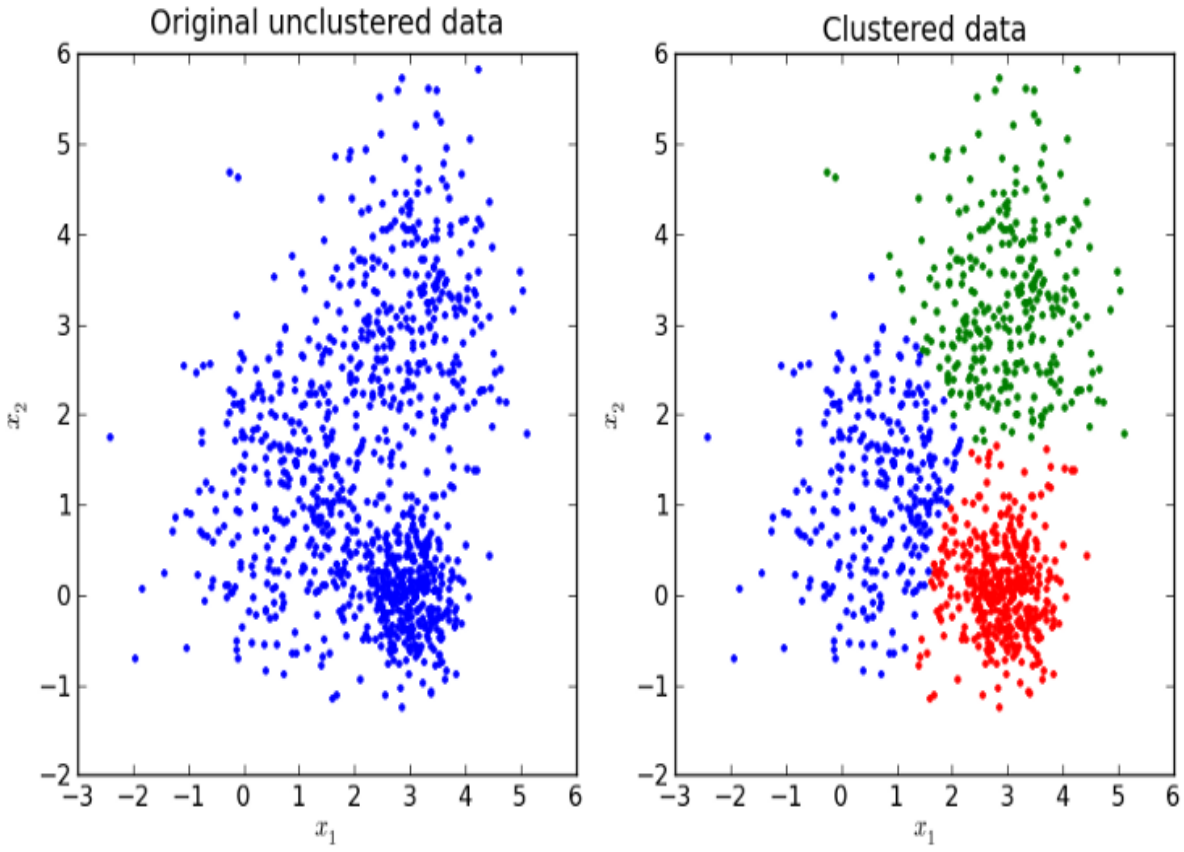
DWM Programming Assignment

Name - Vishal Kriplani (69)
Piyush Chotiya (49)
Ganesh Karwa (39)

Topic - Single Linkage Hierarchical Clustering

What is Clustering??

Clustering is basically a technique that groups similar data points such that the points in the same group are more similar to each other than the points in the other groups. The group of similar data points is called a **Cluster.**



Hierarchical clustering Technique:

Hierarchical clustering is one of the popular and easy to understand clustering technique. This clustering technique is divided into two types:

1. Agglomerative
2. Divisive

Agglomerative Hierarchical clustering Technique: In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

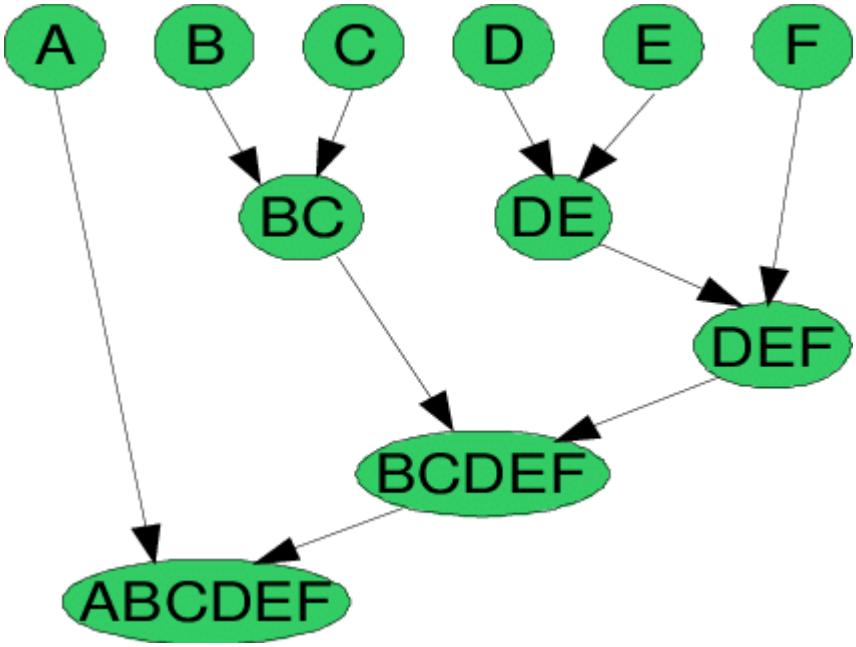
The basic algorithm of Agglomerative is straight forward.

- Compute the proximity matrix
- Let each data point be a cluster
- Repeat: Merge the two closest clusters and update the proximity matrix
- Until only a single cluster remains

Key operation is the computation of the proximity of two clusters

To understand better let's see a pictorial representation of the Agglomerative Hierarchical clustering Technique. Lets say we have six data points {A,B,C,D,E,F}.

- Step- 1: In the initial step, we calculate the proximity of individual points and consider all the six data points as individual clusters as shown in the image below.

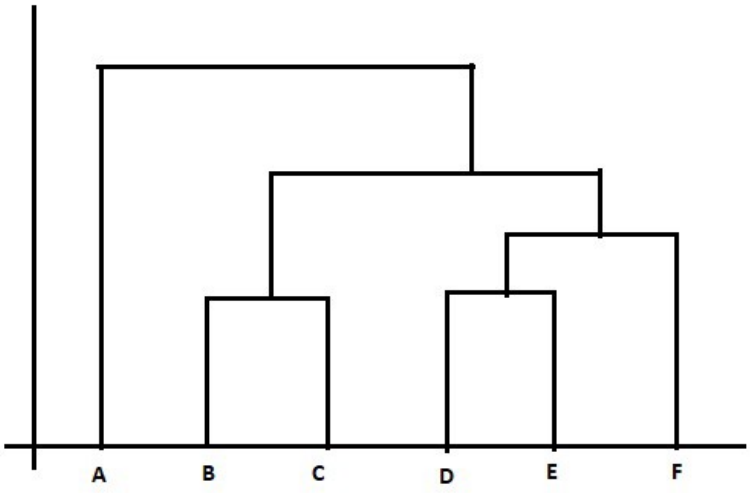


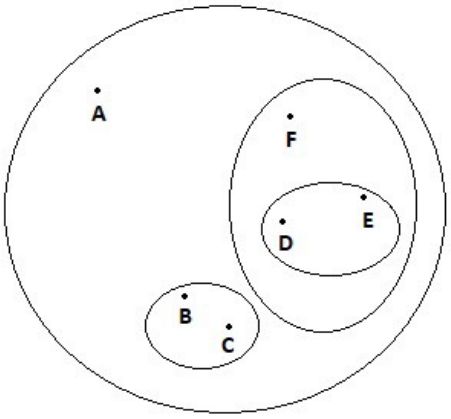
Agglomerative Hierarchical Clustering Technique

- Step- 2: In step two, similar clusters are merged together and formed as a single cluster. Let's consider B,C, and D,E are similar clusters that are merged in step two. Now, we're left with four clusters which are A, BC, DE, F.
- Step- 3: We again calculate the proximity of new clusters and merge the similar clusters to form new clusters A, BC, DEF.
- Step- 4: Calculate the proximity of the new clusters. The clusters DEF and BC are similar and merged together to form a new cluster. We're now left with two clusters A, BCDEF.
- Step- 5: Finally, all the clusters are merged together and form a single cluster.

The Hierarchical clustering Technique can be visualized using a **Dendrogram.**

A **Dendrogram** is a tree-like diagram that records the sequences of merges or splits.





Dendrogram representation

2. Divisive Hierarchical clustering Technique: Since the Divisive Hierarchical clustering Technique is not much used in the real world, I'll give a brief of the Divisive Hierarchical clustering Technique.

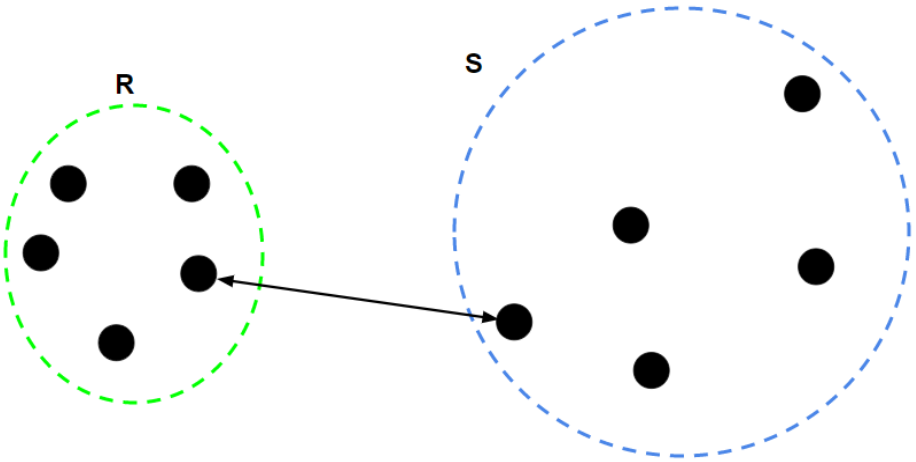
In simple words, we can say that the Divisive Hierarchical clustering is exactly the opposite of the **Agglomerative Hierarchical clustering.** In Divisive Hierarchical clustering, we consider all the data points as a single cluster and in each iteration, we separate the data points from the cluster which

are not similar. Each data point which is separated is considered as an individual cluster. In the end, we'll be left with n clusters.

As we're dividing the single clusters into n clusters, it is named as **Divisive Hierarchical clustering.**

Single Linkage: For two clusters R and S, the single linkage returns the minimum distance between two points i and j such that i belongs to R and j belongs to S.

$$L(R,S) = min(D(i,j)), i \epsilon R, j \epsilon S$$



Problem statement

You are owing a supermarket mall and through membership cards , you have some basic data about your customers like Customer ID, age, gender, annual income and spending score. Problem Statement You own the mall and want to understand the customers like who can be easily converge [Target Customers] so that the sense can be given to marketing team and plan the strategy accordingly.

All Imports

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import random
import seaborn as sns
```

```
import scipy as sp
from sklearn import datasets
from numpy import linalg as LA
import scipy.cluster.hierarchy as shc
from sklearn.metrics.pairwise import pairwise_distances
import sys
```

Reading Raw Mall Customer Data from csv

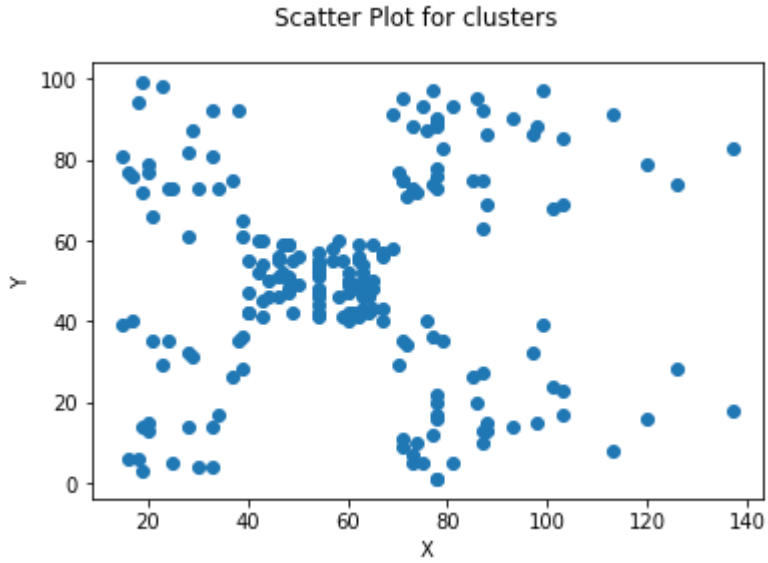
```
import pandas as pd

url = 'https://raw.githubusercontent.com/Vk76/Single-Linkage-Hierarchical-Clust
df = pd.read_csv(url)
data = df.iloc[:, [3,4]].values
```

Plotting all the data set Points

```
fig = plt.figure()
fig.suptitle('Scatter Plot for clusters')
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.scatter(data[:,0],data[:,1])
```

```
<matplotlib.collections.PathCollection at 0x7f4391c0d0f0>
```



Function for plotting the clusters with different color

input -

data : two dimensional data frame

linkage : single

no_of_cluseters : clusters to make

```
def hierarchical_clustering(data,linkage,no_of_clusters):
    #first step is to calculate the initial distance matrix
    #it consists distances from all the point to all the point
    color = ['r','b','g','c','m','y','k','w']

    initial_distances = pairwise_distances(data,metric='euclidean')
    #making all the diagonal elements infinity
    np.fill_diagonal(initial_distances,sys.maxsize)
    clusters = find_clusters(initial_distances,linkage)

    #plotting the clusters
    iteration_number = initial_distances.shape[0] - no_of_clusters
    clusters_to_plot = clusters[iteration_number]
    arr = np.unique(clusters_to_plot)

    indices_to_plot = []
    fig = plt.figure()
    fig.suptitle('Scatter Plot for clusters')
    ax = fig.add_subplot(1,1,1)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    for x in np.nditer(arr):
        indices_to_plot.append(np.where(clusters_to_plot==x))
    p=0

    clusters.pop(5)
    for i in range(0,len(indices_to_plot)):
        for j in np.nditer(indices_to_plot[i]):
               ax.scatter(data[j,0],data[j,1], c= color[p])
        p = p + 1
    plt.title('Clusters of customers using Hierarchical Clustering')
    plt.xlabel('Annual Income (K$)')
    plt.ylabel('Spending Score (1-100)')
    plt.legend()
    plt.show()


```

Function for finding the clusters in sets

input -

data : two dimensional data frame
linkage : single

output -

clusters : two dimensional list of clusters

```
def find_clusters(input,linkage):
    clusters = {}
    row_index = -1
    col_index = -1
    array = []


    for n in range(input.shape[0]):
        array.append(n)

    clusters[0] = array.copy()

    #finding minimum value from the distance matrix
    #note that this loop will always return minimum value from bottom triangle
    for k in range(1, input.shape[0]):
        min_val = sys.maxsize

        for i in range(0, input.shape[0]):
            for j in range(0, input.shape[1]):
                if(input[i][j]<=min_val):
                    min_val = input[i][j]
                    row_index = i
                    col_index = j

        #once we find the minimum value, we need to update the distance matrix
        #updating the matrix by calculating the new distances from the cluster

        #for Single Linkage
        if(linkage == "single" or linkage =="Single"):
            for i in range(0,input.shape[0]):
                if(i != col_index):
                    #we calculate the distance of every data point from newly f
                    temp = min(input[col_index][i],input[row_index][i])
```

```
                    #we update the matrix symmetrically as our distance matrix
                    input[col_index][i] = temp
                    input[i][col_index] = temp

        #set the rows and columns for the cluster with higher index i.e. the ro
        #Set input[row_index][for_all_i] = infinity
        #set input[for_all_i][row_index] = infinity
        for i in range (0,input.shape[0]):
            input[row_index][i] = sys.maxsize
            input[i][row_index] = sys.maxsize

        #Manipulating the dictionary to keep track of cluster formation in each
        #if k=0,then all datapoints are clusters

        minimum = min(row_index,col_index)
        maximum = max(row_index,col_index)
        for n in range(len(array)):
            if(array[n]==maximum):
                array[n] = minimum
        clusters[k] = array.copy()

    return clusters
```

Plotting the graph for 8 different cluster values using single linkage hierarchical clustering

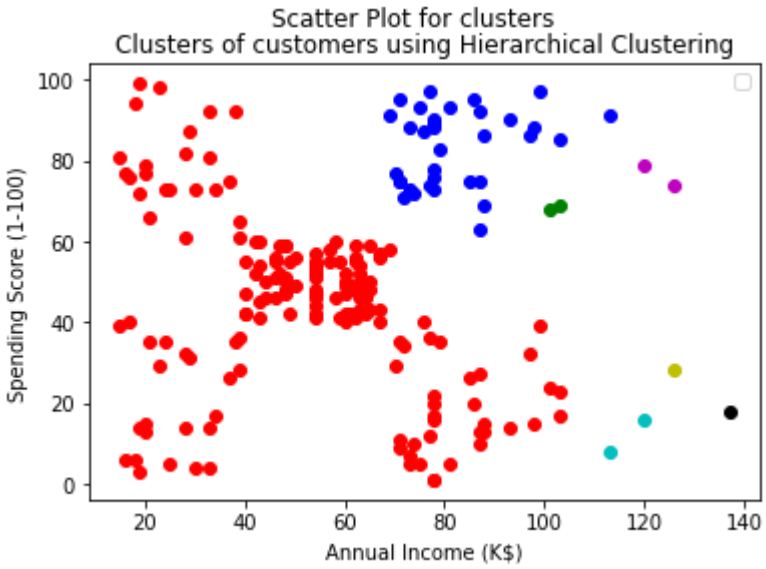
```
# Here we visualize all clusters starting from half of the length of points cor
number_of_clusters = 8

# Here in we just go on decreasing the number of clusters. The points in the sa
while number_of_clusters > 0:
    print(number_of_clusters)
    hierarchical_clustering(data,"single",number_of_clusters)
    number_of_clusters -= 1

```

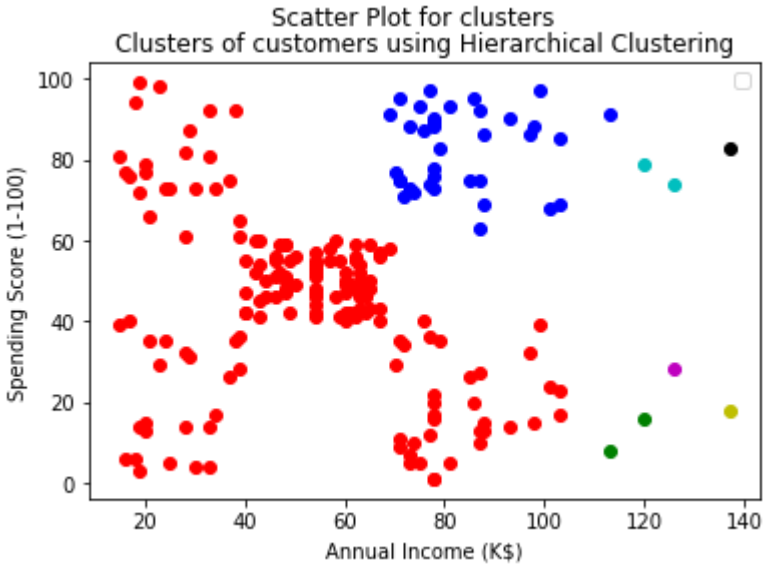
8

```
No handles with labels found to put in legend.
```



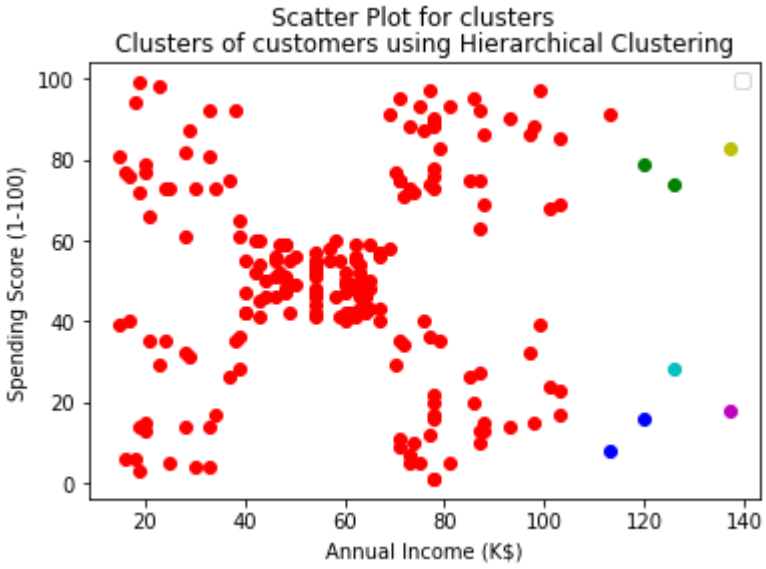
7

```
No handles with labels found to put in legend.
```



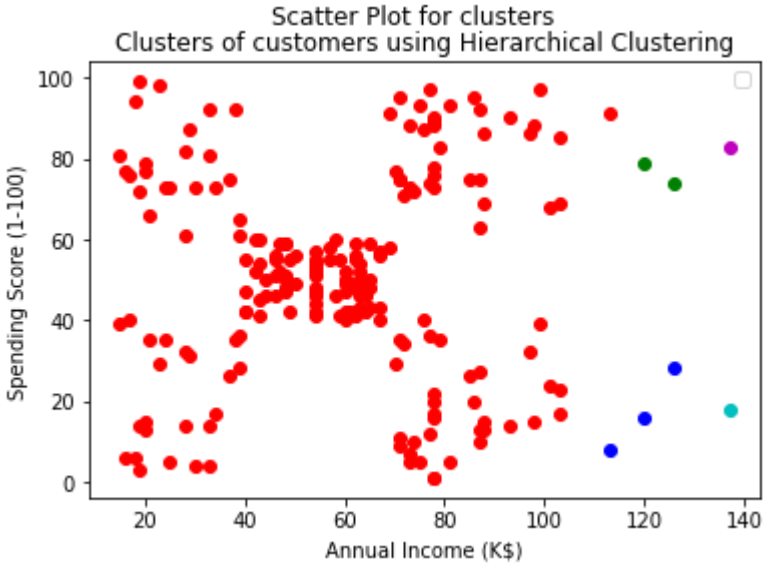
6

```
No handles with labels found to put in legend.
```



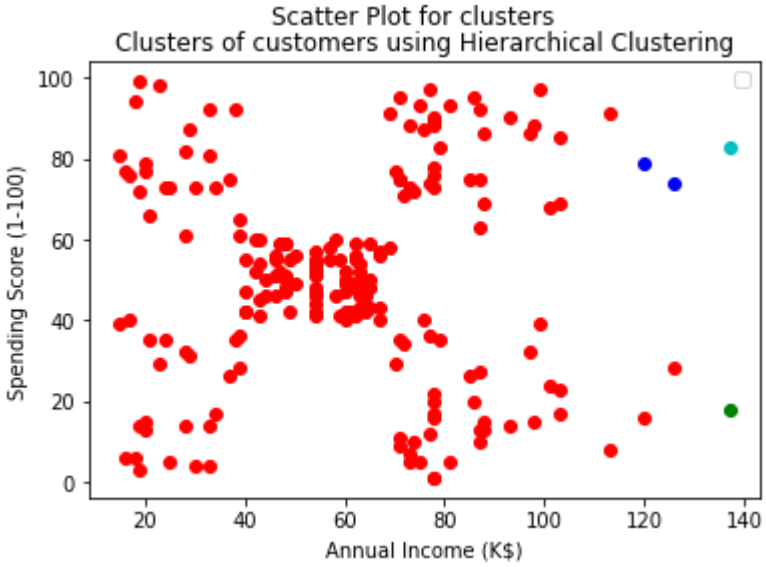
5

No handles with labels found to put in legend.



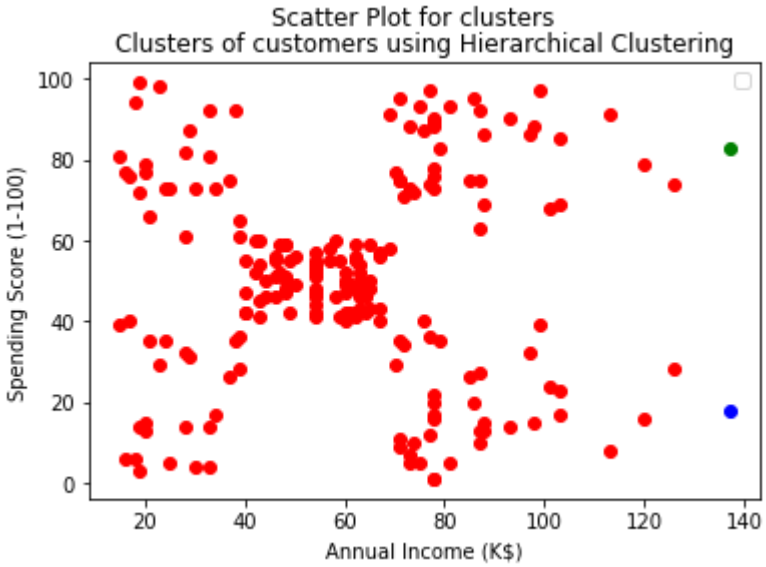
4

No handles with labels found to put in legend.

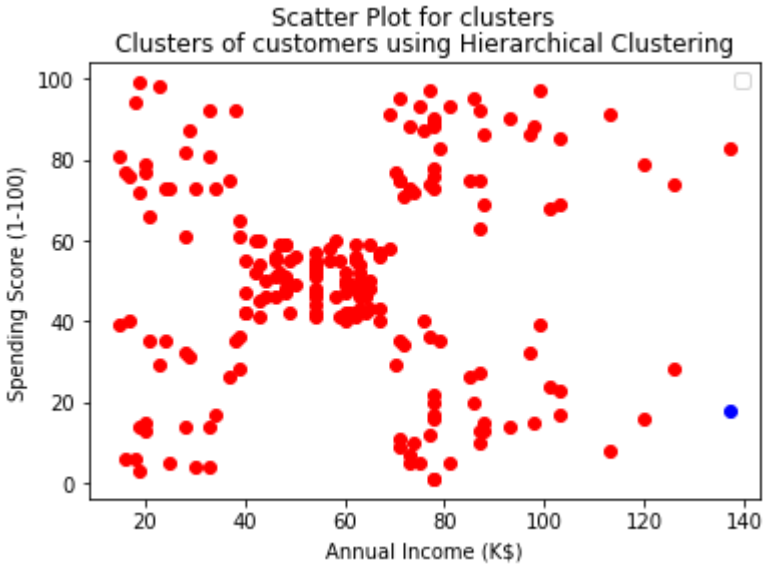


3

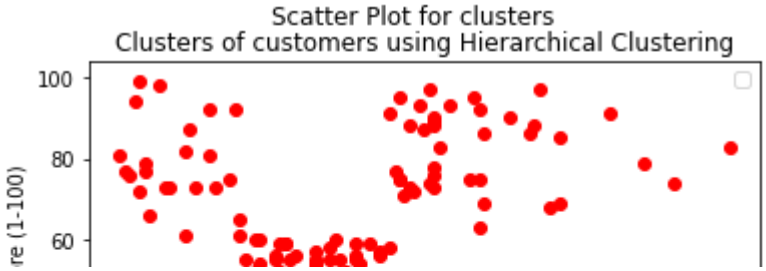
No handles with labels found to put in legend.



```
2
No handles with labels found to put in legend.
```

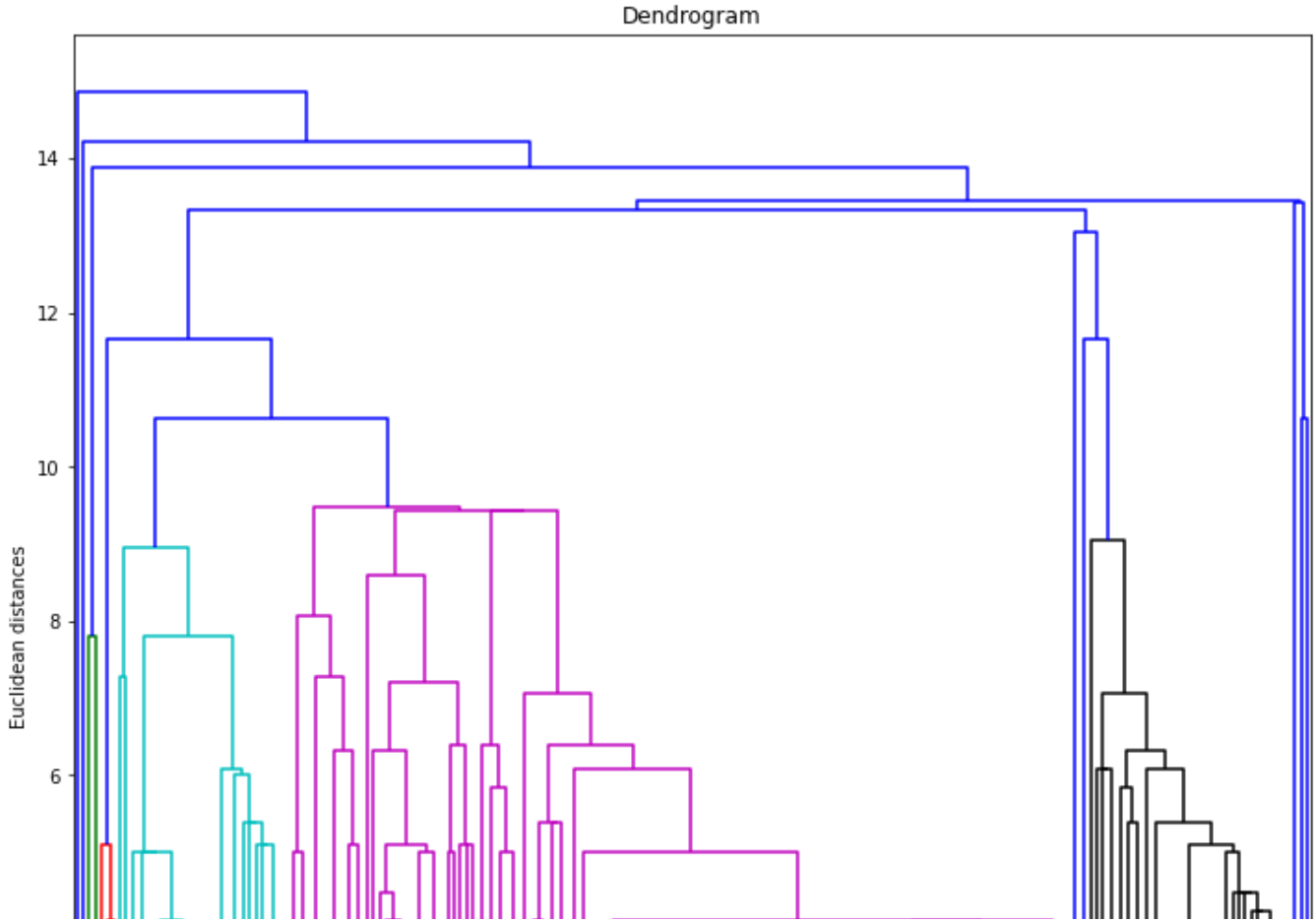



```
1
No handles with labels found to put in legend.
```



Dendogram for the the dataset

```
plt.figure(figsize=(12,12))
dendogram = shc.dendrogram(shc.linkage(data,method = "single"))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



Conclusions

(Results are on basis of single linkage for 8 clusters)

- **Red** color shows unsual behaviour of customers that have low income and high spending and high income and low spending so they are sensible and careless customers.
- **Blue and Green** color has a stable customers and they are **targeted customers** as they have good income and their spending is high.
- **Cyan and yellow** coloured customers are **careful customers** as they have high income but less spending.