Class: BE 3

Batch: P3

Roll no. 41310

Name: Prem Vinod Bansod

Date: 02–09 –2020

Assignment 2

Problem Statement:

Consider a suitable dataset. For clustering of data instances indifferent groups apply different clustering techniques (minimum 2). Visualize the clusters using suitable tool.

Learning Objective:

- To understand different clustering algorithms. Ex. Hierarchical Clustering, K-means Clustering.
- To understand how to visualize data using python libraries.

Software Requirements:

Anaconda, Spyder etc.

Hardware Requirement:

2GB RAM, 500 GB HDD.

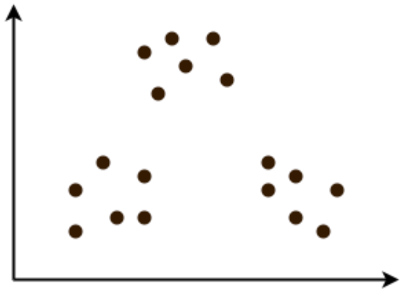
Theory:

K-means Clustering:

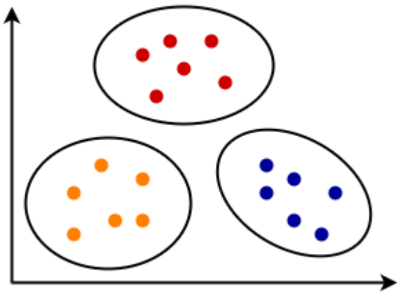
K-Means is probably the most well-known clustering algorithm. It’s taught in a lot of introductory data science and machine learning classes. It’s easy to understand and implement in code! Check out the graphic below for an illustration.

1. To begin, we first select a few classes/groups to use and randomly initialize their respective center points. To figure out the number of classes to use, it’s good to take a quick look at the data and try to identify any distinct groupings. The center points are vectors of the same length as each data point vector and are the “X’s” in the graphic above.
2. Each data point is classified by computing the distance between that point and each group center, and then classifying the point to be in the group whose center is closest to it.
3. Based on these classified points, we recompute the group center by taking the mean of all the vectors in the group.
4. Repeat these steps for a set number of iterations or until the group centers don’t change much between iterations. You can also opt to randomly initialize the group centers a few times, and then select the run that looks like it provided the best results.

K-Means has the advantage that it’s pretty fast, as all we’re really doing is computing the distances between points and group centers; very few computations! It thus has a linear complexity O(n).



Before K-Means



After K-Means

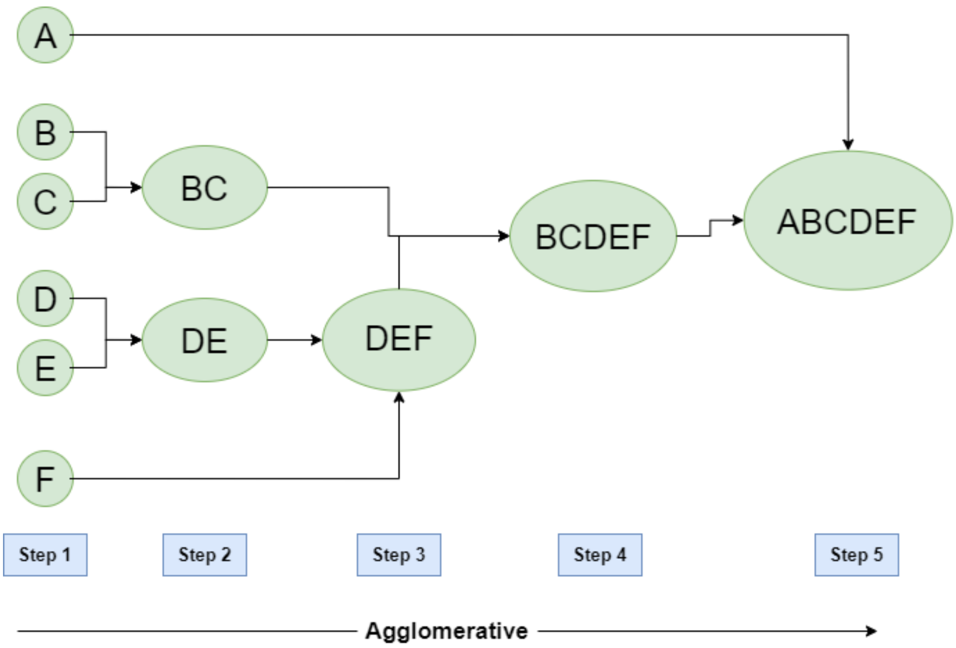
Agglomerative Hierarchical Clustering:

Hierarchical clustering algorithms fall into 2 categories: top-down or bottom-up. Bottom-up algorithms treat each data point as a single cluster at the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all data points. Bottom-up hierarchical clustering is therefore called hierarchical agglomerative clustering or HAC. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.

1. We begin by treating each data point as a single cluster i.e if there are X data points in our dataset then we have X clusters. We then select a distance metric that measures the distance between two clusters. As an example, we will use average linkage which defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster.
2. On each iteration, we combine two clusters into one. The two clusters to be combined are selected as those with the smallest

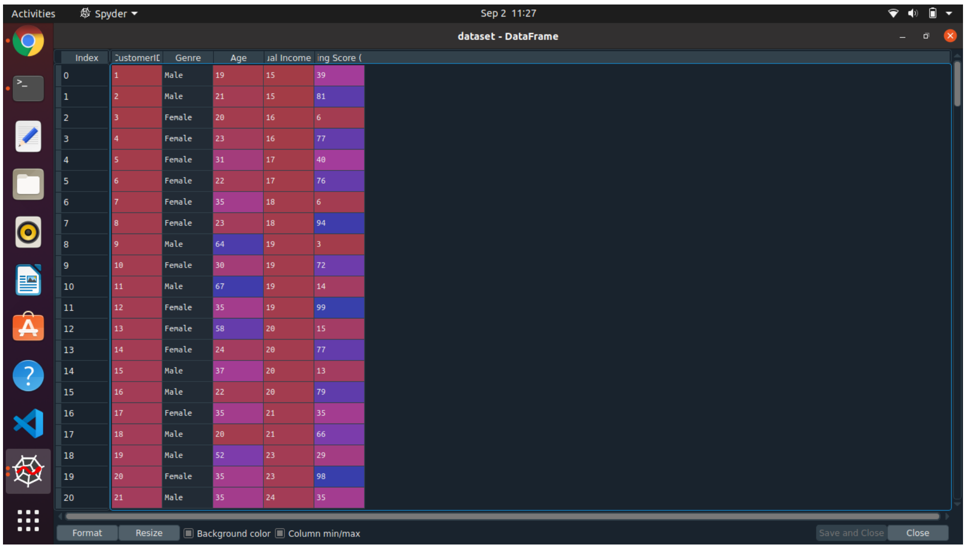
average linkage. I.e. according to our selected distance metric, these two clusters have the smallest distance between each other and therefore are the most similar and should be combined.

3. Step 2 is repeated until we reach the root of the tree i.e., we only have one cluster which contains all data points. In this way we can select how many clusters we want in the end, simply by choosing when to stop combining the clusters i.e. when we stop building the tree!

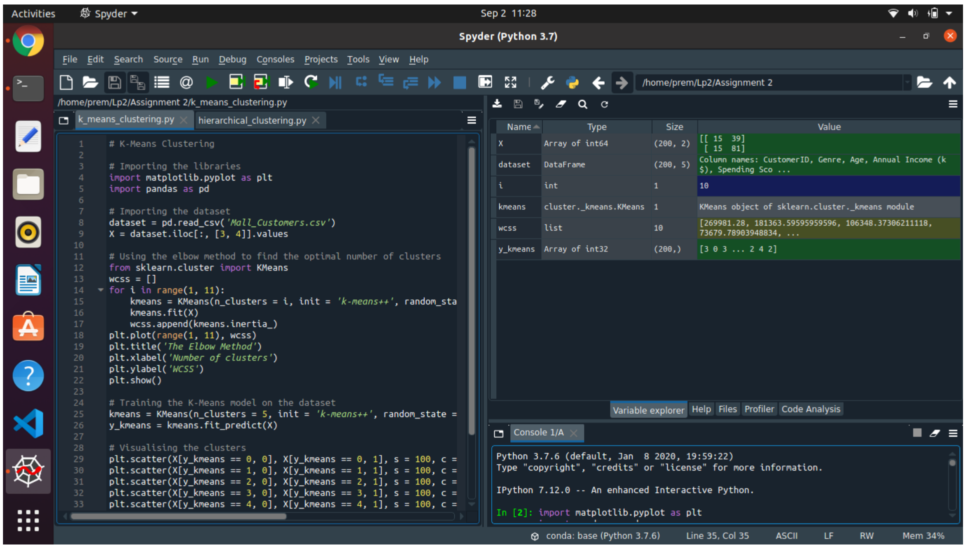


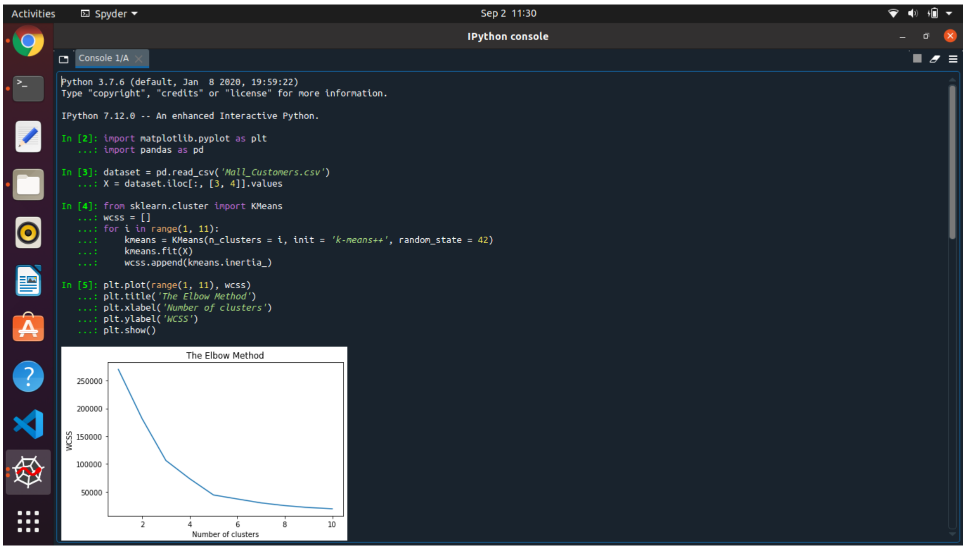
Python Implementation:

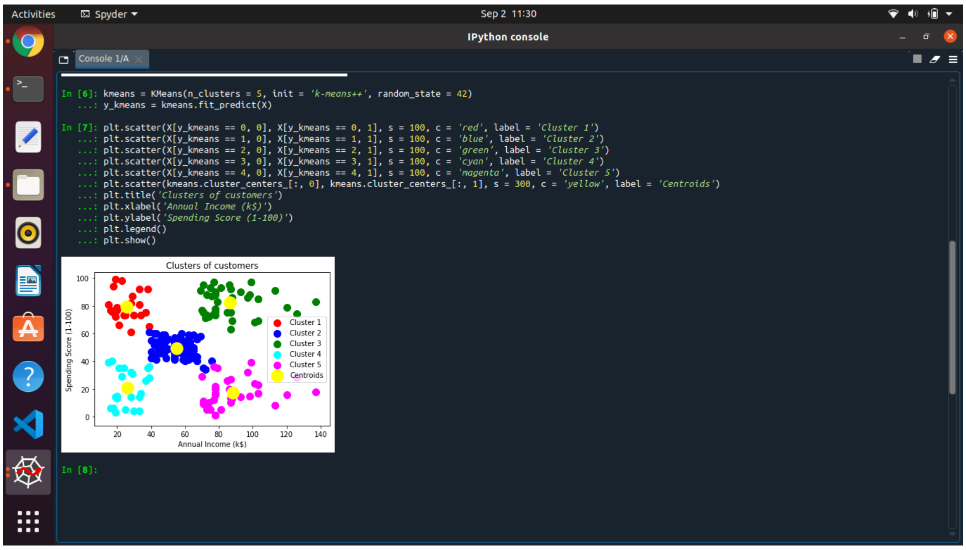
Dataset: Mall_Customer.csv

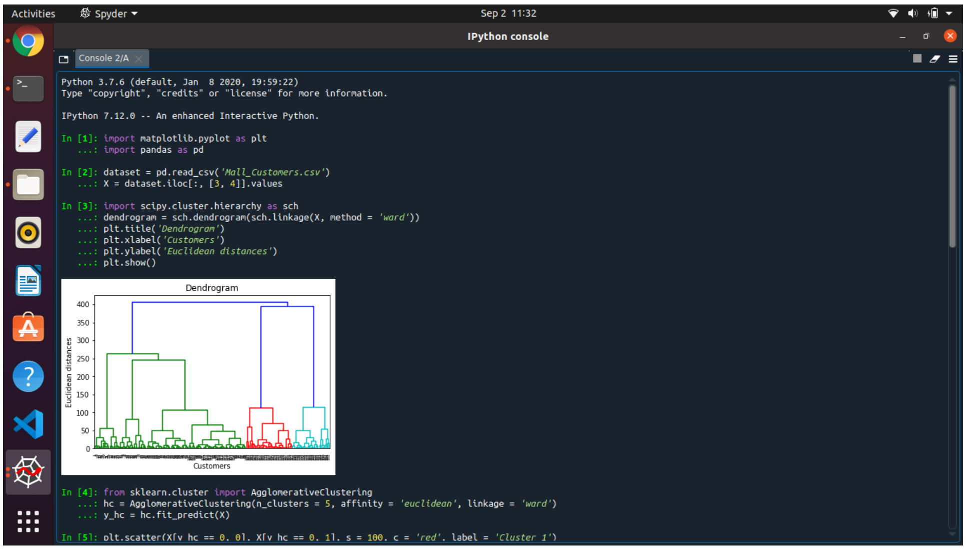


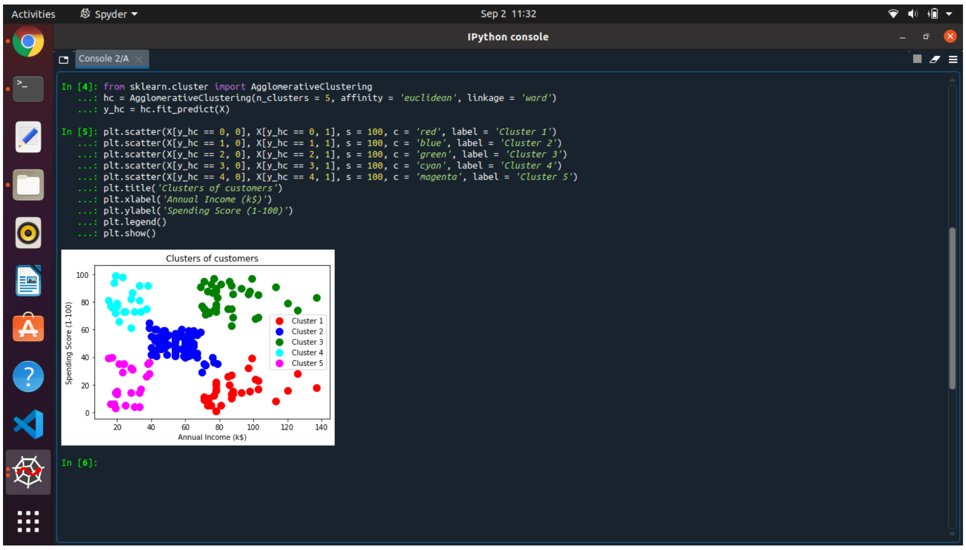
Index	CustomerID	Genre	Age	ual Income	ing Score (
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14
11	12	Female	35	19	99
12	13	Female	58	20	15
13	14	Female	24	20	77
14	15	Male	37	20	13
15	16	Male	22	20	79
16	17	Female	35	21	35
17	18	Male	20	21	66
18	19	Male	52	23	29
19	20	Female	35	23	98
20	21	Male	35	24	35




Activities
Spyder
Sep 2 11:30
IPython console
Console 1/A
Python 3.7.6 (default, Jan 8 2020, 19:59:22)
Type "copyright", "credits" or "license" for more information.
IPython 7.12.0 -- An enhanced Interactive Python.
In [2]: import matplotlib.pyplot as plt
...: import pandas as pd
In [3]: dataset = pd.read_csv('Mall_Customers.csv')
...: X = dataset.iloc[:, [3, 4]].values
In [4]: from sklearn.cluster import KMeans
...: wcss = []
...: for i in range(1, 11):
...: kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
...: kmeans.fit(X)
...: wcss.append(kmeans.inertia_)
In [5]: plt.plot(range(1, 11), wcss)
...: plt.title('The Elbow Method')
...: plt.xlabel('Number of clusters')
...: plt.ylabel('WCSS')
...: plt.show()
The Elbow Method
WCSS
Number of clusters


Activities
Spyder
Sep 2 11:30
IPython console
Console 1/A
In [6]: kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
...: y_kmeans = kmeans.fit_predict(X)
In [7]: plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
...: plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
...: plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
...: plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
...: plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
...: plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
...: plt.title('Clusters of customers')
...: plt.xlabel('Annual Income (k$)')
...: plt.ylabel('Spending Score (1-100)')
...: plt.legend()
...: plt.show()
Clusters of customers
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids
Spending Score (1-100)
Annual Income (k$)
In [8]:


Activities
Spyder
Sep 2 11:32
IPython console
Console 2/A
Python 3.7.6 (default, Jan 8 2020, 19:59:22)
Type "copyright", "credits" or "license" for more information.
IPython 7.12.0 -- An enhanced Interactive Python.
In [1]: import matplotlib.pyplot as plt
...: import pandas as pd
In [2]: dataset = pd.read_csv('Mall_Customers.csv')
...: X = dataset.iloc[:, [3, 4]].values
In [3]: import scipy.cluster.hierarchy as sch
...: dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
...: plt.title('Dendrogram')
...: plt.xlabel('Customers')
...: plt.ylabel('Euclidean distances')
...: plt.show()
Dendrogram
Euclidean distances
Customers
In [4]: from sklearn.cluster import AgglomerativeClustering
...: hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
...: y_hc = hc.fit_predict(X)


Activities
Spyder
Sep 2 11:32
IPython console
Console 2/A
In [4]: from sklearn.cluster import AgglomerativeClustering
...: hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
...: y_hc = hc.fit_predict(X)
In [5]: plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
...: plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
...: plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
...: plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
...: plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
...: plt.title('Clusters of customers')
...: plt.xlabel('Annual Income (k$)')
...: plt.ylabel('Spending Score (1-100)')
...: plt.legend()
...: plt.show()
Clusters of customers
Spending Score (1-100)
Annual Income (k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
In [6]:

Conclusion:

Thus, Clustering techniques like K means and Hierarchical Clustering were used for performing clustering on Mall Customer.csv dataset. Moreover, the clusters were visualize using matplotlib. The assignment was implemented successfully using Spyder.