Hierarchical-Clustering

January 14, 2025

[ ]:
```
'''
   Hierarchical Clustering -->

   Hierarchical clustering is a method of cluster analysis that seeks to build
   a hierarchy of clusters. It is widely used in data mining and statistics
   to group objects based on their similarities.

   This approach can be divided into two main types -->

   #   Agglomerative Hierarchical Clustering (Bottom-Up Approach) -->

       Starts with each data point as its own cluster.
       Gradually merges the closest clusters step by step until all points␣
 ↪belong to a single cluster.
       Commonly visualized using a dendrogram

   #   Divisive Hierarchical Clustering (Top-Down Approach) -->

       Starts with all data points in a single cluster.
       Splits the cluster into smaller clusters recursively.
       Continues until each data point is in its own cluster or a stopping␣
 ↪criterion is met.
'''
```

[ ]:
```
'''
   Steps (Agglomerative HC) ->

   Compute the distance (or similarity) matrix for all pairs of points.
   Identify the two closest clusters and merge them.
   Recompute the distance matrix considering the newly formed cluster.
   Repeat steps 2 and 3 until only one cluster remains.

   Linkage Criteria ->

   Single Linkage: Minimum distance between points in two clusters.
   Complete Linkage: Maximum distance between points in two clusters.
   Average Linkage: Average distance between points in two clusters.
```

```
    Centroid Linkage: Distance between centroids of two clusters.
'''
```

[ ]:
```
'''
    Advantages -->

    Does not require specifying the number of clusters in advance.
    Can produce a dendrogram that visually represents the data's structure.
    Suitable for small to medium-sized datasets.

    Disadvantages -->

    Computationally expensive for large datasets (time complexity: O(n^3)).
    Sensitive to noise and outliers.
    Requires careful selection of linkage criteria and distance metric.

    Applications -->

    Biology: Grouping species or genes with similar characteristics.
    Market Segmentation: Grouping customers with similar buying patterns.
    Image Analysis: Identifying similar regions in an image.
'''
```

[ ]:
```
'''
    Dendogram -->

    A dendrogram is a tree-like diagram used to represent the arrangement of␣
↪clusters
    formed through hierarchical clustering. It visually shows the steps in␣
↪which clusters
    are merged or split. Each branch of the tree represents a cluster, and the␣
↪height
    at which two branches merge indicates the distance or dissimilarity between␣
↪the clusters.

    Parts of a Dendrogram -->

    Leaves (Bottom): Represent individual data points or initial clusters.
    Branches (Middle): Show how clusters are merged at each step.
    Height (Vertical Axis): Indicates the distance (or dissimilarity) between␣
↪clusters when they are merged.
    Horizontal Axis: Represents the data points or their order.

    How to Interpret a Dendrogram -->

    Horizontal lines: Represent cluster merging.
```

```
    The height of these lines shows the distance or dissimilarity at which␣
↪clusters are joined.
    Vertical lines: Connect data points or clusters to the merging step.
    Cut the dendrogram: To determine the clusters, draw a horizontal line␣
↪across the dendrogram.
    The number of vertical lines intersected gives the number of clusters.
'''
```

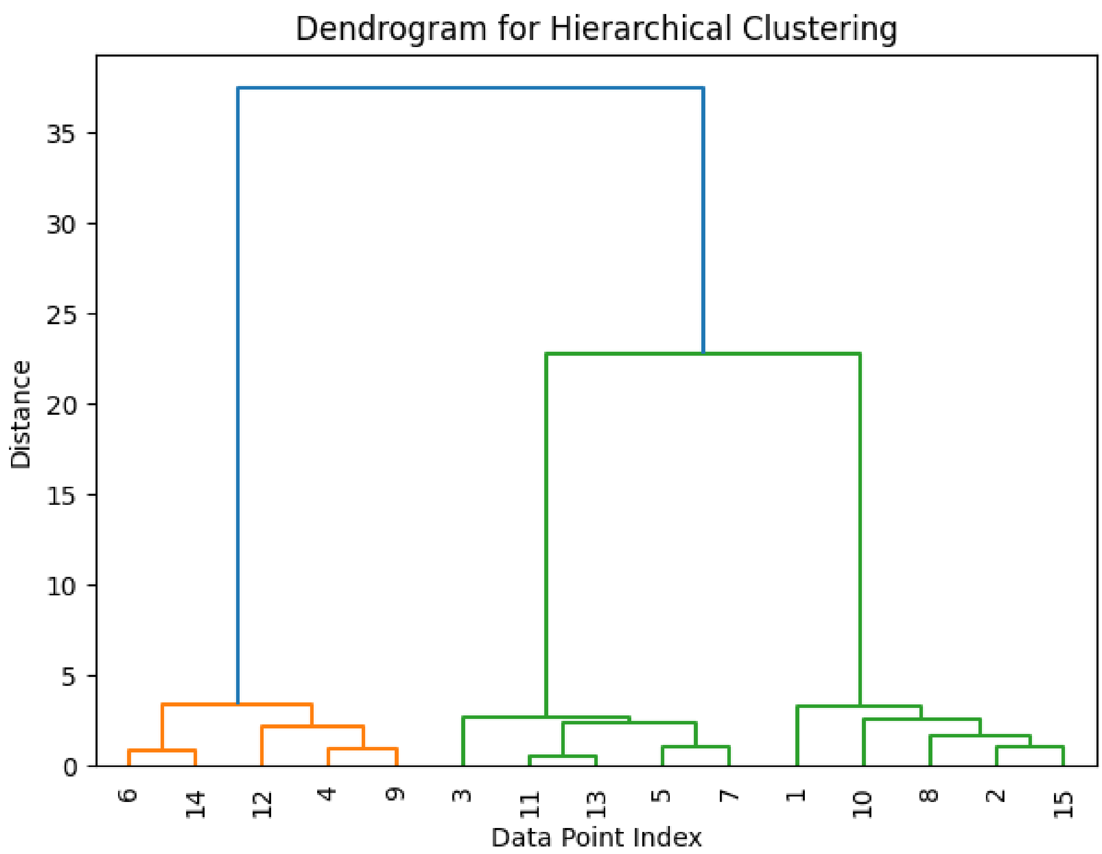
[1]:
```
#   Dendogram -->

import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.datasets import make_blobs

# Generate synthetic data
X, _ = make_blobs(n_samples=15, centers=3, random_state=42)

# Perform hierarchical clustering
linked = linkage(X, method='ward')

# Plot dendrogram with a width of 500px
plt.figure(figsize=(7, 5))  # Width: 5 inches, Height: 3 inches
dendrogram(linked,
           labels=np.arange(1, len(X)+1),
           leaf_rotation=90,
           leaf_font_size=10)
plt.title('Dendrogram for Hierarchical Clustering')
plt.xlabel('Data Point Index')
plt.ylabel('Distance')
plt.show()
```




```
[ ]: '''
         Visual inspection of dendrogram to find optimal clusters -->

         Step 1: Create a dendrogram using hierarchical clustering.
         Step 2: Observe the vertical lines (linkages) that connect clusters.
         Step 3: Identify the longest vertical line (largest distance)
         that you can "cut" without crossing more than one horizontal line.
         Step 4: Draw a horizontal line across the dendrogram where this longest
         vertical line exists. The number of vertical lines (clusters) that the
         horizontal line intersects indicates the number of clusters.
     '''
```

```
[8]: #   Importing Libraries -->

     import pandas as pd
     from sklearn.cluster import AgglomerativeClustering
```

```
[5]: #   Importing Data -->

     data = pd.read_csv('Data/Mall_Customers.csv')
```

```
data.head(10)
```

```
[5]:    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
     5           6  Female   22                  17                      76
     6           7  Female   35                  18                       6
     7           8  Female   23                  18                      94
     8           9    Male   64                  19                       3
     9          10  Female   30                  19                      72
```

```
[6]: x_data = data.iloc[:,[3,4]].values
     x_data
```

```
[6]: 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],
```

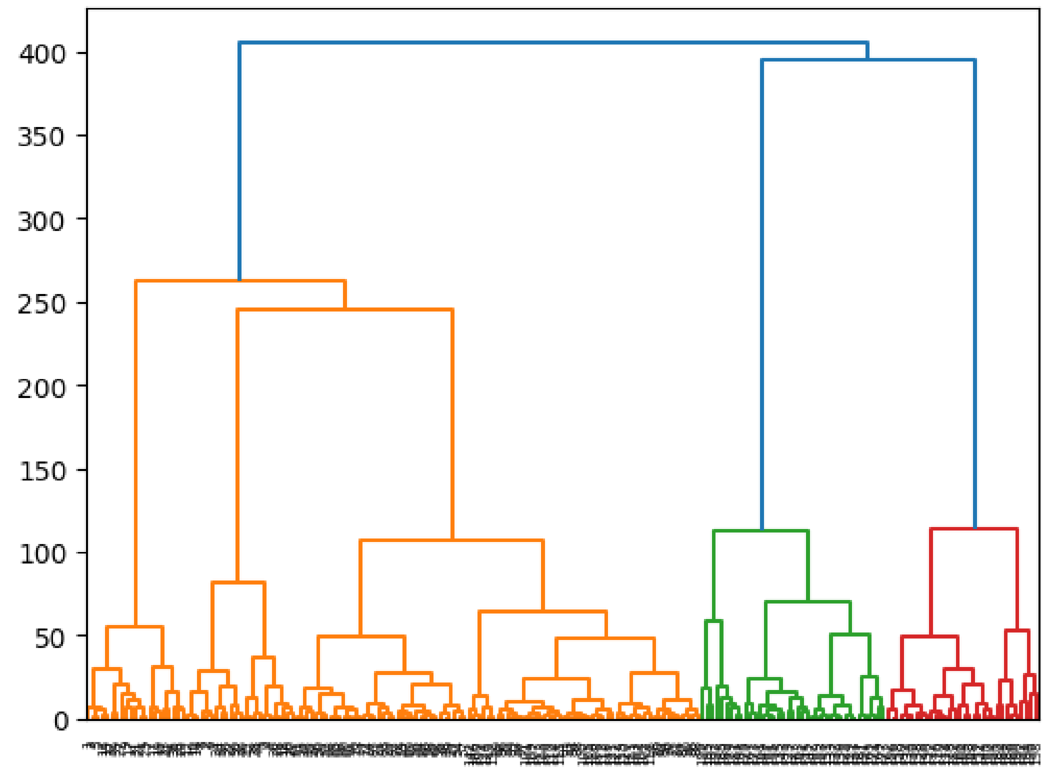
```
[ 70,  29],
[ 70,  77],
[ 71,  35],
[ 71,  95],
[ 71,  11],
[ 71,  75],
[ 71,   9],
[ 71,  75],
[ 72,  34],
[ 72,  71],
[ 73,   5],
[ 73,  88],
[ 73,   7],
[ 73,  73],
[ 74,  10],
[ 74,  72],
[ 75,   5],
[ 75,  93],
[ 76,  40],
[ 76,  87],
[ 77,  12],
[ 77,  97],
[ 77,  36],
[ 77,  74],
[ 78,  22],
[ 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]], dtype=int64)
```

[7]:

```
#   Optimal Clusters by Dendrogram -->

dendro = dendrogram(linkage(x_data, method='ward'))
plt.show()
```



[11]:
```
#   Training the model -->

model = AgglomerativeClustering(n_clusters=5, linkage='ward')
y_pred = model.fit_predict(x_data)
y_pred
```

[11]:
```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 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, 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, 2, 1, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2], dtype=int64)
```

[14]:
```
#   Visualizing The Clusters -->

plt.scatter(x_data[y_pred==0,0], x_data[y_pred==0,1], s=50, c='magenta')
plt.scatter(x_data[y_pred==1,0], x_data[y_pred==1,1], s=50, c='blue')
plt.scatter(x_data[y_pred==2,0], x_data[y_pred==2,1], s=50, c='green')
```

```
plt.scatter(x_data[y_pred==3,0], x_data[y_pred==3,1], s=50, c='red')
plt.scatter(x_data[y_pred==4,0], x_data[y_pred==4,1], s=50, c='cyan')
plt.xlabel('Annual Income ($)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```