**Assignment No : 3**

**3.1 Title** : Consider a suitable dataset .For clustering of data instances in different groups, apply different clustering techniques(minimum 2).Visualize the clusters using suitable tool.

**3.2 Prerequisite** : Install Anaconda Python,Jupyter Notebbook,Spyder o Ubuntu 18.04.Add bashrc path.

**3.3 Software Requirement** : Jupyter Notebook,Spyder on Ubuntu.

**3.4 Hardware Requirement** : 2GB RAM500 GB HDD.

**3.5 Objective** : 1) Understanding the dataset

2) Implementing different clustering techniques

3) Visualizing the clusters using suitable tool

**3.6 Outcome :** After completion of this assignment students can develop and implement clustering techniques on different dataset and can find patterns.

**3.7 Theory**:

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Broadly speaking, clustering can be divided into two subgroups :

* Hard Clustering**:** In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
* Soft Clustering: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.

Since the task of clustering is subjective, the means that can be used for achieving this goal are plenty. Every methodology follows a different set of rules for defining the ‘*similarity’* among data points. In fact, there are more than 100 clustering algorithms known. But few of the algorithms are used popularly:

* **Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.
* Centroid models: These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset. These models run iteratively to find the local optima.
* Distribution models**:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.
* Density Models:These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

**3.7.1 K Means Clustering**

K means is an iterative clustering algorithm that aims to find local maxima in each iteration. This algorithm works in these 5 steps :

1. Specify the desired number of clusters K : Let us choose k=2 for these 5 data points in 2-D space.
2. Randomly assign each data point to a cluster :
3. Compute cluster centroids : The centroid of data points in the red cluster is shown using red cross and those in grey cluster using grey cross.
4. Re-assign each point to the closest cluster centroid : Note that only the data point at the bottom is assigned to the red cluster even though its closer to the centroid of grey cluster. Thus, we assign that data point into grey cluster
5. Re-compute cluster centroids : Now, re-computing the centroids for both the clusters.
6. Repeat steps 4 and 5 until no improvements are possible : Similarly, we’ll repeat the 4th and 5th steps until we’ll reach global optima. When there will be no further switching of data points between two clusters for two successive repeats. It will mark the termination of the algorithm if not explicitly mentione

**3.7.2 Hierarchical Clustering**

Hierarchical clustering, as the name suggests is an algorithm that builds hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.

Two important things that you should know about hierarchical clustering are:

* This algorithm has been implemented above using bottom up approach. It is also possible to follow top-down approach starting with all data points assigned in the same cluster and recursively performing splits till each data point is assigned a separate cluster.
* The decision of merging two clusters is taken on the basis of closeness of these clusters. There are multiple metrics for deciding the closeness of two clusters :
  + Euclidean distance: ||a-b||2 = √(Σ(ai-bi))
  + Squared Euclidean distance: ||a-b||22 = Σ((ai-bi)2)
  + Manhattan distance: ||a-b||1 = Σ|ai-bi|
  + Maximum distance:||a-b||INFINITY = maxi|ai-bi|
  + Mahalanobis distance: √((a-b)T S-1 (-b))   {where, s : covariance matrix}

**3.7.3 Applications of Clustering**

Clustering has a large no. of applications spread across various domains. Some of the most popular applications of clustering are:

* Recommendation engines
* Market segmentation
* Social network analysis
* Search result grouping
* Medical imaging
* Image segmentation
* Anomaly detection

**3.8 Code Explanation:**

*In [1]: #Consider a suitable dataset. For clustering of data instances in different groups, ap #clustering techniques (minimum 2)(K-Means and Hierarchichal clustering). Visualize th*

***# Importing the libraries***

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

*In [6]:*

*# Importing the dataset*

dataset = pd.read\_csv('/home/bvcoew/Desktop/Mall\_Customers.csv') X = dataset.iloc[:, [3, 4]].values

Challenges in K-Means Algorithm: 1)Random Initialization of Centroid (solution: K-Means++Algorithm) 2)Decision of number of clusters

*In [3]:*

*# Using the elbow method to find the optimal number of clusters****from sklearn.cluster import****KMeans*

wcss = []

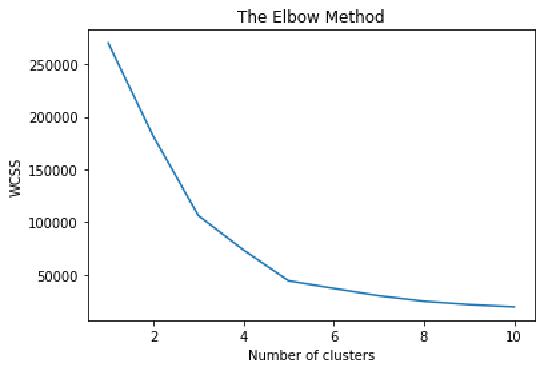
**for**i **in**range(1, 11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42) kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss) plt.title('The Elbow Method') plt.xlabel('Number of clusters') plt.ylabel('WCSS')

plt.show()



*In [4]:*

*# Fitting K-Means to the dataset*

kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42) y\_kmeans = kmeans.fit\_predict(X)

*In [5]:*

*# Visualising the clusters*

plt.scatter(X[y\_kmeans ==0, 0],X[y\_kmeans ==0, 1],s =100,c = 'red',label = 'Clu

plt.scatter(X[y\_kmeans ==1, 0],X[y\_kmeans ==1, 1],s =100,c = 'blue',label = 'Cl

 plt.scatter(X[y\_kmeans ==2, 0],X[y\_kmeans ==2, 1],s =100,c = 'green',label = 'C

plt.scatter(X[y\_kmeans ==3, 0],X[y\_kmeans ==3, 1],s =100,c = 'cyan',label = 'Cl

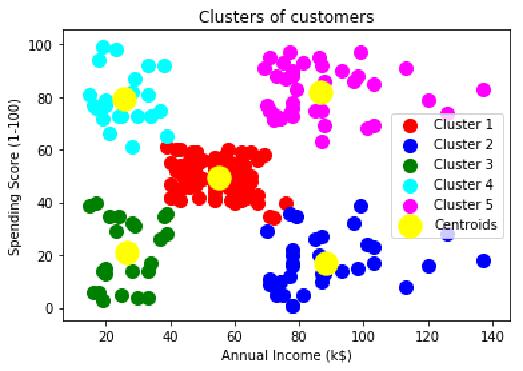
plt.scatter(X[y\_kmeans ==4, 0],X[y\_kmeans ==4, 1],s =100,c = 'magenta',label =

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')plt.legend()

plt.show()



Hierarchical clustering: Agglomerative and Divisive. In Agglomerative clustering, data points are clustered using a bottom-up approach starting with individual data points. Divisive cluster- ing follows top-down approach where all the data points are treated as one big cluster and the clustering process involves dividing the one big cluster into several small clusters. Following is the implementation of Agglomerative Hierarchical clustering.

Hierarchichal clustering based on 1) ﬁnd a pair of closest data points. 2) draw a “bar graph” (height of bar equal to ED between 2 points) euclidian distance on the Y axis and datapoints on the x axis. 3) draw "bar graph for all close pairs. This is layer 1 horizontally. 4)Second layer will connect next close point to each bar. 5) Process repeats till all points over.

*In [7]:*

*# Using the dendrogram to find the optimal number of clusters*

**import scipy.cluster.hierarchy as sch**

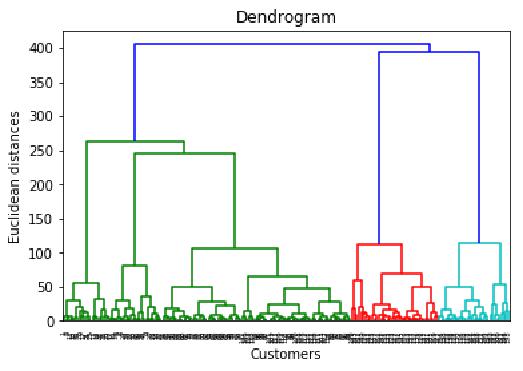
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.show()



*In [8]:*

*# Fitting Hierarchical Clustering to the dataset****from sklearn.cluster import****AgglomerativeClustering*

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward') y\_hc = hc.fit\_predict(X)

*In [9]:*

*# Visualising the clusters*

plt.scatter(X[y\_hc ==0, 0],X[y\_hc ==0, 1],s =100,c = 'red',label = 'Cluster1' plt.scatter(X[y\_hc ==1, 0], X[y\_hc ==1, 1], s =100, c = 'blue', label = 'Cluster2'

 plt.scatter(X[y\_hc ==2, 0],X[y\_hc ==2, 1],s =100, c = 'green',label = 'Cluster 3

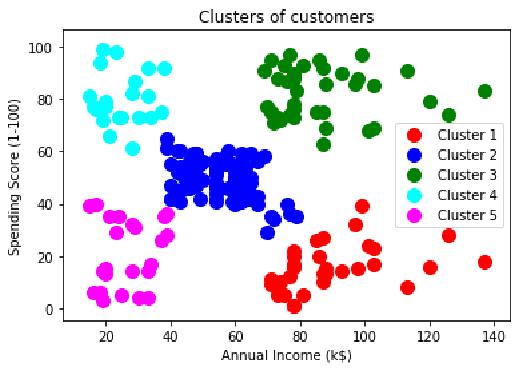
 plt.scatter(X[y\_hc ==3, 0],X[y\_hc ==3, 1],s =100,c = 'cyan', label = 'Cluster 4'

 plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')plt.legend()

plt.show()



**3.9 Conclusion:**

Thus after completing this assignment we are able to understand and implement clustering techniques.