DATA SCIENCE WITH PYTHON PROJECT (HIERARCHIAL CLUSTERING)

T. SANJIV

ABSTRACT

In this project, we shall be using the concept of hierarchical clustering. In this case, the client has given us a set of information about the annual income of people visiting a mall and the percentage of business they do with the mall, as in how much percentage of their income they spend on the mall. The client wants us to categorize people into different clusters using hierarchical clustering. The number of clusters can however be ambiguous as the client is also unsure of it. Once the clusters are obtained, with respect to it, the client can start giving different kind of offers in such a way that he can ensure to have repetitive customer inflow. The basic interest of the client, thus is to increase the number of customers visiting the mall and making purchases.

OBJECTIVE

The objective is to utilize the dataset, MallCustomers.csv to obtain the input of Annual Income and Spending Score (1-100) of people visiting the mall and to come up with a suitable dendrogram and agglomerative clustering and finally, to plot and visualize the same.

INTRODUCTION

The dataset has records of 200 people. Different applications include mall customers recommending different products and support, market segmentation, customer segmentation, social network analysis, research-oriented clustering, biological analysis etc. The coding is completely done using Python language. The modules/libraries imported are:

1)Pandas

2)Matplotlib

3)Scipy

4)Sklearn

METHODOLOGY

Hierarchical clustering is a popular method for grouping objects. It creates groups so that objects within a group are similar to each other and different from objects in other groups. Clusters are visually represented in a hierarchical tree called a dendrogram. The dendrogram can be cut at the appropriate level to obtain the desired number of clusters. They make it easy to examine and interpret clusters.

There are two main types of hierarchical clustering:

Agglomerative: Initially, each object is considered to be its own cluster and then, the clusters are merged step by step until a single cluster remains.

Divisive: Initially, all objects are considered in a single cluster. Then the division process is performed step by step until each object forms a different cluster.

Hierarchical clustering employs a measure of distance/similarity to create new clusters.

Min (Single) Linkage: One way to measure the distance between clusters is to find the minimum distance between points in those clusters.

Max (Complete) Linkage: Another way to measure the distance is to find the maximum distance between points in two clusters.

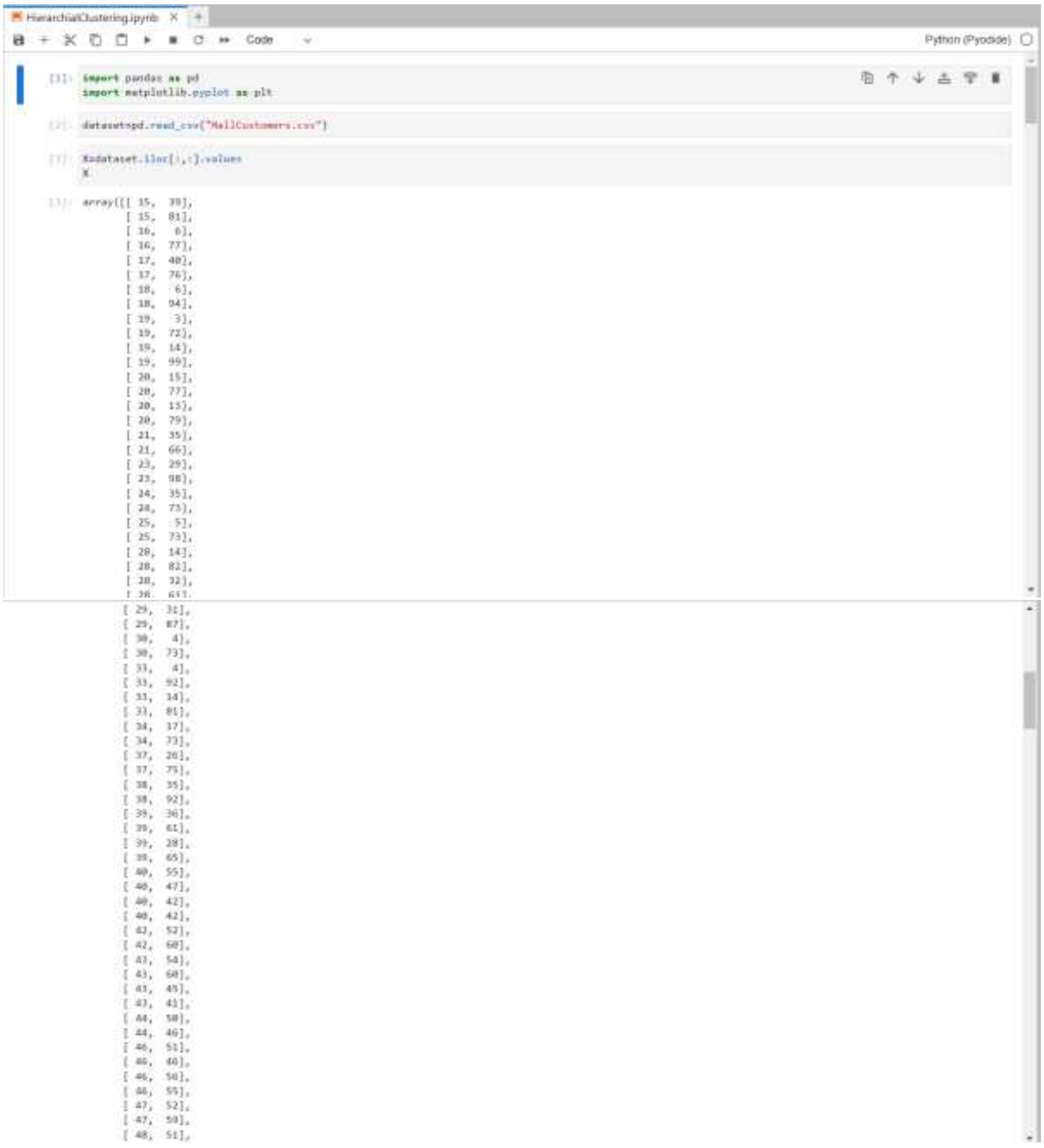
Centroid Linkage: The Centroid method defines the distance between clusters as being the distance between their centers/centroids.

Average Linkage: The Average method defines the distance between clusters as the average pairwise distance among all pairs of points in the clusters.

Ward Linkage: The Ward approach analyzes the variance of the clusters rather than measuring distances directly, minimizing the variance between clusters.

We use matplotlib for plotting. Scipy library has the linkage function for hierarchical (agglomerative) clustering. The linkage function has several methods available for calculating the distance between clusters: single, average, weighted, centroid, median and ward. To draw the dendrogram, we'll use the dendrogram function. Scikit-Learn library has its own function for agglomerative hierarchical clustering: AgglomerativeClustering. Options for calculating the distance between clusters include ward, complete, average, and single. We need to import the AgglomerativeClustering class, then instantiate it with the number of desired clusters and the distance(linkage) function to use.

CODE



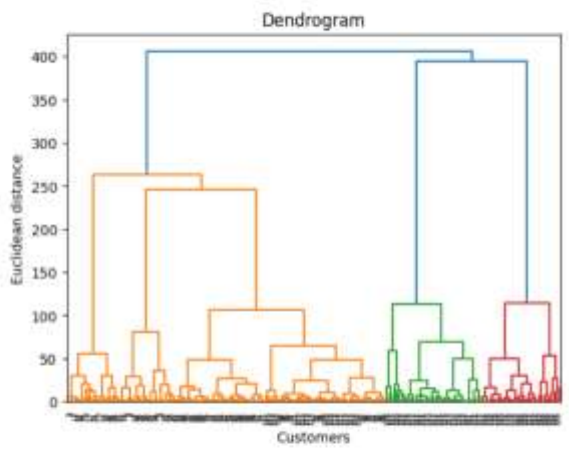
```
[1]: import pandas as pd
     import matplotlib.pyplot as plt

[2]: dataset=pd.read_csv("MallCustomers.csv")

[3]: X=dataset.iloc[:,:].values
     X

[3]: 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],
[ 50,  56],
[ 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,  45],
[ 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,  39],
[ 70,  77],
[ 71,  35],
[ 71,  95],
[ 71,  11],
[ 71,  75],
[ 71,   9],
[ 71,  75],
[ 72,  34],
[ 72,  71],
[ 73,   5],
[ 73,  80],
[ 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)
```

```
[4]: import scipy.cluster.hierarchy as sch
```

```
[5]: dendogram=sch.dendrogram(sch.linkage(X,method='ward'))
     plt.title("Dendrogram")
     plt.xlabel("Customers")
     plt.ylabel("Euclidean distance")
     plt.show()
```

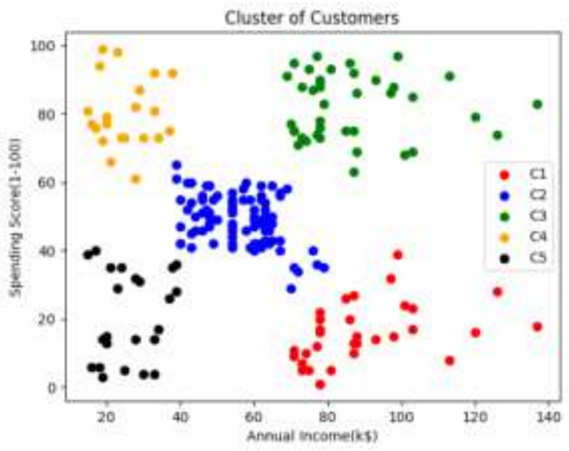


```
[6]: from sklearn.cluster import AgglomerativeClustering
     clustering=AgglomerativeClustering(n_clusters=5)
```

```
[7]: y_hc=clustering.fit_predict(X)
     y_hc
```

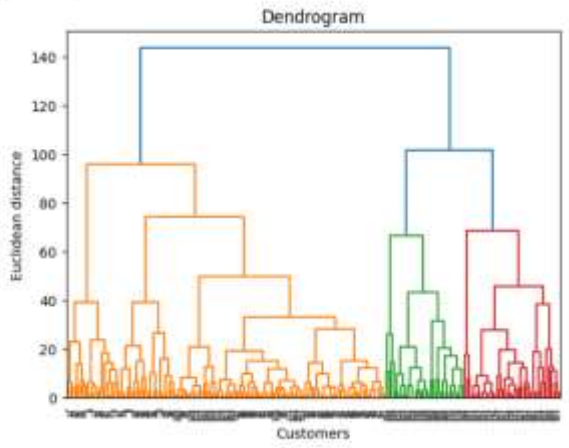
```
[7]: array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
            4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
            4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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=int32)
```

```
[8]: plt.scatter(X[y_hc==0,0], X[y_hc==0,1], c="red", label="C1")
     plt.scatter(X[y_hc==1,0], X[y_hc==1,1], c="blue", label="C2")
     plt.scatter(X[y_hc==2,0], X[y_hc==2,1], c="green", label="C3")
     plt.scatter(X[y_hc==3,0], X[y_hc==3,1], c="orange", label="C4")
     plt.scatter(X[y_hc==4,0], X[y_hc==4,1], c="black", label="C5")
     plt.title("Cluster of Customers")
     plt.xlabel("Annual Income(k$)")
     plt.ylabel("Spending Score(1-100)")
     plt.legend()
     plt.show()
```

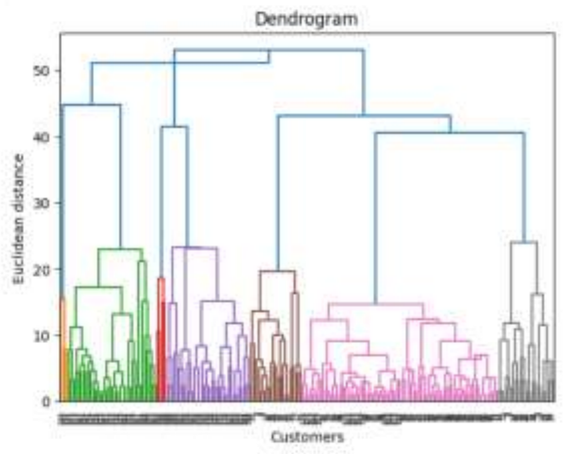


```
[ ]:
```

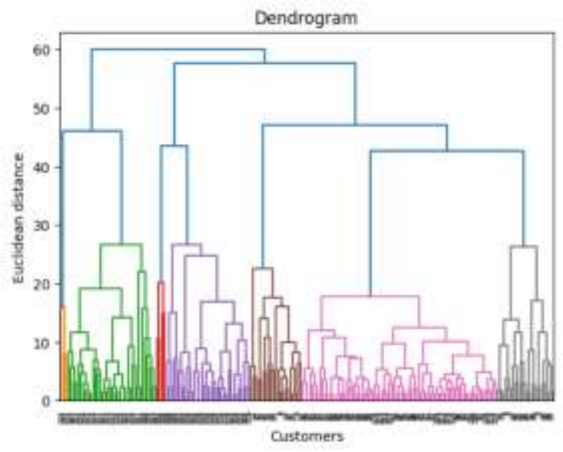
```
[9]: dendogram=sch.dendrogram(sch.linkage(X,method='complete'))
     plt.title("Dendrogram")
     plt.xlabel("Customers")
     plt.ylabel("Euclidean distance")
     plt.show()
```



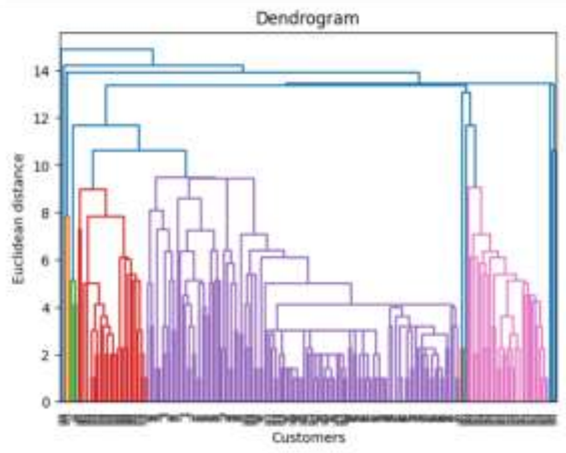
```
[10]: dendrogram=sch.dendrogram(sch.linkage(X,method='centroid'))
      plt.title("Dendrogram")
      plt.xlabel("Customers")
      plt.ylabel("Euclidean distance")
      plt.show()
```



```
[11]: dendrogram=sch.dendrogram(sch.linkage(X,method='average'))
      plt.title("Dendrogram")
      plt.xlabel("Customers")
      plt.ylabel("Euclidean distance")
      plt.show()
```



```
[12]: dendrogram=sch.dendrogram(sch.linkage(X,method='single'))
      plt.title("Dendrogram")
      plt.xlabel("Customers")
      plt.ylabel("Euclidean distance")
      plt.show()
```



```
//Code for implementing
import pandas as pd
import matplotlib.pyplot as plt
dataset=pd.read_csv("MallCustomers.csv")
X=dataset.iloc[:,:].values
import scipy.cluster.hierarchy as sch
dendogram=sch.dendrogram(sch.linkage(X,method='ward'))
plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distance")
plt.show()
from sklearn.cluster import AgglomerativeClustering
clustering=AgglomerativeClustering(n_clusters=5)
y_hc=clustering.fit_predict(X)
plt.scatter(X[y_hc==0,0], X[y_hc==0,1], c="red", label="C1")
plt.scatter(X[y_hc==1,0], X[y_hc==1,1], c="blue", label="C2")
plt.scatter(X[y_hc==2,0], X[y_hc==2,1], c="green", label="C3")
plt.scatter(X[y_hc==3,0], X[y_hc==3,1], c="orange", label="C4")
plt.scatter(X[y_hc==4,0], X[y_hc==4,1], c="black", label="C5")
plt.title("Cluster of Customers")
plt.xlabel("Annual Income(k$)")
plt.ylabel("Spending Score(1-100)")
plt.legend()
plt.show()
// To get other dendograms the linkage name can alone be changed appropriately
```

CONCLUSION

We have classified the data into different clusters successfully. Namely, one with lesser income and lesser spending score, one with higher income and lesser spending score, one with lesser income and higher spending score, one with higher income and higher spending score and lastly one with optimum income and optimum spending score. We have also obtained dendrograms for various linkage types successfully. Thus, this can be used by the client according to his/her needs and decide the further steps forward.