Clustering_

December 9, 2022

1 unsuperised machine learning/ clustering

Dataset : https://github.com/tirthajyoti/Machine-Learning-with-Python/blob/master/Datasets/Mall_Customers.csv

[1]:
```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
# Data ingestion
df = pd.read_csv("https://github.com/NelakurthiSudheer/
↪Mall-Customers-Segmentation/raw/main/Dataset/Mall_Customers.csv")
df.head()
```

[2]:

	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

[3]:
```
df.drop(["CustomerID"], axis=1, inplace=True)
```

[4]:
```
df.head()
```

[4]:

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

[5]:
```
df.shape
```

[5]: (200, 4)

[6]:
```
df.isnull().sum()
```

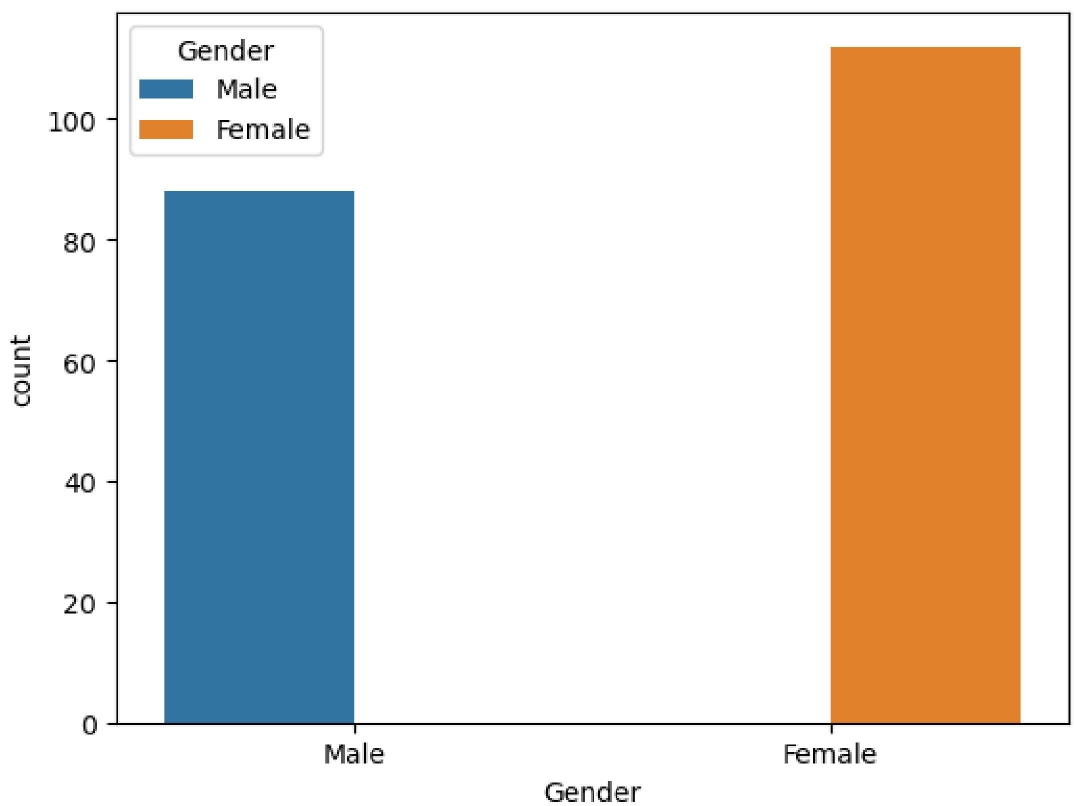
[6]:
```
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

[7]:
```
df["Gender"].value_counts()
```

[7]:
```
Female    112
Male       88
Name: Gender, dtype: int64
```

[8]:
```
sns.countplot(data=df, x="Gender", hue="Gender")
```

[8]:
```
<AxesSubplot: xlabel='Gender', ylabel='count'>
```



[9]:
```
df.Age.max(), df.Age.min()
```

[9]:
```
(70, 18)
```

[10]:

```
def grouping(x):
    if x>10 and x<=20:
        return "10-20"
    elif x>20 and x<=30:
        return "20-30"
    elif x>30 and x<=40:
        return "30-40"
    elif x>40 and x<=50:
        return "40-50"
    elif x>50 and x<=60:
        return "50-60"
    elif x>70 and x<=80:
        return "70-80"
    elif x>80 and x<=90:
        return "80-90"
```

[11]:

```
df["Age_group"] = df.Age.apply(lambda x: grouping(x))
```

[12]:

```
df.head()
```

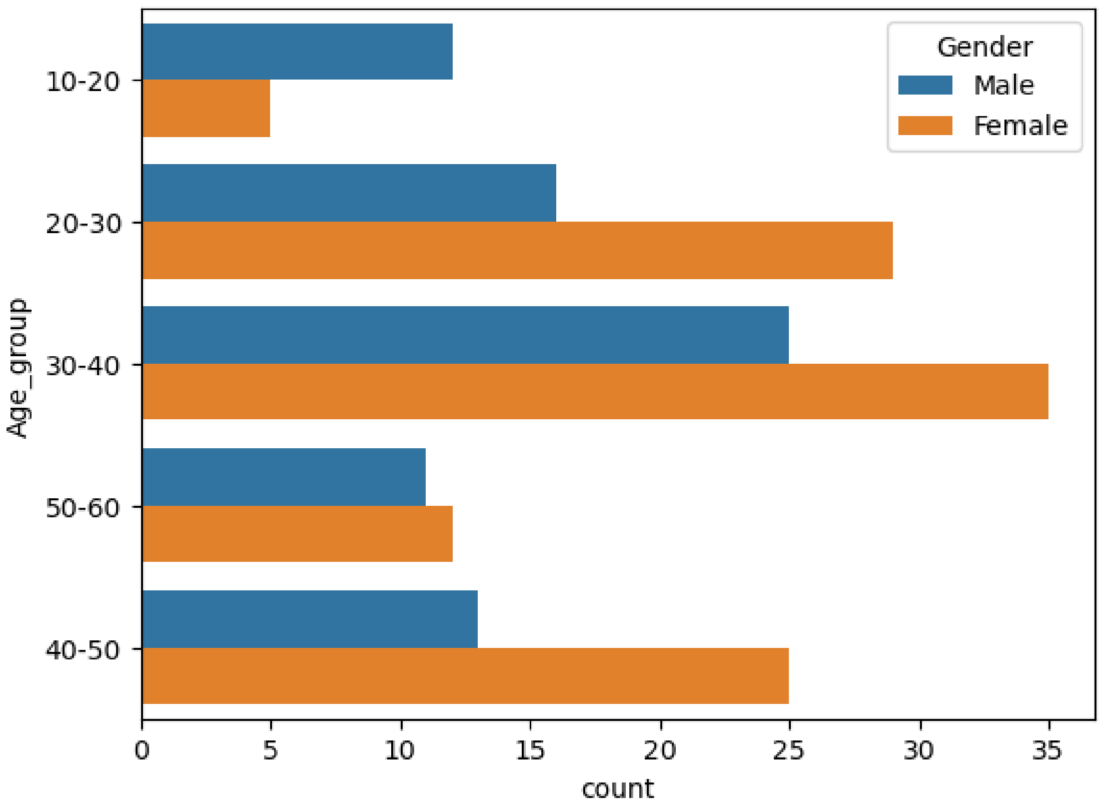
[12]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Age_group
0	Male	19	15	39	10-20
1	Male	21	15	81	20-30
2	Female	20	16	6	10-20
3	Female	23	16	77	20-30
4	Female	31	17	40	30-40

[13]:

```
sns.countplot(data=df, y="Age_group", hue="Gender")
```

[13]: <AxesSubplot: xlabel='count', ylabel='Age_group'>



2 Bi-variate Clustering

2.1 K-means algorithm

[14]:
```
# importing libreries
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

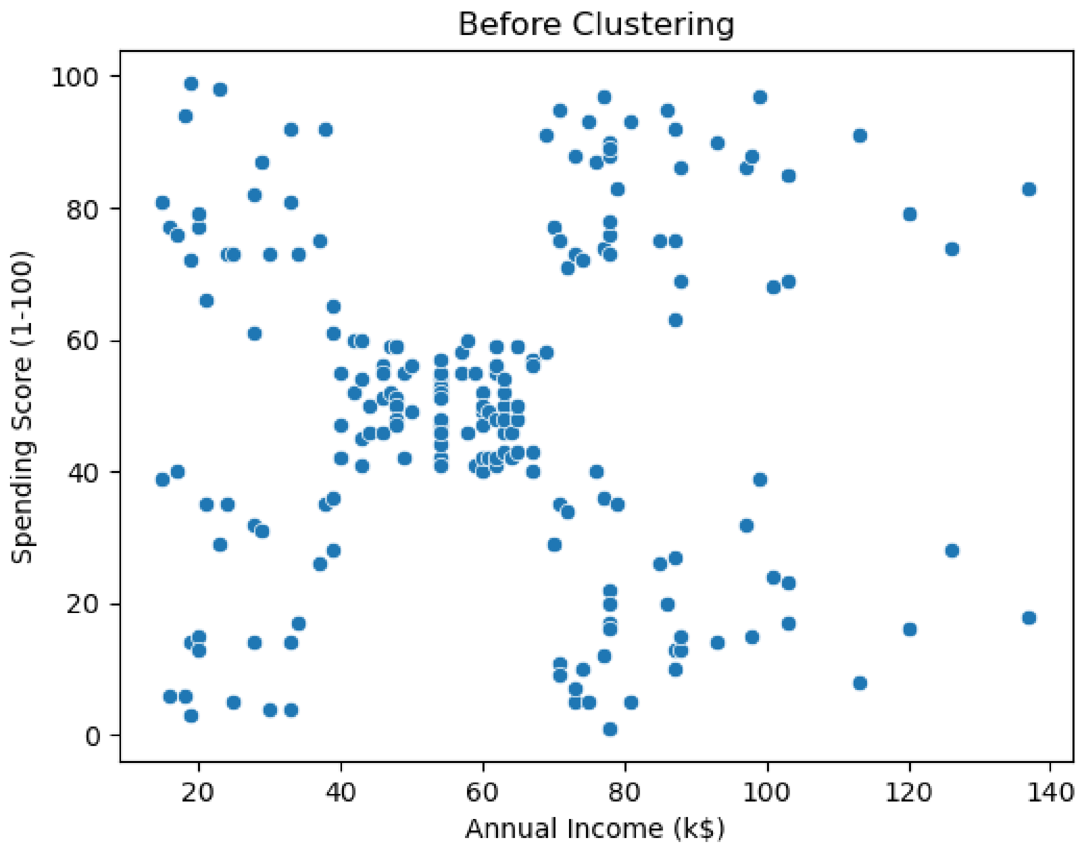
[15]:
```
# clustering datase preperation
data = df.iloc[:,-3:-1]
data.head()
```

[15]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

[16]:
```
sns.scatterplot(data=data, x="Annual Income (k$)", y="Spending Score (1-100)")
plt.title("Before Clustering")
```

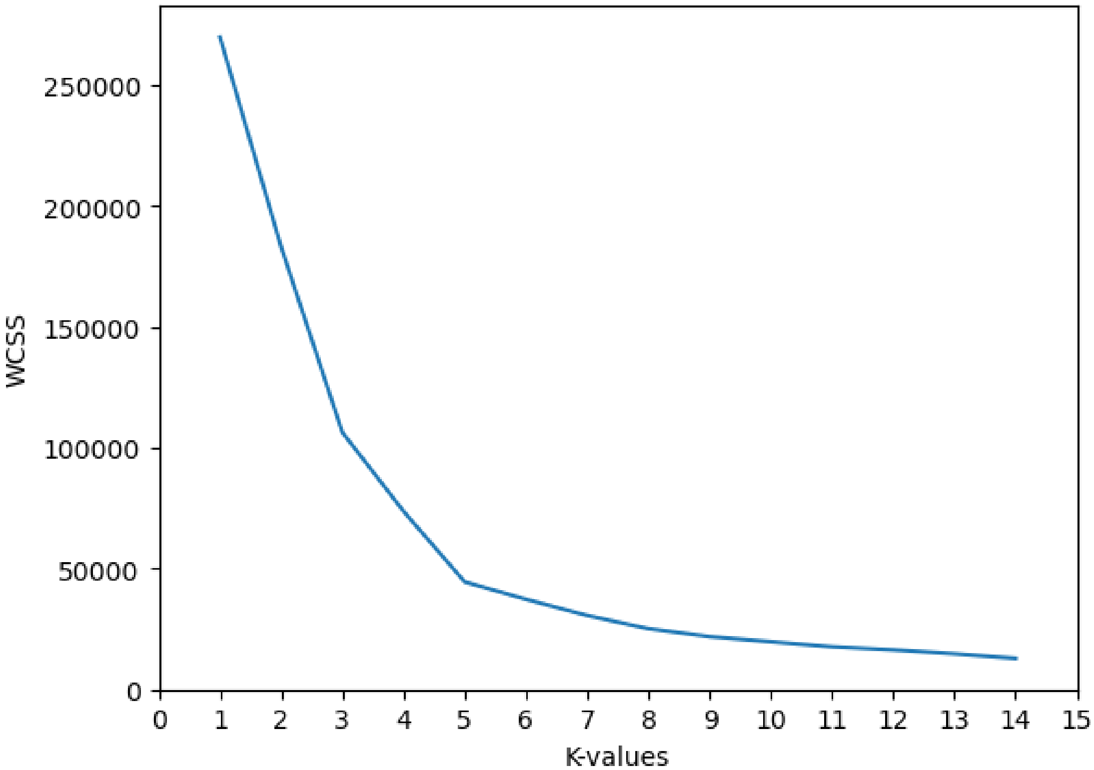
[16]:
```
Text(0.5, 1.0, 'Before Clustering')
```



[17]:
```
X = data.to_numpy()
```

[18]:
```
wcss=[]
for i in range(1,15):
    kmeans = KMeans(n_clusters=i, init= 'k-means++')
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

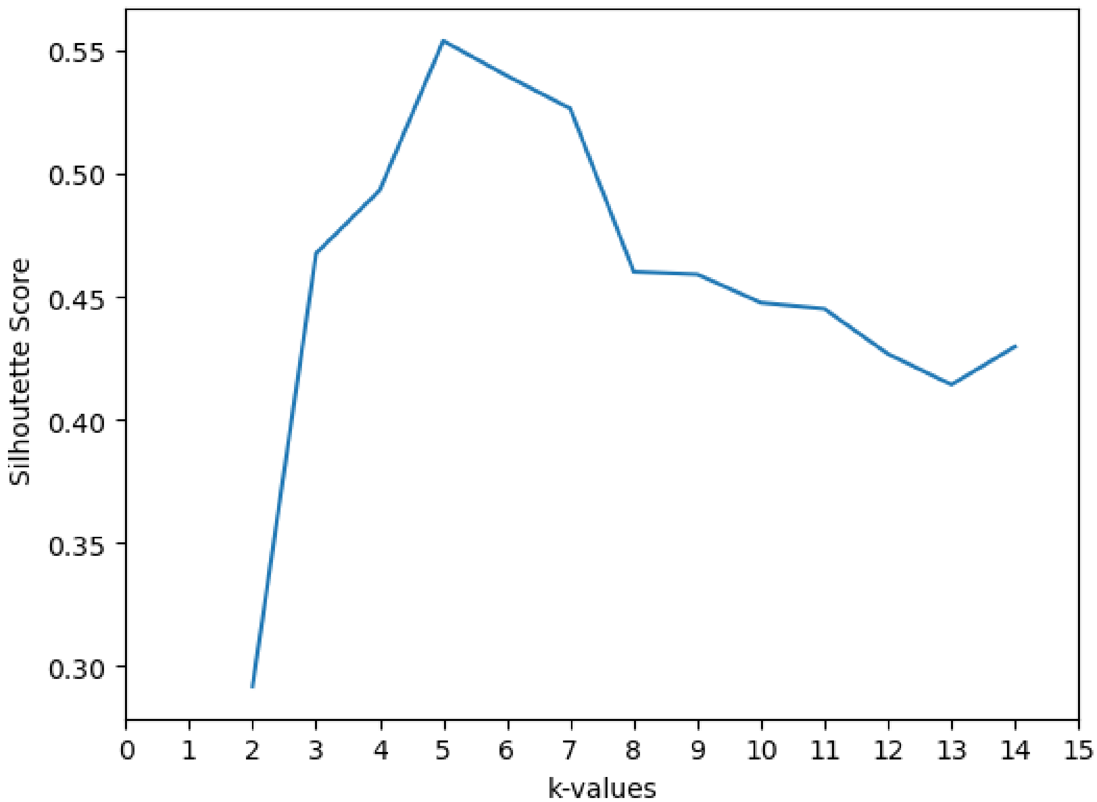
[19]:
```
# ELBOW curve
plt.plot(range(1,15), wcss)
plt.xticks(range(0,16))
plt.xlabel("K-values")
plt.ylabel("WCSS")
plt.show()
```



2.2 unable to conclude the Optimal value of k using ELBOW curve

[20]:
```
silh=[]
for i in range(2,15):
    kmeans = KMeans(n_clusters=i, init= 'k-means++')
    kmeans.fit(X)
    labels =  kmeans.labels_
    silh.append(silhouette_score(X, labels, metric='euclidean'))
```

[21]:
```
# Silhouette Value-versus-k plot.
plt.plot(range(2,15), silh)
plt.xticks(range(0,16))
plt.xlabel("k-values")
plt.ylabel("Silhoutette Score ")
plt.show()
```



2.3 Global maxima at k=5

2.4 Optimal value of k is 5

[22]:
```
best_kmeans_model = KMeans(n_clusters=5, init= 'k-means++')
```

[23]:
```
best_kmeans_model.fit(X)
```

[23]:
```
KMeans(n_clusters=5)
```

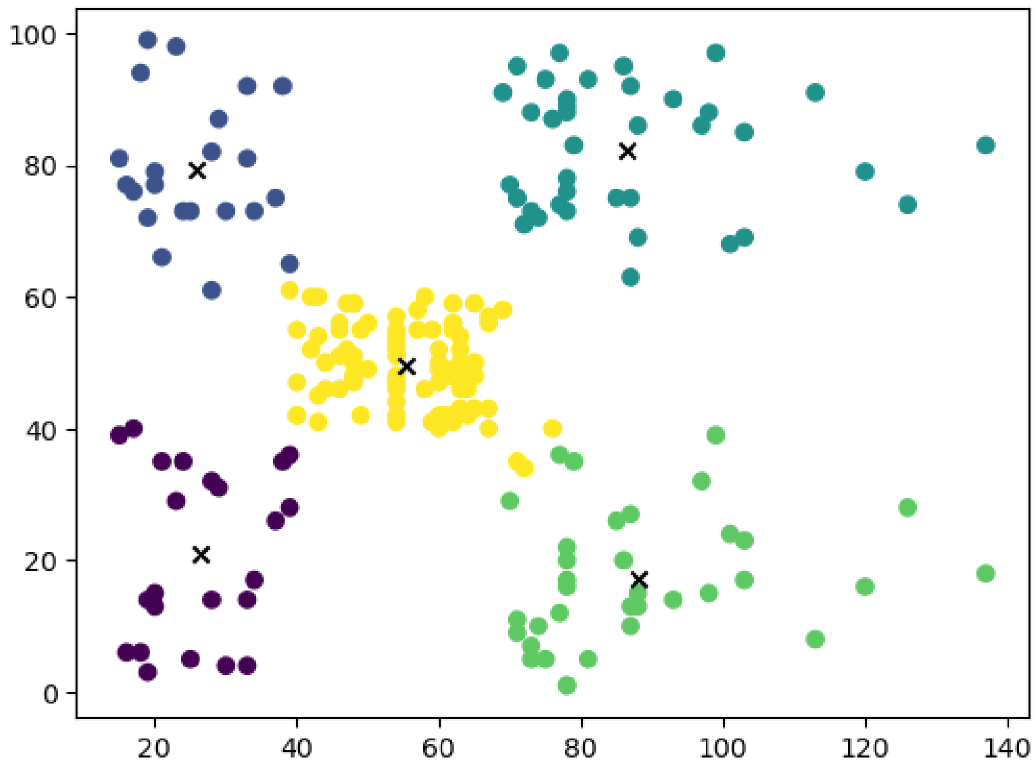
2.4.1 Plot of the Clusters

[25]:
```
labels = best_kmeans_model.labels_
```

[24]:
```
centroids = best_kmeans_model.cluster_centers_
```

[42]:
```
plt.scatter(x=data["Annual Income (k$)"], y=data["Spending Score (1-100)"],␣
 ↪c=labels)
plt.scatter(x=centroids[:,0], y=centroids[:,1], c='black', marker='x')
plt.title("After Clustering")
```

[42]: <matplotlib.collections.PathCollection at 0x7f1ec2a0afe0>



[44]:
```
silhouette_score(X,labels)
```

[44]: 0.553931997444648

2.4.2 Accuracy obtained by using K-means method = 55%

2.5 Hierarchical Clustering

[66]:
```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram
```

[72]:
```
hier_clus = AgglomerativeClustering(distance_threshold=0, n_clusters=None,␣
 ↪affinity='euclidean', linkage='ward')
```

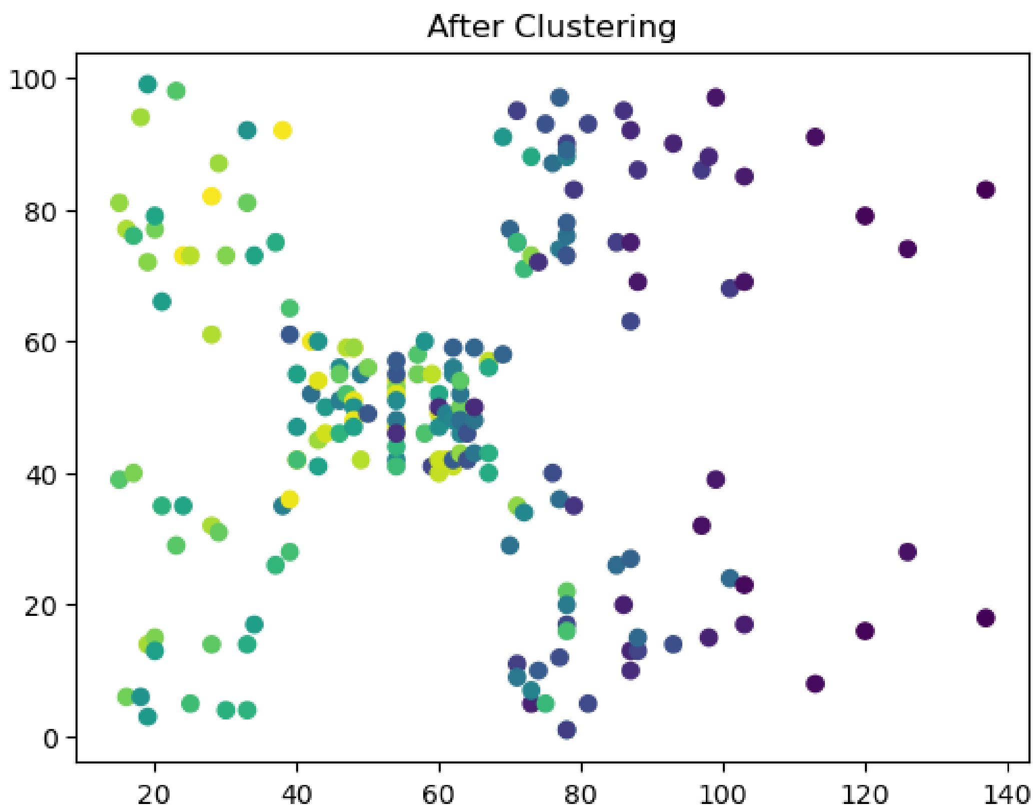
[74]:
```
hier_clus = hier_clus.fit(X)
```

[82]:
```
labels=hier_clus.fit_predict(X)
```

[83]:
```
plt.scatter(x=data["Annual Income (k$)"], y=data["Spending Score (1-100)"],␣
 ↪c=labels)
```

```python
# plt.scatter(x=centroids[:,0], y=centroids[:,1], c='black', marker='x')
plt.title("After Clustering")
```

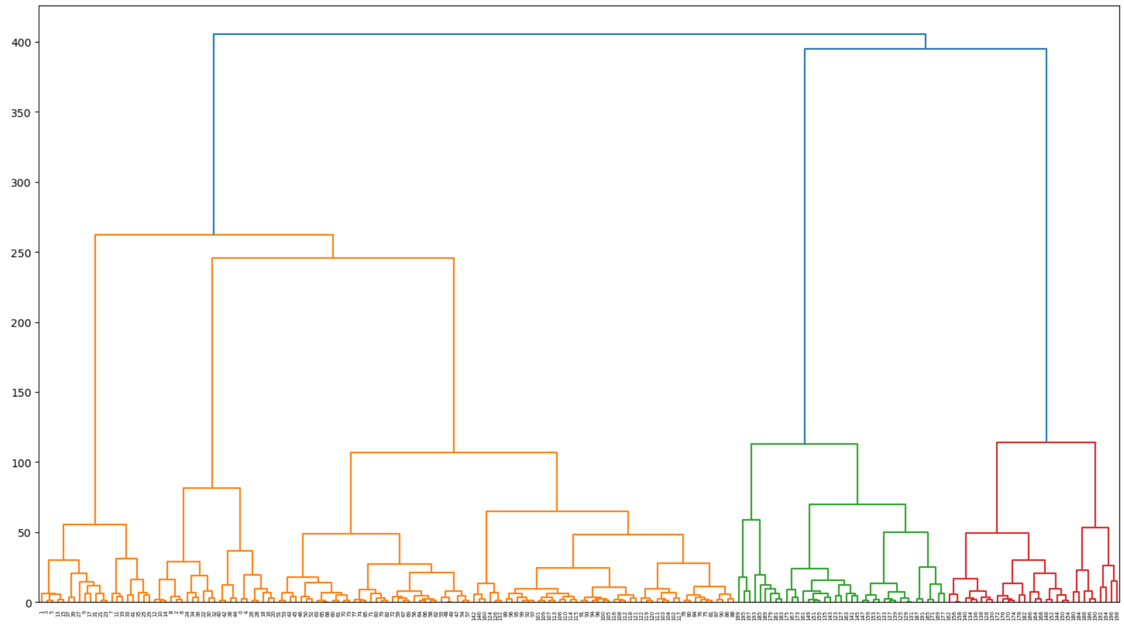
[83]: Text(0.5, 1.0, 'After Clustering')



[84]:
```python
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
```

```
        [model.children_, model.distances_, counts]
    ).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

[88]:
```
plt.figure(figsize=(18,10))
plot_dendrogram(hier_clus, truncate_mode="level")
```



[ ]:
```
```