5],
[ 75,  93],
[ 76,  40],
[ 76,  87],
[ 77,  12],
[ 77,  97],
[ 77,  36],
[ 77,  74],
[ 78,  22],
[ 78,  90],
[ 78,  17],
```

```
[ 78,  88],
[ 78,  20],
[ 78,  76],
[ 78,  16],
[ 78,  89],
[ 78,   1],
[ 78,  78],
[ 78,   1],
[ 78,  73],
[ 79,  35],
[ 79,  83],
[ 81,   5],
[ 81,  93],
[ 85,  26],
[ 85,  75],
[ 86,  20],
[ 86,  95],
[ 87,  27],
[ 87,  63],
[ 87,  13],
[ 87,  75],
[ 87,  10],
[ 87,  92],
[ 88,  13],
[ 88,  86],
[ 88,  15],
[ 88,  69],
[ 93,  14],
[ 93,  90],
[ 97,  32],
[ 97,  86],
[ 98,  15],
[ 98,  88],
[ 99,  39],
[ 99,  97],
[101,  24],
[101,  68],
[103,  17],
[103,  85],
[103,  23],
[103,  69],
[113,   8],
[113,  91],
[120,  16],
[120,  79],
[126,  28],
[126,  74],
```

```
       [137,  18],
       [137,  83]])
```

1 Exploratory Data Analysis (EDA)

[9]:
```python
# Renaming a column in the dataset
df.rename(
    columns={"Genre": "Gender"}, inplace=True
)  # To rename column 2 from Genre to Gender
df.head()  # Checking if the correction has been effected
```

[9]:
```
   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
```

[10]:
```python
# Checking data types and shape
df.dtypes  # returns the data types of the variables
```

[10]:
```
CustomerID                 int64
Gender                    object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object
```

[11]:
```python
# Descriptive statistics
df.describe()  # returns the descriptive statistics of the dataset.
```

[11]:
```
       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
```

[12]:
```python
# Looking for null or missing values
df.isnull().sum()  # returns the number of missing values
```

[12]:
```
CustomerID                0
Gender                    0
```

```
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

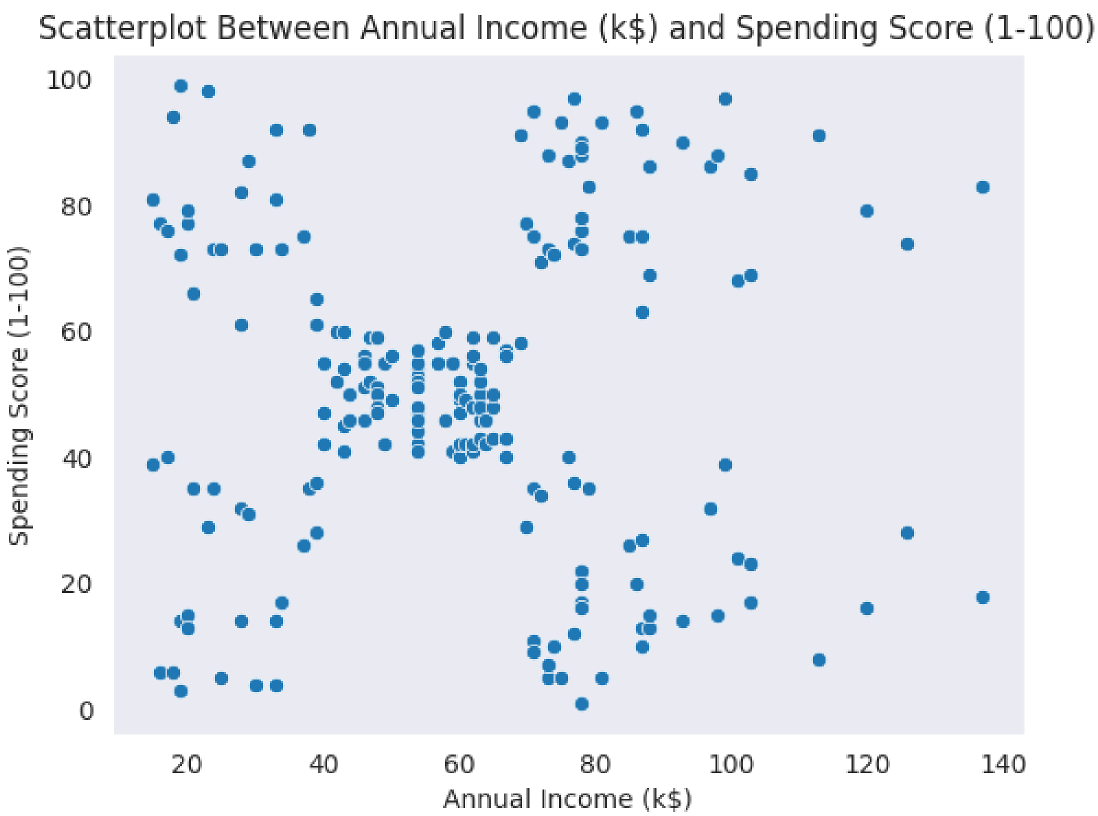
[13]:
```python
# Looking for duplicated values
df.duplicated()  # Checking for duplicate values.
```

[13]:
```
0      False
1      False
2      False
3      False
4      False
       …
195    False
196    False
197    False
198    False
199    False
Length: 200, dtype: bool
```

2 Bivariate Analysis — Scatterplot

[14]:
```python
sns.set_style("dark")
sns.scatterplot(x="Annual Income (k$)", y="Spending Score (1-100)", data=df)
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.title("Scatterplot Between Annual Income (k$) and Spending Score (1-100)")
```

[14]:
```
Text(0.5, 1.0, 'Scatterplot Between Annual Income (k$) and Spending Score
(1-100)')
```



[15]:
```
# Feature Selection(Choosing the columns of interest for clustering)
X = df.loc[:, ["Annual Income (k$)", "Spending Score (1-100)"]].values
X
```

[15]:
```
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
       [ 19,  14],
       [ 19,  99],
       [ 20,  15],
       [ 20,  77],
       [ 20,  13],
       [ 20,  79],
       [ 21,  35],
```

```
[ 21,  66],
[ 23,  29],
[ 23,  98],
[ 24,  35],
[ 24,  73],
[ 25,   5],
[ 25,  73],
[ 28,  14],
[ 28,  82],
[ 28,  32],
[ 28,  61],
[ 29,  31],
[ 29,  87],
[ 30,   4],
[ 30,  73],
[ 33,   4],
[ 33,  92],
[ 33,  14],
[ 33,  81],
[ 34,  17],
[ 34,  73],
[ 37,  26],
[ 37,  75],
[ 38,  35],
[ 38,  92],
[ 39,  36],
[ 39,  61],
[ 39,  28],
[ 39,  65],
[ 40,  55],
[ 40,  47],
[ 40,  42],
[ 40,  42],
[ 42,  52],
[ 42,  60],
[ 43,  54],
[ 43,  60],
[ 43,  45],
[ 43,  41],
[ 44,  50],
[ 44,  46],
[ 46,  51],
[ 46,  46],
[ 46,  56],
[ 46,  55],
[ 47,  52],
[ 47,  59],
```

```
[ 48,  51],
[ 48,  59],
[ 48,  50],
[ 48,  48],
[ 48,  59],
[ 48,  47],
[ 49,  55],
[ 49,  42],
[ 50,  49],
[ 50,  56],
[ 54,  47],
[ 54,  54],
[ 54,  53],
[ 54,  48],
[ 54,  52],
[ 54,  42],
[ 54,  51],
[ 54,  55],
[ 54,  41],
[ 54,  44],
[ 54,  57],
[ 54,  46],
[ 57,  58],
[ 57,  55],
[ 58,  60],
[ 58,  46],
[ 59,  55],
[ 59,  41],
[ 60,  49],
[ 60,  40],
[ 60,  42],
[ 60,  52],
[ 60,  47],
[ 60,  50],
[ 61,  42],
[ 61,  49],
[ 62,  41],
[ 62,  48],
[ 62,  59],
[ 62,  55],
[ 62,  56],
[ 62,  42],
[ 63,  50],
[ 63,  46],
[ 63,  43],
[ 63,  48],
[ 63,  52],
```

```
[ 63,  54],
[ 64,  42],
[ 64,  46],
[ 65,  48],
[ 65,  50],
[ 65,  43],
[ 65,  59],
[ 67,  43],
[ 67,  57],
[ 67,  56],
[ 67,  40],
[ 69,  58],
[ 69,  91],
[ 70,  29],
[ 70,  77],
[ 71,  35],
[ 71,  95],
[ 71,  11],
[ 71,  75],
[ 71,   9],
[ 71,  75],
[ 72,  34],
[ 72,  71],
[ 73,   5],
[ 73,  88],
[ 73,   7],
[ 73,  73],
[ 74,  10],
[ 74,  72],
[ 75,   5],
[ 75,  93],
[ 76,  40],
[ 76,  87],
[ 77,  12],
[ 77,  97],
[ 77,  36],
[ 77,  74],
[ 78,  22],
[ 78,  90],
[ 78,  17],
[ 78,  88],
[ 78,  20],
[ 78,  76],
[ 78,  16],
[ 78,  89],
[ 78,   1],
[ 78,  78],
```

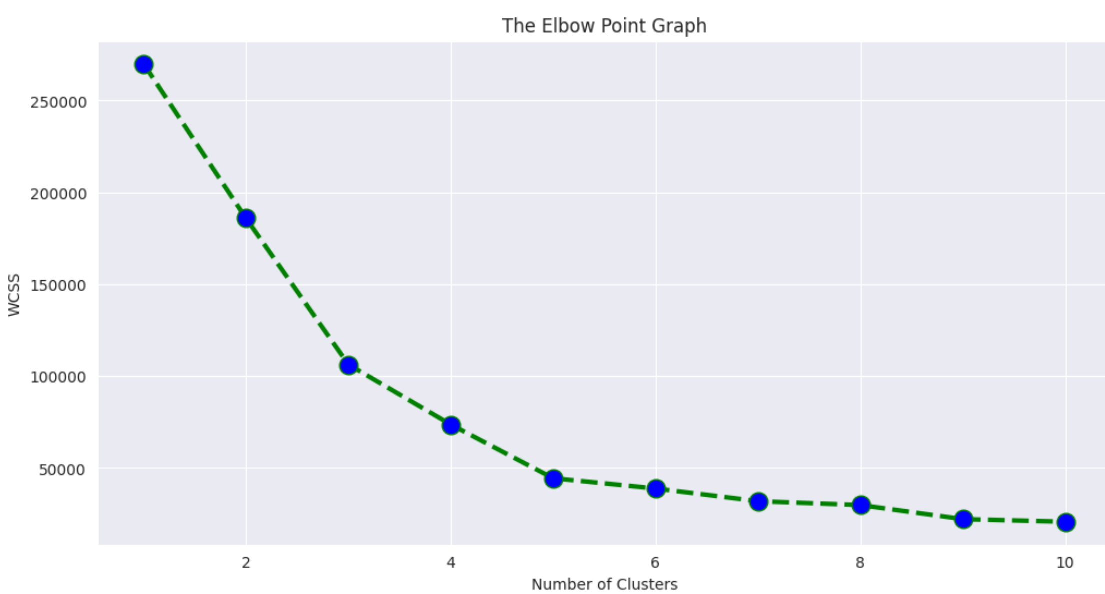
```
[ 78,   1],
[ 78,  73],
[ 79,  35],
[ 79,  83],
[ 81,   5],
[ 81,  93],
[ 85,  26],
[ 85,  75],
[ 86,  20],
[ 86,  95],
[ 87,  27],
[ 87,  63],
[ 87,  13],
[ 87,  75],
[ 87,  10],
[ 87,  92],
[ 88,  13],
[ 88,  86],
[ 88,  15],
[ 88,  69],
[ 93,  14],
[ 93,  90],
[ 97,  32],
[ 97,  86],
[ 98,  15],
[ 98,  88],
[ 99,  39],
[ 99,  97],
[101,  24],
[101,  68],
[103,  17],
[103,  85],
[103,  23],
[103,  69],
[113,   8],
[113,  91],
[120,  16],
[120,  79],
[126,  28],
[126,  74],
[137,  18],
[137,  83]])
```

3 Step 2: Perform Elbow Method To Find Optimal No.Of Clusters

[16]:
```
wcss = []
```

[17]:
```
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init="k-means++", random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

[18]:
```
plt.figure(figsize=(12, 6))
plt.grid()
plt.plot(
    range(1, 11),
    wcss,
    color="green",
    linestyle="dashed",
    linewidth=3,
    marker="o",
    markerfacecolor="blue",
    markersize=12,
)
plt.title("The Elbow Point Graph")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```



4 Training the K-Means Clustering Model

[19]:
```
kmeans = KMeans(n_clusters=5, init="k-means++")  # initialize the class object
label = kmeans.fit_predict(X)  # returns a cluster number for each of the data␣
 ↪points
print(label)
```

```
[1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1
 4 1 4 1 4 1 0 1 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 3 2 3 0 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 0 3 2 3 2 3
 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3]
```

5 Checking the centers of out clusters (Also known as Centroids)

[20]:
```
print(kmeans.cluster_centers_)
```

```
[[55.0875     49.7125    ]
 [26.30434783 20.91304348]
 [87.75       17.58333333]
 [86.53846154 82.12820513]
 [25.72727273 79.36363636]]
```

6 Visualizing all the clusters

[21]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 8))

# Scatter plot for 5 clusters
plt.scatter(X[label == 0, 0], X[label == 0, 1], s=50, c="green", label="Cluster␣
 ↪1")
plt.scatter(X[label == 1, 0], X[label == 1, 1], s=50, c="yellow",␣
 ↪label="Cluster 2")
plt.scatter(X[label == 2, 0], X[label == 2, 1], s=50, c="red", label="Cluster␣
 ↪3")
plt.scatter(X[label == 3, 0], X[label == 3, 1], s=50, c="purple",␣
 ↪label="Cluster 4")
plt.scatter(X[label == 4, 0], X[label == 4, 1], s=50, c="blue", label="Cluster␣
 ↪5")

# Scatter plot for cluster centers
plt.scatter(
    kmeans.cluster_centers_[:, 0],
```

```
    kmeans.cluster_centers_[:, 1],
    s=100,
    c="black",
    label="Centroids",
    marker="*",
)

plt.title("Customer groups")
plt.xlabel("Annual Income")
plt.ylabel("Spending Score (1-100)")
plt.legend()

plt.show()
```