8.2. Hierarchical Clustering

Like K Means clustering, hierarchical clustering is another commonly used unsupervised machine learning technique for data clustering.

Hierarchical clustering can be broadly divided into two types: agglomerative clustering and divisive clustering. Agglomerative clustering follows a bottom-up approach, where individual data points are clustered together to form multiple small clusters leading to a big cluster, which can then be divided into small clusters using dendrograms. On the other hand, in the case of divisive clustering, you have one big cluster, which you divide into N number of small clusters.

In this chapter, you will perform agglomerative clustering using the Sklearn library.

Steps for Hierarchical Agglomerative Clustering

The steps required to perform agglomerative clustering are as follows:

1. Consider each data point in the dataset as one cluster. Hence, the number of in the beginning is equal to the number of data points.

2. Join the two closest data points to form a cluster.

3. Form more clusters by joining the closest clusters. Repeat this process until cluster is formed.

4. Use dendrograms to divide the one big cluster into multiple small clusters. (
concept of dendrograms is explained later in the chapter.)

Why Use Hierarchical Clustering?

Hierarchical clustering has the following advantages:

1. Unlike K Means clustering, for hierarchical clustering, you do not have to s
the number of centroids clustering.

2. With dendrograms, it is easier to interpret how data has been clustered.

Disadvantages of Hierarchical Clustering Algorithm

The following are some of the disadvantages of the hierarchical clustering algorithm:

1. Doesn't scale well on unseen data.

2. Has higher time complexity compared to K Means clustering.

3. Difficult to determine the number of clusters in case of a large dataset.

In the next section, you will see how to perform agglomerative clustering via Sklearn.

8.2.1. Clustering Dummy Data

First, we will see how to perform hierarchical clustering on dummy data, and then we will perform hierarchical clustering on Iris data.

Example 1

In the first example, we will perform agglomerative clustering of 10 2-dimensional data points only.

The following script imports the required libraries:

Script 17:

```
import numpy as np
import pandas as pd
from sklearn.datasets.samples_generator import make_blobs
from matplotlib import pyplot as plt
```

```
%matplotlib inline
```

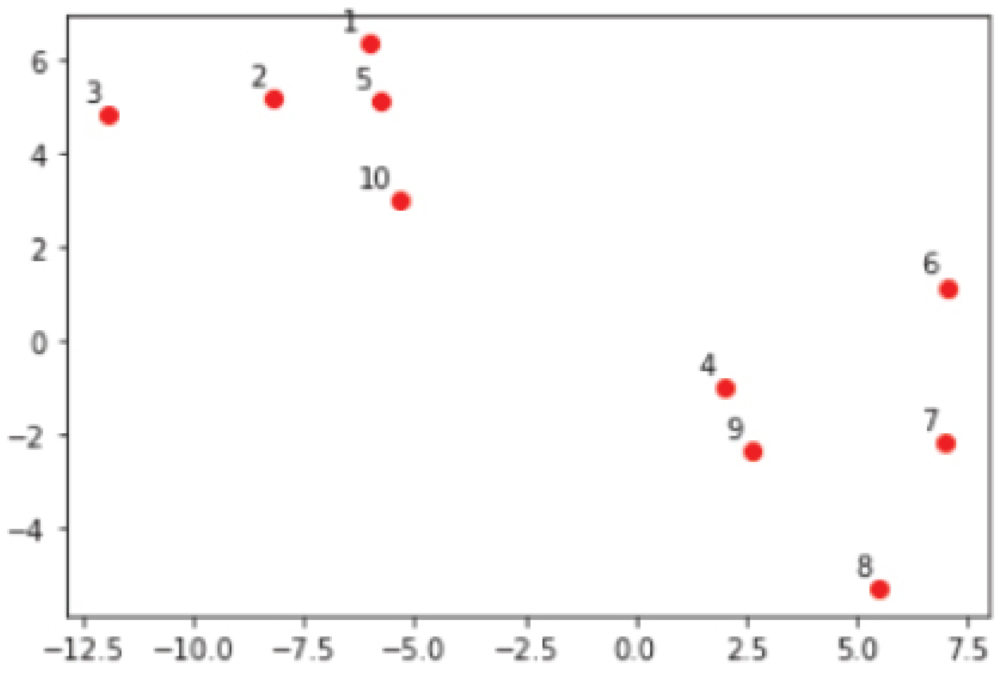
The following script randomly creates data points and then labels the data points from 1 to 10. The data points are plotted as a scatter plot.

Script 18:

```
# generating dummy data of 10 records with 2 clusters
features, labels = make_blobs(n_samples=10, centers=2, cluster_std = 2.00)

#plotting the dummy data
plt.scatter(features[:,0], features[:,1], color ='r' )

#adding numbers to data points
annots = range(1, 11)
for label, x, y in zip(annots, features[:, 0], features[:, 1]):
        plt.annotate(
                label,
                xy=(x, y), xytext=(-3, 3),
                textcoords='offset points' , ha='right' , va='bottom' )
plt.show()
```

The output is as follows. From the output below, it can be clearly seen that the data points 1, 2, 3, 5, and 10 belong to one cluster and the data points 4, 6, 7, 8, and 9 belong to the other cluster.

Output:



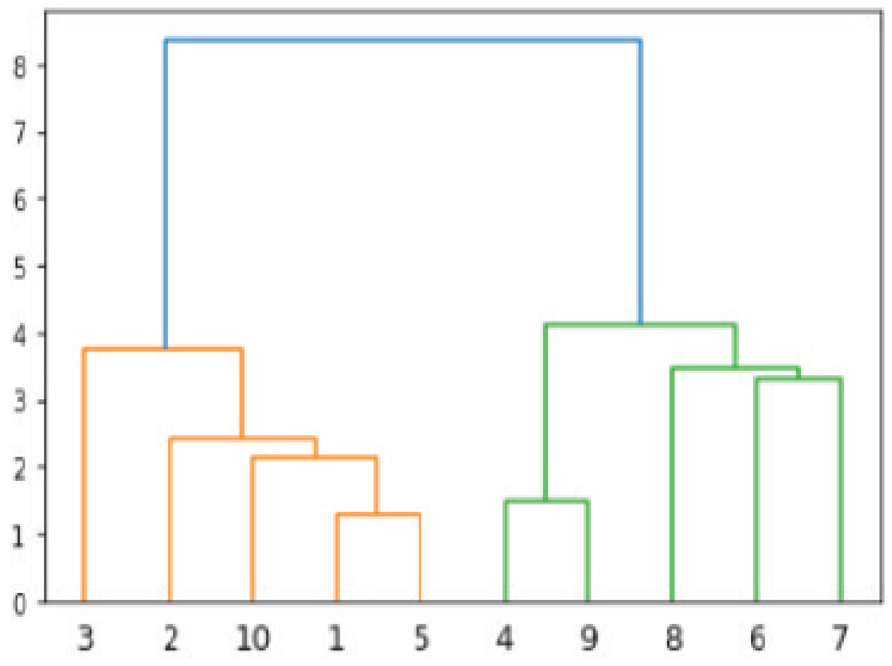
Let’s now plot dendrograms for the above 10 data points. To plot dendrograms, you can use the dendrogram and linkage classes from the scipy.cluster.hierarchy module. The features are passed to the linkage class. And the object of the linkage class is passed to the dendrogram class to plot dendrogram for the features, as shown in the following script:

Script 19:

```
from scipy.cluster.hierarchy import dendrogram, linkage

dendos = linkage(features, ‘single’ )

annots = range(1, 11)

dendrogram(dendos,
                    orientation=’top’ ,
                     labels=annots,
                     distance_sort=’descending’ ,
                    show_leaf_counts=True)
plt.show()
```

Here is the output of the above script.

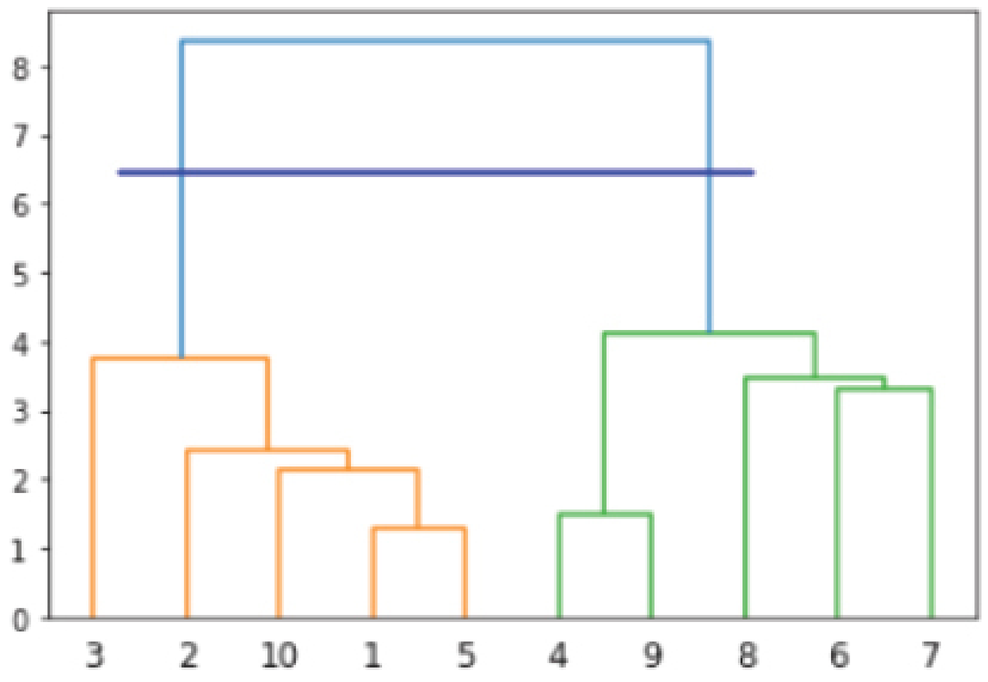
Output:



From the figure above, it can be seen that points 1 and 5 are closest to each other. Hence, a cluster is formed by connecting these points. The cluster of 1 and 5 is closest to data point 10, resulting in a cluster containing points 1, 5, and 10. In the same way, the remaining clusters are formed until a big cluster is formed.

After a big cluster is formed, select the longest vertical line. Then, draw a horizontal line through it. The number of clusters formed is equal to the number of vertical lines this newly created horizontal line passes.

For instance, in the following figure, two clusters are formed.



In real world scenarios, there can be thousands of data points, and hence, the dendrogram method cannot be used to manually cluster the data. This is where we can use the AgglomerativeClustering class from the sklearn.cluster module. The number of clusters and the distance types are passed as parameters to the AgglomerativeClustering class.

The following script applies agglomerative clustering to our dummy dataset.

Script 20:

```
from sklearn.cluster import AgglomerativeClustering

# training agglomerative clustering model
hc_model = AgglomerativeClustering(n_clusters=2, affinity='euclidean' , linkage='ward' )
hc_model.fit_predict(features)
```

Output:

```
array([0, 0, 0, 1, 0, 1, 1, 1, 1, 0], dtype=int64)
```

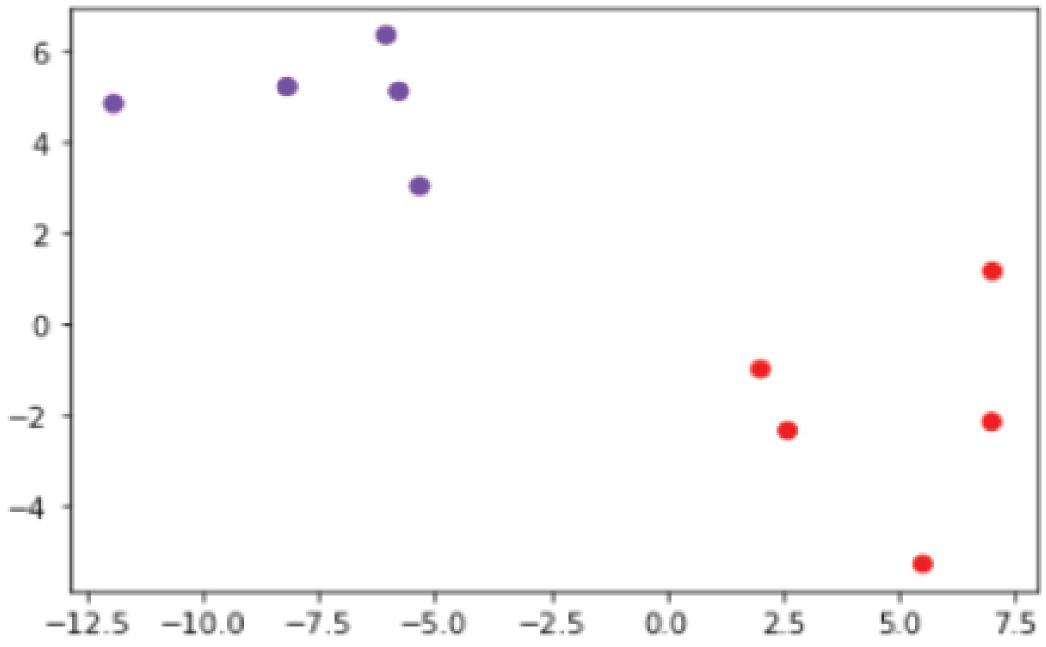
And the following script plots the predicted clusters.

Script 21:

```
#pring the data points
plt.scatter(features[:,0], features[:,1], c= hc_model.labels_, cmap='rainbow' )
```

The output shows that our clustering algorithm has successfully clustered the data points.

Output:



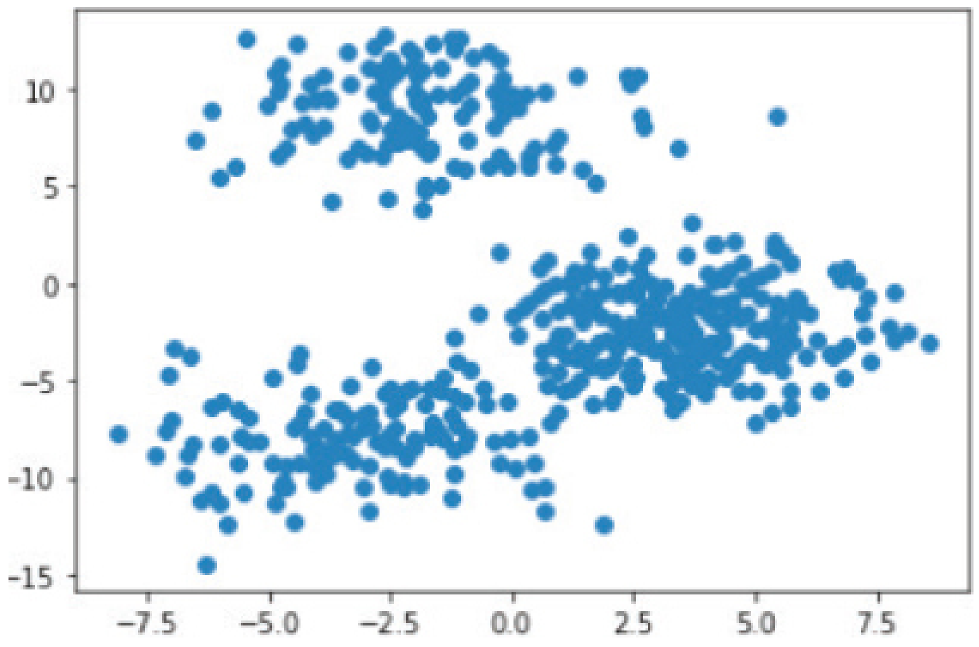
Example 2

In the previous example, we had 10 data points with 2 clusters. Let's now see an example with 500 data points. The following script creates 500 data points with 4 cluster centers.

Script 22:

```
# generating dummy data of 500 records with 4 clusters
features, labels = make_blobs(n_samples=500, centers=4, cluster_std = 2.00)

#plotting the dummy data
plt.scatter(features[:,0], features[:,1] )
```

Output:



The following script applies agglomerative hierarchical clustering on the dataset. The number of predicted clusters is 4.

Script 23:

```
# performing kmeans clustering using AgglomerativeClustering class
hc_model = AgglomerativeClustering(n_clusters=4, affinity= ’euclidean’ , linkage=’ward’ )
hc_model.fit_predict(features)
```

The output shows the labels of some of the data points in our dataset. You can see that since there are 4 clusters, there are 4 unique labels, i.e., 0, 1, 2, and 3.

Output:

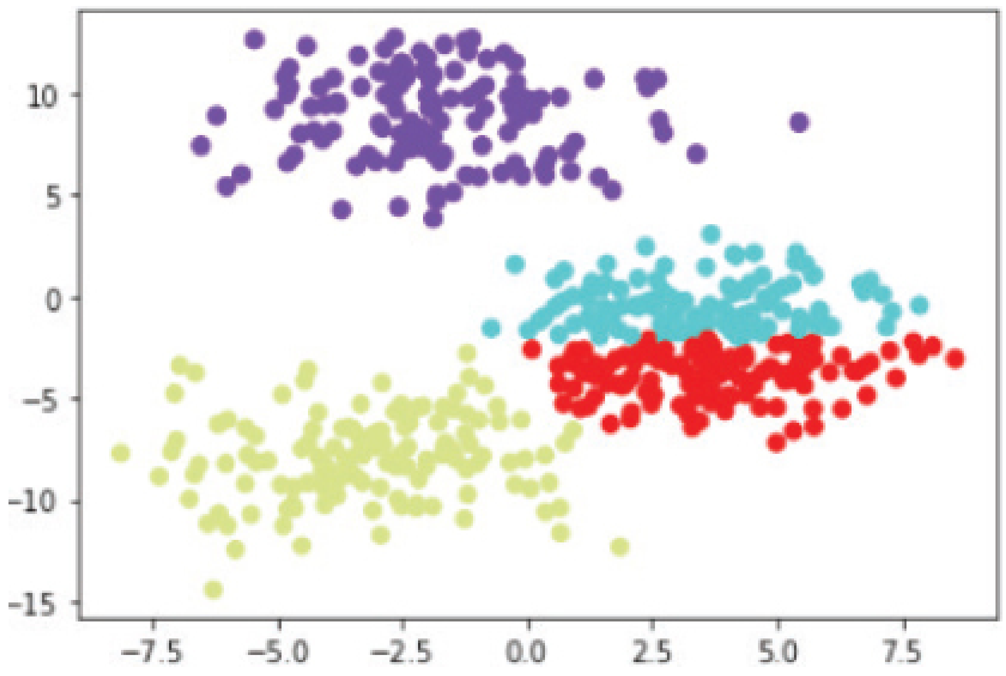
```
array([0, 1, 1, 0, 1, 0, 3, 0, 0, 1, 0, 0, 1, 3, 0, 2, 0, 3, 1, 0, 0, 0,], dtype=int64)
```

To plot the predicted clusters, execute the following script.

Script 24:

```
#pring the data points
plt.scatter(features[:,0], features[:,1], c= hc_model.labels_, cmap=’rainbow’ )
```

Output:

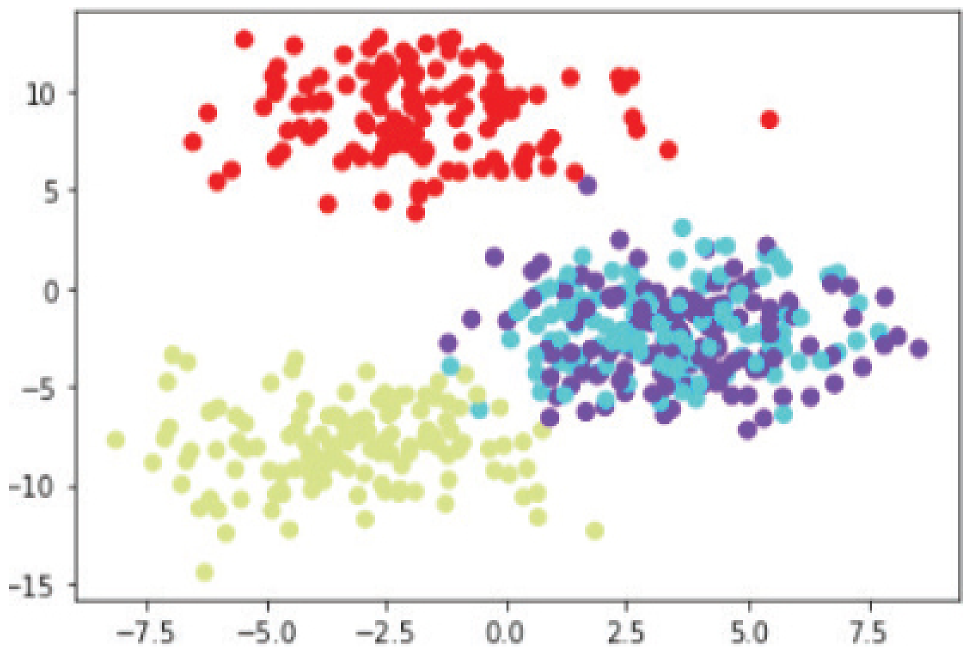


Similarly, to plot the actual clusters in the dataset (for the sake of comparison), execute the following script.

Script 25:

```
#print actual datapoints
plt.scatter(features[:,0], features[:,1], c= labels, cmap='rainbow' )
```

Output:



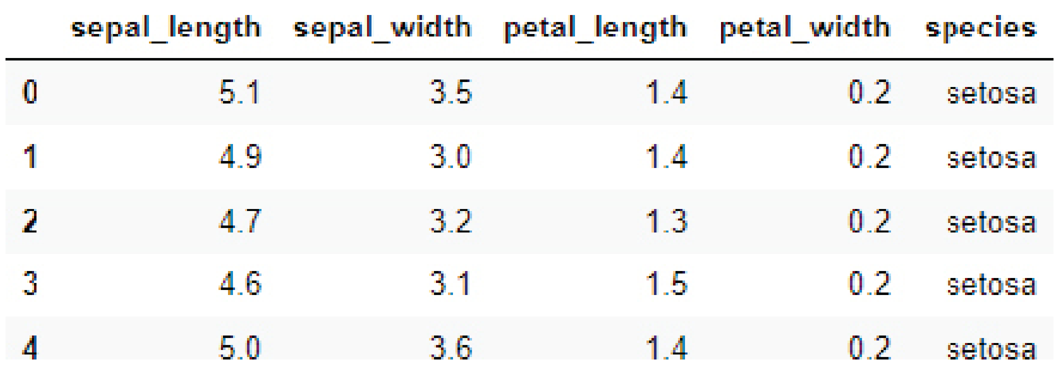
8.2.2. Clustering the Iris Dataset

In this section, you will see how to cluster the Iris dataset using hierarchical agglomerative clustering. The following script imports the Iris dataset and displays the first five rows of the dataset.

Script 26:

```
import seaborn as sns

iris_df = sns.load_dataset("iris" )
iris_df.head()
```

Output:



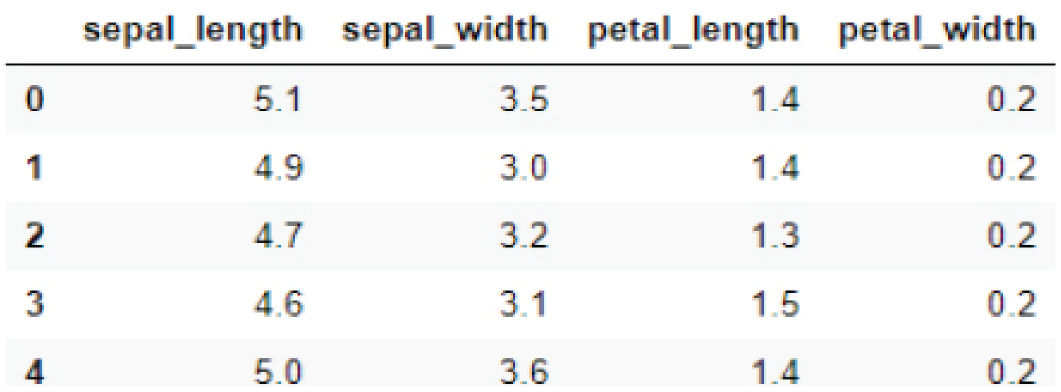
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

The following script divides the data into features and labels sets and displays the first five rows of the labels set.

Script 27:

```
# dividing data into features and labels
features = iris_df.drop([“species” ], axis = 1)
labels = iris_df.filter([“species” ], axis = 1)
features.head()
```

Output:



	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Similarly, the following script applies the agglomerative clustering on the feature set using the AgglomerativeClustering class from the sklearn.cluster module.

Script 28:

```
# training Hierarchical clustering model
from sklearn.cluster import AgglomerativeClustering

# training agglomerative clustering model
features = features.values
hc_model = AgglomerativeClustering(n_clusters=3, affinity=’euclidean’ , linkage=’ward’ )
hc_model.fit_predict(features)
```

The output below shows the predicted cluster labels for the feature set in the Iris dataset.

Output:

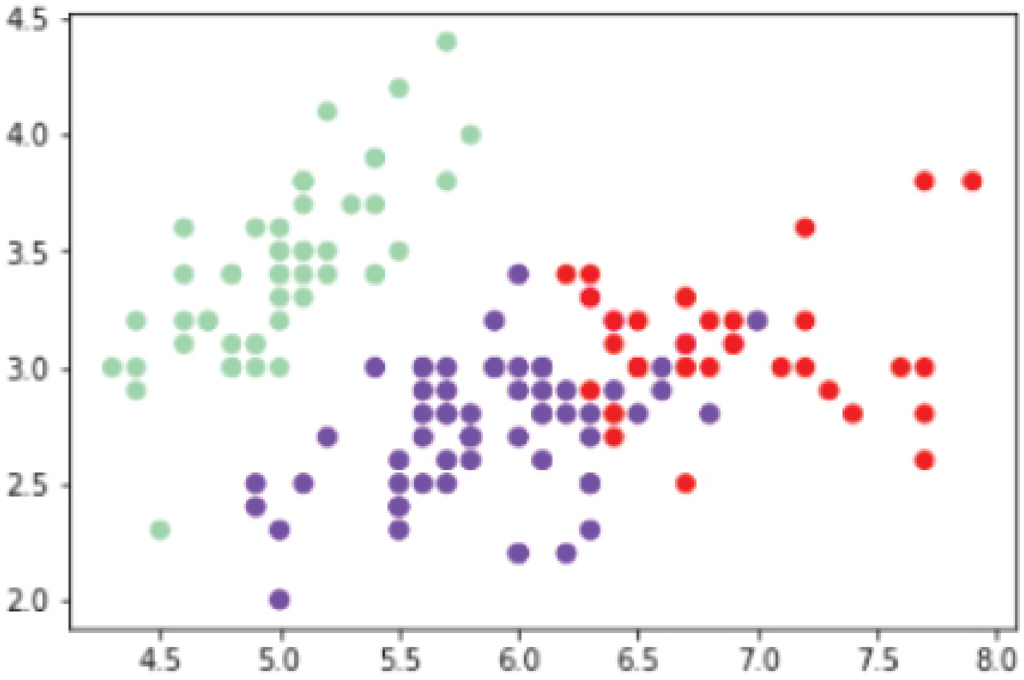
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0,
       2, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0],
       dtype=int64)
```

The predicted clusters are printed via the following script.

Script 29:

```
#pring the data points
plt.scatter(features[:,0], features[:,1], c= hc_model.labels_, cmap='rainbow' )
```

Output:



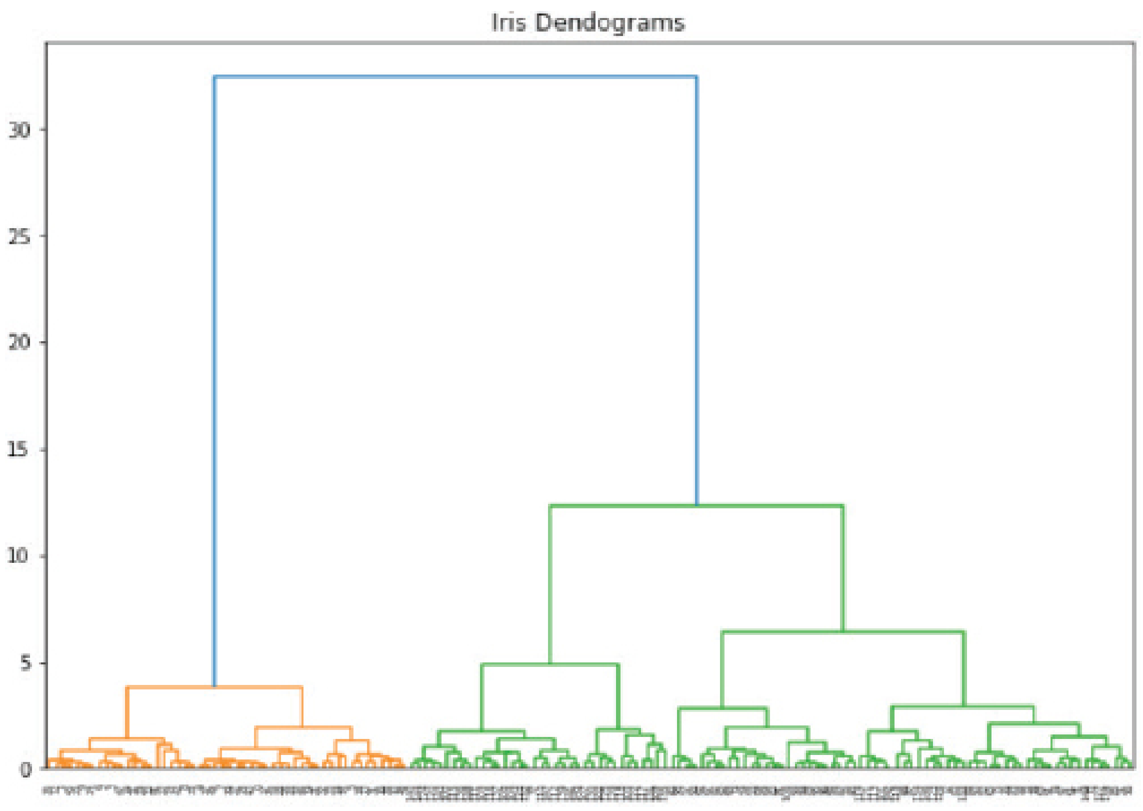
You can also create dendrograms using the feature set using the shc module from the scipy.cluster.hierarchy library. You have to pass the feature set to the linkage class of the shc module, and then the object of the linkage class is passed to the dendrogram class to plot the dendrograms, as shown in the following script.

Script 30:

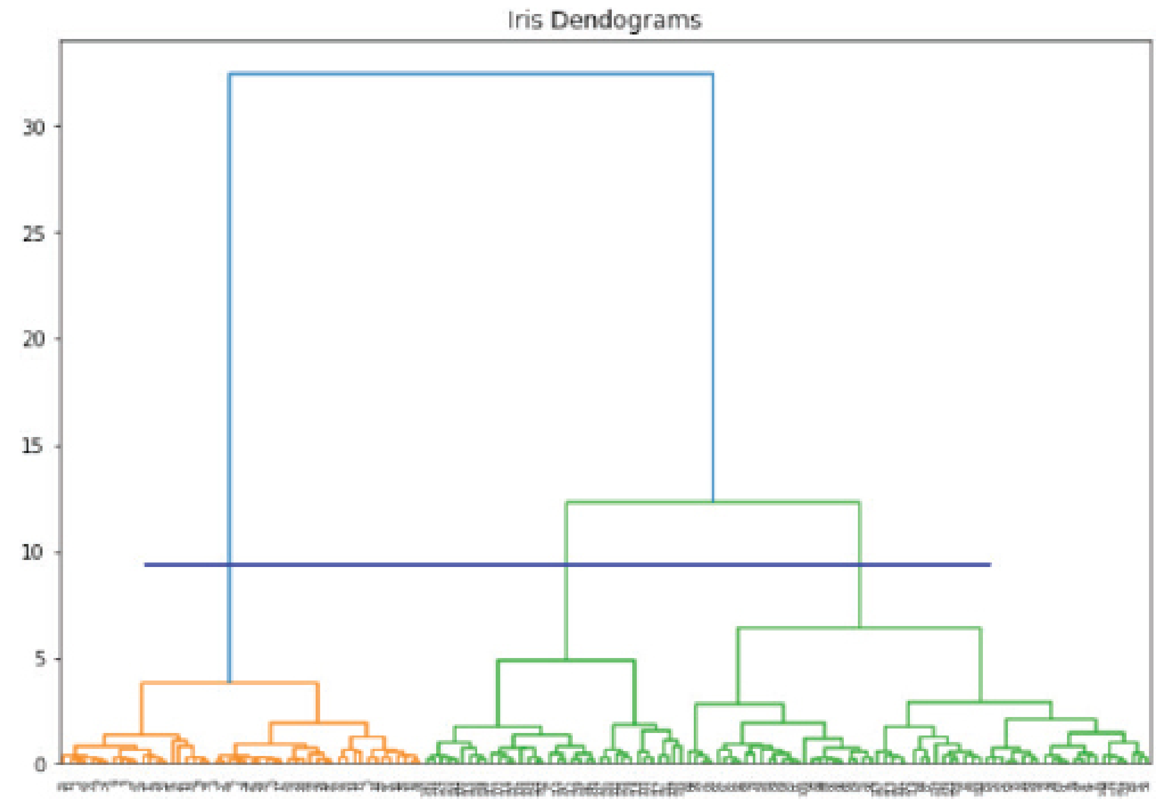
```
import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title(“Iris Dendograms” )
dend = shc.dendrogram(shc.linkage(features, method=’ward’ ))
```

Here is the output of the script above.

Output:



If you want to cluster the dataset into three clusters, you can simply draw a horizontal line that passes through the three vertical lines, as shown below. The clusters below the horizontal line are the resultant clusters. In the following figure, we form three clusters.



Iris Dendograms
30
25
20
15
10
5
0