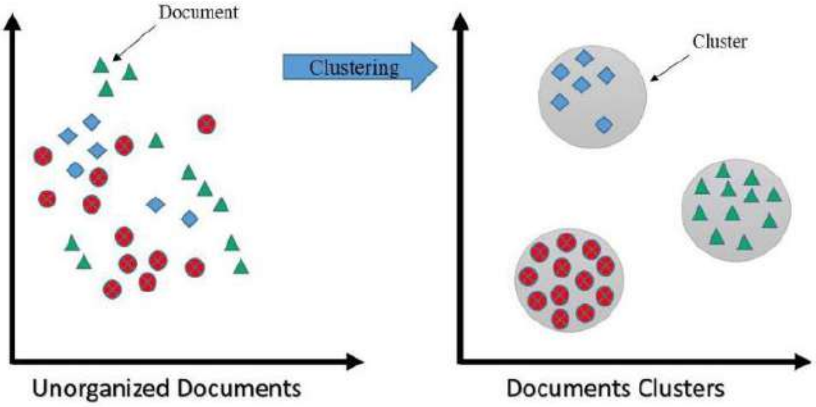
Unit-7
Cluster Analysis

Introduction

Cluster analysis is the process of finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters. A ***cluster*** is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters. While doing cluster analysis, we first partition the set of data into groups/clusters based on data similarity and then assign the labels to the groups.



Clustering analysis is broadly used in many applications such as market research, pattern recognition, data analysis, and image processing. Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases. Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce.

Types of Data in Cluster Analysis

- ***Nominal or Categorical Attributes:*** A nominal attribute is an attribute that can take more than two categories, states, or 'name of things' that do not have a natural order or ranking. E.g. Religion (Hindu, Christian, Muslim, Buddhist. etc.), Hari_color (black, brown, grey, red etc.)
- ***Binary Attributes:*** A binary attribute is a nominal attribute with only two states. Binary variable further can be of two types:
  - ***Symmetric Binary:*** Both outcomes equally important. E.g. gender (Male & Female)
  - ***Asymmetric Binary:*** Outcomes not equally important. E.g. Medical test (positive vs. negative).
- ***Ordinal Attributes:*** Ordinal attribute has two or more values but these values have a meaningful order (ranking), and magnitude between successive value is not known. E.g. Size (small, medium, large), military rank (soldier, sergeant, lutenant, captain, etc.), grades etc.
- ***Interval-scaled Attributes:*** Interval attribute is one where there is order and the difference between the values are equally spaced. E.g. temperature in Fahrenheit or Celsius, calendar dates, etc. An interval attribute doesn't have a true zero-point.

- ***Ratio-scaled Attributes:*** A ratio attribute, has all the properties of an interval attribute, and also has an inherent zero-point just means "no value". E.g. weight, length, temperature in Kelvin (0.0 Kelvin really does mean "no heat"), survival time.
- ***Attributes of Mixed Type:*** A database may contain all the six types of attributes symmetric binary, asymmetric binary, nominal, ordinal, interval, and ratio. And those combinedly called as mixed-type attributes.

Similarity and Dissimilarity Between Objects

The ***similarity*** measure is a way of measuring how data samples are related or closed to each other. On the other hand, the ***dissimilarity*** measure is to tell how much the data objects are distinct.

The similarity measure is usually expressed as a numerical value: It gets higher when the data samples are more alike. It is often expressed as a number between zero and one by conversion: zero means low similarity (the data objects are *dissimilar*). One means high similarity (the data objects are very *similar*).

Similarity might be used to identify

- duplicate data that may have differences due to typos.
- equivalent instances from different data sets. E.g. names and/or addresses that are the same but have misspellings.
- groups of data that are very close (clusters)

Dissimilarity might be used to identify

- outliers
- interesting exceptions, e.g. credit card fraud
- boundaries to clusters

Distance measures are used in order to find similarity or dissimilarity between data objects. The most popular distance measure is ***Euclidean distance***, which is defined as:

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where, $x = (x_1, y_1\ ) \& \ y = (x_2, y_2\ )$

Another well-known metric is ***Manhattan distance***, defined as

$$d(x, y) = |x_2 - x_1| + |y_2 - y_1|$$

Minkowski distance is a generalization of both Euclidean distance and Manhattan distance. It is defined as

$$d(x, y) = \sqrt[p]{|x_2 - x_1|^p + |y_2 - y_1|^p}$$

Where, p is a positive integer can range from 1 to ∞. When $p = 1$ Minkowski distance is equivalent to Manhattan distance & when $p = 2$ Minkowski distance is equivalent to Euclidean distance.

Categories of Clustering Algorithm

In general, the major clustering methods can be classified into the following categories.

- ***Partitioning Methods:*** Given a database of *n* objects or data tuples, a partitioning method constructs *k* partitions of the data, where each partition represents a cluster and *k* <*n*. Given *k*, the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. Typical example of algorithms based on portioning approach are ***k-means***, ***k-means++***, ***mini-batch k means***, ***k-medoids*** etc.

- ***Hierarchical Methods:*** A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either *agglomerative* or *divisive*.
  - The ***agglomerative approach***, also called the *bottom-up* approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition holds.
  - The ***divisive approach***, also called the *top-down* approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

- ***Density-based Methods:*** Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes. This method is based on the notion of *density*. The basic idea is to continue growing the given cluster as long as the density (number of objects or data points) in the neighborhood exceeds some threshold, i.e., for each data point within a given cluster, the radius of a given cluster has to contain at least a minimum number of points. The algorithm or technique comes under density based clustering is ***DBSCAN*** (Density based spatial clustering of applications with noise).

- ***Model-based Methods:*** In this method, a model is hypothesized for each cluster to find the best fit of data for a given model. This method locates the clusters by clustering the density function. It reflects spatial distribution of the data points. This method also provides a way to automatically determine the number of clusters based on standard statistics, taking outlier or noise into account. It therefore yields robust clustering methods.

K-Means Algorithm

K-means Clustering is a type of unsupervised learning which is used when we have unlabeled data (data without define categories or group). It aims to partition *n observations* into *k clusters.*

To perform K-means clustering, we must first specify the desired number of clusters K; then the K-means algorithm will assign each observation to exactly one of the K clusters. Each cluster in the k-means clustering algorithm is represented by a centroid point. Centroid point is the average of all the points in the set.

The idea of the K-Means algorithm is to find k-centroid points and every point in the dataset will belong either of k-sets having minimum Euclidean distance.

Algorithm

1. Specify number of clusters *K*.
2. Randomly select *'K'* points as the initial cluster center.
3. Calculate the distance between each data point and cluster centers.
4. Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
5. Recalculate the new cluster centers by taking the average of the all data points that belong to each cluster.
6. Recalculate the distance between each data point and new obtained cluster centers.
7. Repeat steps 4, 5 and 6 until the same points are assigned to each cluster in consecutive rounds.

Example

Q. Cluster the following eight points (with(x, y) representing locations) into three clusters using K-Means algorithm: (2, 10), (2, 5), (8, 4), (5, 8), (7, 5), (6, 4), (1, 2) and (4, 9).

Solution:

No. of clusters $K = 3$
Let initial cluster centers are: $C1 = (2, 10), C2 = (5, 8)$ and $C3 = (1, 2)$. Suppose considering the *Euclidean distance* as the distance measure: $D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Iteration 1:

Calculating distance between cluster centers and each data points:

	Distance to			Point Belongs
Point	$\mathbf{C1(2,10)}$	$\mathbf{C2(5,8)}$	$\mathbf{C3(1,2)}$	**to Cluster**
(2, 10)	0	3.60	8.06	C1
(2, 5)	5	4.24	3.16	C3
(8, 4)	8.48	5	7.28	C2
(5, 8)	3.61	0	7.21	C2
(7, 5)	7.07	3.60	6.71	C2
(6, 4)	7.21	4.12	5.38	C2
(1, 2)	8.06	7.21	0	C3
(4, 9)	2.24	1.41	7.61	C2

Thus after first iteration:

$$Clusters\ 1 = \{(2, 10)\},$$

$$Cluster\ 2 = \{(8, 4), (5, 8), (7, 5), (6, 4), (4, 9)\},$$

$$Cluster\ 3 = \{(2, 5), (1, 2)\}$$

Now, new cluster centers are:

$$C1 = (2, 10)$$

$$C2 = ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5) = (6, 6)$$

$$C3 = ((2 + 1)/2, (5 + 2)/2) = (1.5, 3.5)$$

Iteration 2:

Calculating distance between new cluster centers and each data points:

	Distance to			Point Belongs to Cluster
Point	$C1(2,10)$	$C2(6,6)$	$C3(1.5,3.5)$	
(2, 10)	0	5.66	6.52	C1
(2, 5)	5	4.12	1.58	C3
(8, 4)	8.48	2.83	6.52	C2
(5, 8)	3.61	2.24	5.70	C2
(7, 5)	7.07	1.41	5.70	C2
(6, 4)	7.21	2	4.53	C2
(1, 2)	8.06	6.40	1.58	C3
(4, 9)	2.24	3.60	6.04	C1

Thus after second iteration:

$Clusters\ 1 = \{(2,10),(4,9)\},$
$Cluster\ 2 = \{(8,4),(5,8),(7,5),(6,4)\},$
$Cluster\ 3 = \{(2,5),(1,2)\}$

Now, new cluster centers are:

$C1 = ((2+4)/2,(10+9)/2) = (3,9.5)$
$C2 = ((8+5+7+6)/4,(4+8+5+4)/4) = (6.5,5.25)$
$C3 = ((2+1)/2,(5+2)/2) = (1.5,3.5)$

Iteration 3:

Calculating distance between new cluster centers and each data points:

	Distance to			Point Belongs to Cluster
Point	$C1(3,9.5)$	$C2(6.5,5.25)$	$C3(1.5,3.5)$	
(2, 10)	1.12	6.54	6.52	C1
(2, 5)	4.61	4.51	1.58	C3
(8, 4)	7.43	1.95	6.52	C2
(5, 8)	2.5	3.13	5.70	C1
(7, 5)	6.02	0.56	5.70	C2
(6, 4)	6.26	1.32	4.53	C2
(1, 2)	7.76	6.39	1.58	C3
(4, 9)	1.12	4.51	6.04	C1

Thus after third iteration:

$Clusters\ 1 = \{(2,10),(5,8),(4,9)\},$
$Cluster\ 2 = \{(8,4),(7,5),(6,4)\},$
$Cluster\ 3 = \{(2,5),(1,2)\}$

Now, new cluster centers are:

$C1 = ((2+5+4)/3,(10+8+9)/3) = (3.67,9)$
$C2 = ((8+7+6)/3,(4+5+4)/3) = (7,4.3)$
$C3 = ((2+1)/2,(5+2)/2) = (1.5,3.5)$

Iteration 4:

Calculating distance between new cluster centers and each data points:

	Distance to			Point Belongs
Point	$C1(3.67,9)$	$C2(7,4.3)$	$C3(1.5,3.5)$	**to Cluster**
(2, 10)	1.95	7.58	6.52	C1
(2, 5)	4.33	5.05	1.58	C3
(8, 4)	6.61	1.04	6.52	C2
(5, 8)	1.66	4.20	5.70	C1
(7, 5)	5.20	0.7	5.70	C2
(6, 4)	5.52	1.04	4.53	C2
(1, 2)	7.49	6.42	1.58	C3
(4, 9)	0.33	5.57	6.04	C1

Thus after fourth iteration:

$$Clusters\ 1 = \{(2,10),(5,8),(4,9)\},$$
$$Cluster\ 2 = \{(8,4),(7,5),(6,4)\},$$
$$Cluster\ 3 = \{(2,5),(1,2)\}$$

The cluster of the data points obtained in fourth iteration is same as third iteration. It means that none of the data points has moved to other cluster. So this becomes the stopping condition for our algorithm.

<u>Advantages of K-Means</u>

- Fast, robust and easier to understand.
- Gives best result when data set are distinct or well separated from each other.

<u>Disadvantages of K-Means</u>

- Require number of clusters in advance.
- Sensitive to noise and outlier data points.
- Randomly choosing of the cluster centers cannot lead us to fruitful result.
- Applicable when mean is defined.
- Algorithm fails for non-linear data set.

K-Means++

K-Means is quite sensitive to initialization of centroids or the mean points, if the initialization of centroid is not good, our algorithm is not able to make desired number of clusters. The final formed clusters depend on how initial centroids were picked. To overcome this problem, we use technique called K-Means++ which chooses initial centers so that they are statically close to final ones.

K-Means++ is a smart centroid initialization technique and the rest of the algorithm is the same as that of K-Means. The ***steps to initialize the centroids*** using K-Means++ are:

1. Randomly select the first centroid from the data points.
2. For each data point compute its distance from the nearest, previously chosen centroid.
3. Select the next centroid from the data points such that the probability of choosing a point as centroid is directly proportional to its squared-distance from the nearest, previously chosen centroid.
4. Repeat steps 2 and 3 until k-centroids have been chosen.

Example

Q. Cluster the following data points into three clusters using K-Means++ algorithm: $\mathbf{(7,4), (8,3), (5,9), (3,3), (1,3), (10,1)}$.

Solution:

Centroid Initialization:

Select the first cluster center randomly. Let $c_1 = (7,4)$

x	$d(x,c_1)^2$
(7,4)	-
(8,3)	2
(5,9)	29
(3,3)	17
(1,3)	37
(10,1)	18

x	$prob$
(7,4)	-
(8,3)	2/103 = 0.0194
(5,9)	29/103 = 0.2815
(3,3)	17/103 = 0.1650
(1,3)	**37/103 = 0.3592**
(10,1)	18/103 = 0.1747

d is the distance between two points calculated using *Euclidean distance:*

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The data point (1,3) has the highest probability of being selected as cluster center. Thus, $c_2 = (1,3)$.

Again,

x	$Min(d(x,c_1)^2, d(x,c_2)^2)$
(7,4)	-
(8,3)	Min(2, 49) = 2
(5,9)	Min(29, 52) = 29
(3,3)	Min(17, 4) = 4
(1,3)	-
(10,1)	Min(18, 85) = 18

x	$prob$
(7,4)	-
(8,3)	2/53 = 0.0377
(5,9)	**29/53 = 0.5471**
(3,3)	4/53 = 0.0754
(1,3)	-
(10,1)	18/53 = 0.3396

The data point $(5,9)$ has the highest probability of being selected as next cluster center. Thus three initial cluster centers are: $c_1 = (7,4)$, $c_2 = (1,3), c_3 = (5,9)$.

Iteration 1:

Calculating distance between initial cluster centers and each data points:

	Distance to			**Point Belongs**
Point	$\mathbf{C1(7,4)}$	$\mathbf{C2(1,3)}$	$\mathbf{C3(5,9)}$	**to Cluster**
(7,4)	0	6.083	5.385	C1
(8,3)	1.414	7	6.708	C1
(5,9)	5.385	7.211	0	C3
(3,3)	4.123	2	6.324	C2
(1,3)	6.083	0	7.211	C2
(10,1)	4.243	9.219	9.434	C1

Thus after first iteration:

$Clusters\ 1 = \{(7,4), (8,3), (10,1)\}$,

$Cluster\ 2 = \{(3,3), (1,3)\}$,

$Cluster\ 3 = \{(5,9)\}$

Now, new cluster centers are:

$C1 = ((7+8+10)/3, (4+3+1)/3) = (8.33, 2.67)$
$C2 = ((3+1)/2, (3+3)/2) = (2,3)$
$C3 = (5,9)$

Iteration 2:

Calculating distance between new cluster centers and each data points:

	Distance to			Point Belongs
Point	$\boldsymbol{C1(8.33, 2.67)}$	$\boldsymbol{C2(2,3)}$	$\boldsymbol{C3(5,9)}$	**to Cluster**
(7,4)	1.881	5.099	5.385	C1
(8,3)	0.467	6	6.708	C1
(5,9)	7.152	6.708	0	C3
(3,3)	5.340	1	6.324	C2
(1,3)	7.337	1	7.211	C2
(10,1)	2.362	8.246	9.434	C1

Thus after second iteration:

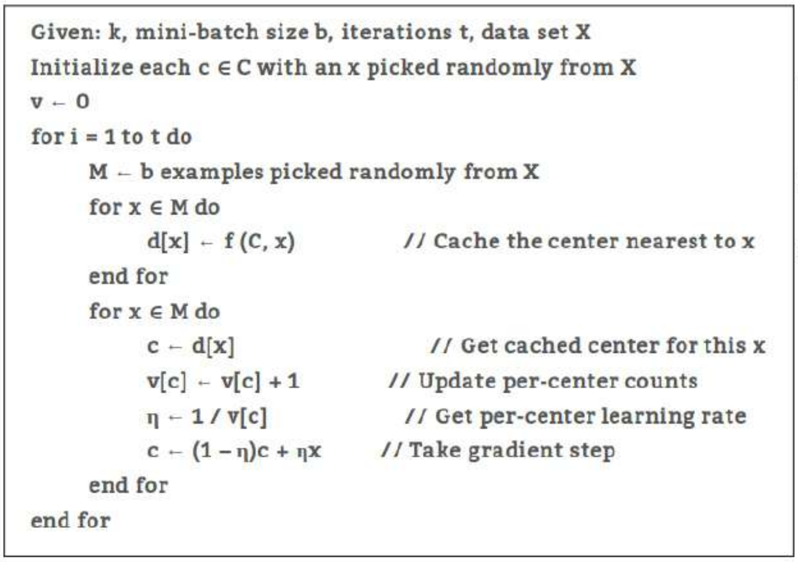
$Clusters\ 1 = \{(7,4), (8,3), (10,1)\},$
$Cluster\ 2 = \{(3,3), (1,3)\},$
$Cluster\ 3 = \{(5,9)\}$

The cluster of the data points obtained in second iteration is same as first iteration, so terminate.

Mini-Batch K-Means

The Mini-Batch K-means clustering algorithm is a version of the K-means algorithm which can be used instead of the K-means algorithm when clustering on huge datasets. Sometimes it performs better than the standard K-means algorithm while working on huge datasets because it doesn't iterate over the entire dataset. It creates random batches of data to be stored in memory, then a random batch of data is collected on each iteration to update the clusters. The main advantage of using the Mini-batch K-means algorithm is that it reduces the computational cost of finding a cluster.

Mini Batch K-Means Algorithm



```
Given: k, mini-batch size b, iterations t, data set X
Initialize each c ∈ C with an x picked randomly from X
v ← 0
for i = 1 to t do
    M ← b examples picked randomly from X
    for x ∈ M do
        d[x] ← f (C, x)          // Cache the center nearest to x
    end for
    for x ∈ M do
        c ← d[x]                 // Get cached center for this x
        v[c] ← v[c] + 1          // Update per-center counts
        η ← 1 / v[c]             // Get per-center learning rate
        c ← (1 − η)c + ηx        // Take gradient step
    end for
end for
```

$C = set\ of\ cluster\ centers\ c\ with\ |C| = k$

$M = mini - batch$

$f(C, x)\ returns\ the\ nearest\ cluster\ center\ c \in C\ to\ x\ using\ Euclidean\ distance.$

K-Medoids Algorithm

K-Medoids algorithm is also called as ***Partitioning Around Medoid (PAM)*** algorithm. This algorithm represents a cluster by mediod. Medoid is the most centrally located object of the cluster, with minimum sum of distances to other points. *K-Means* algorithm selects the average of a cluster's points as its center whereas the *K-Medoid* algorithm always picks the actual data points from the clusters as their centers.

The *k-medoids* method is more robust than *k-means* in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean. However, its processing is more costly than the *k-means* method. Both methods require the user to specify *k*, the number of clusters.

Algorithm

1. Randomly choose *K* data points as the initial mediods.
2. Associate each remaining data points to the closest mediod by using any common distance measure.
3. While the cost decreases:
   For each mediod m, for each non-mediod data point o:
   - Swap m and o, associate each data point to the closest mediod, recompute the cost.
   - If the total cost is more than that in the previous step, undo the swap.

For simplicity, we will take Manhattan distance as our cost measure:

$$c = \sum_{C_i} \sum_{P_i \in C_i} |P_i - C_i|$$

Examples

Q. Cluster the following data set of five objects into two clusters i.e. k = 2 using K-Mediods algorithm. Data Points are {(5,6), (4,5), (4,6), (6,7), (7,8)}

Solution:

No. of clusters K = 2

Let $C_1 = (4,6)$ and $C_2 = (6,7)$ are two initial mediods. Suppose considering the Manhattan distance metric as the distance measure: $d = |x_2 - x_1| + |y_2 - y_1|$

Iteration 1:

Calculating distance between mediods and each non-mediods points:

Points	Distance to		Cluster
	$C_1(4,6)$	$C_2(6,7)$	Assignment
(5,6)	1	2	C1
(4,5)	1	4	C1
(4,6)	-	-	C1
(6,7)	-	-	C2
(7,8)	5	2	C2

Thus after first iteration:

$Cluster\ 1 = \{(5,6), (4,5), (4,6)\}$

$Cluster\ 2 = \{(6,7), (7,8)\}$

$Total\ cost = (1 + 1) + (2) = 3$

Iteration 2:

Now let us choose some other point to be a mediod instead of (4, 6).
Let us randomly choose (4, 5).
Now the new mediod set is: $C_1 = (4,5)$ and $C_2 = (6,7)$

Calculating distance between new mediods and each non-mediod points:

Points	**Distance to**		**Cluster Assignment**
	$\boldsymbol{C_1(4,5)}$	$\boldsymbol{C_2(6,7)}$	
(5,6)	2	2	C1
(4,5)	-	-	C1
(4,6)	1	3	C1
(6,7)	-	-	C2
(7,8)	6	2	C2

Thus after second iteration:

$$Cluster\ 1 = \{(5{,}6), (4{,}5), (4{,}6)\}$$
$$Cluster\ 2 = \{(6{,}7), (7{,}8)\}$$

$$Total\ cost = (2 + 1) + (2) = 5$$

The cost at this time is 5, which is larger than the previous cost = 3. So undo the swap. Hence (4, 6) and (6, 7) are the final medoids. Therefore the clusters obtained finally are:

$$Cluster\ 1 = \{(5{,}6), (4{,}5), (4{,}6)\}$$
$$Cluster\ 2 = \{(6{,}7), (7{,}8)\}$$

Q. Cluster the following data set of five objects into two clusters i.e. k = 2. Data Points are {(1, 3), (4, 2), (6, 2), (3, 5), (4, 1)}

Solution:

No. of clusters K = 2
Let $C_1 = (1,3)$ and $C_2 = (6,2)$ are two initial mediods. Suppose considering the Manhattan distance metric as the distance measure:

$$d = |x_2 - x_1| + |y_2 - y_1|$$

Iteration 1:

Calculating distance between mediods and each non-mediod points:

Points	**Distance to**		**Cluster Assignment**
	$\boldsymbol{C_1(1,3)}$	$\boldsymbol{C_2(6,2)}$	
(1,3)	-	-	C1
(4,2)	4	2	C2
(6,2)	-	-	C2
(3,5)	4	6	C1
(4,1)	5	3	C2

Thus after first iteration:

$$Cluster\ 1 = \{(1{,}3), (3{,}5)\}$$
$$Cluster\ 2 = \{(4{,}2), (6{,}2), (4{,}1)\}$$

$$Total\ cost = (4) + (2 + 3) = 9$$

Iteration 2:

Now let us choose some other point to be a mediod instead of (1, 3).
Let us randomly choose (3, 5).
Now the new mediod set is: $C_1 = (3,5)$ and $C_2 = (6,2)$

Calculating distance between new mediods and each non-mediod points:

Points	**Distance to**		**Cluster Assignment**
	$\boldsymbol{C_1(3,5)}$	$\boldsymbol{C_2(6,2)}$	
(1,3)	4	6	C1
(4,2)	4	2	C2
(6,2)	-	-	C2
(3,5)	-	-	C1
(4,1)	5	3	C2

Thus after second iteration:

$Cluster\ 1 = \{(1,3),(3,5)\}$
$Cluster\ 2 = \{(4,2),(6,2),(4,1)\}$

$Total\ cost = (4) + (2 + 3) = 9$

The cost is same as previous, thus no undo swap.

Iteration 3:

Again let us choose some other point to be a mediod instead of (6, 2).
Let us randomly choose (4, 2).
Now the new mediod set is: $C_1 = (3,5)$ and $C_2 = (4,2)$

Calculating distance between new mediods and each non-mediod points:

Points	**Distance to**		**Cluster Assignment**
	$\boldsymbol{C_1(3,5)}$	$\boldsymbol{C_2(4,2)}$	
(1,3)	4	4	C1
(4,2)	-	-	C2
(6,2)	6	2	C2
(3,5)	-	-	C1
(4,1)	5	1	C2

Thus after third iteration:

$Cluster\ 1 = \{(1,3),(3,5)\}$
$Cluster\ 2 = \{(4,2),(6,2),(4,1)\}$

$Total\ cost = (4) + (2 + 1) = 7$

The cost at this time is 7, which is less than the previous cost = 9. So we will iterate again.

Iteration 4:

Lets swap (4, 2) and (4, 1)
Now the new mediod set is: $C_1 = (3,5)$ and $C_2 = (4,1)$.

Calculating distance between new mediods and each non-mediod points:

Points	Distance to		Cluster Assignment
	$C_1(3,5)$	$C_2(4,1)$	
(1,3)	4	5	C1
(4,2)	4	1	C2
(6,2)	6	3	C2
(3,5)	-	-	C1
(4,1)	-	-	C2

Thus after fourth iteration:

$$Cluster\ 1 = \{(1,3),(3,5)\}$$
$$Cluster\ 2 = \{(4,2),(6,2),(4,1)\}$$

$$Total\ cost = (4) + (3 + 1) = 8$$

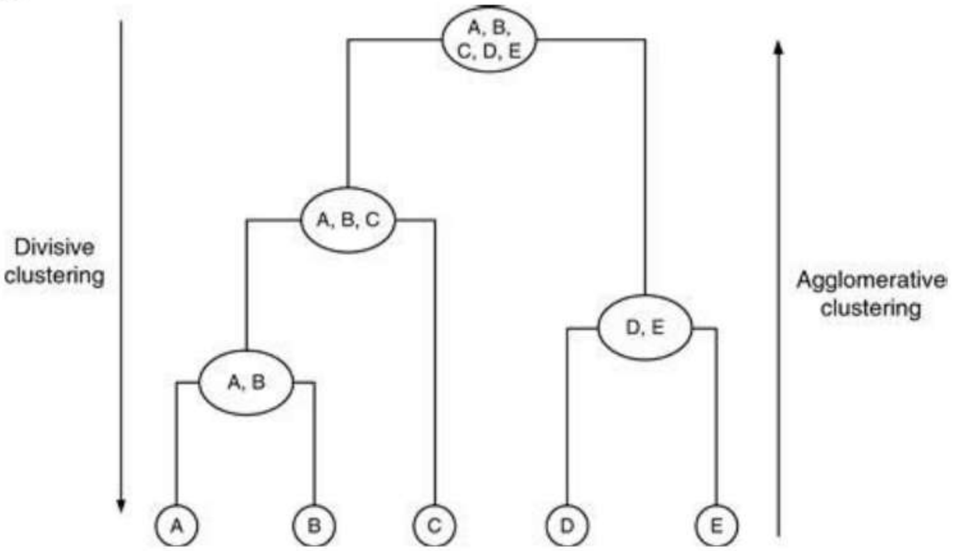
The cost at this time is 8, which is larger than the previous cost = 7. So undo the swap. Hence (3, 5) and (4, 2) are the final medoids. Therefore the clusters obtained finally are:

$$Cluster\ 1 = \{(1,3),(3,5)\}$$
$$Cluster\ 2 = \{(4,2),(6,2),(4,1)\}$$

Hierarchical Clustering

Hierarchical clustering or hierarchical cluster analysis (HCA) is a method of clustering which seeks to build a hierarchy of clusters in a given dataset. The hierarchical clustering technique has two approaches:

- ***Agglomerative:*** It is a *bottom-up approach*. In Agglomerative clustering each object creates its own cluster. The single clusters are merged to make larger cluster and the process of merging continues until the single cluster or required number of clusters are formed.
- ***Divisive:*** It is a *top-down approach*. In divisive hierarchical clustering method all objects are arranged within a big single cluster and the large cluster is continuously divided into smaller clusters until each cluster has single object or required numbers of clusters are formed.



The process of merging can be done using three techniques:

- ***Complete-linkage:*** the distance between two clusters is defined as the *longest* distance between two points in each cluster.

- ***Single-linkage:*** the distance between two clusters is defined as the *shortest* distance between two points in each cluster. This linkage may be used to detect high values in your dataset which may be outliers as they will be merged at the end.
- ***Average-linkage:*** the distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster.

Agglomerative Clustering Algorithm

1. Compute the distance matrix between the input data points.
2. Let each data points to be a cluster.
3. ***Repeat***
    - Merge the two clusters
    - Update the distance matrix
4. ***Until*** only *K* cluster remains

Example

Q. Cluster the data points (1, 1), (1.5, 1.5), (5, 5), (3, 4), (4, 4), (3, 3.5) into two clusters using single linkage Agglomerative clustering (nearest neighbor clustering).

Solution:

Assume A = (1, 1), B = (1.5, 1.5), C = (5, 5), D = (3, 4), E = (4, 4), F = (3, 3.5)

The distance matrix is:

	A	**B**	**C**	**D**	**E**	**F**
A	0					
B	0.71	0				
C	5.66	4.95	0			
D	3.61	2.92	2.24	0		
E	4.24	3.54	1.41	1.00	0	
F	3.20	2.50	2.50	0.50	1.12	0

In above table, the minimum value is 0.50 which is the distance between D and F. Thus, we group cluster D and F into (D, F). Then we update the distance matrix as below:

	A	**B**	**C**	**D, F**	**E**
A	0				
B	0.71	0			
C	5.66	4.95	0		
D, F	3.20	2.50	2.24	0	
E	4.24	3.54	1.41	1.00	0

$$d_{(D,F)\mapsto A} = \min\left(d_{DA}, d_{FA}\right) = \min\left(3.61, 3.20\right) = 3.20$$

$$d_{(D,F)\mapsto B} = \min\left(d_{DB}, d_{FB}\right) = \min\left(2.92, 2.50\right) = 2.50$$

$$d_{(D,F)\mapsto C} = \min\left(d_{DC}, d_{FC}\right) = \min\left(2.24, 2.50\right) = 2.24$$

$$d_{E\rightarrow(D,F)} = \min\left(d_{ED}, d_{EF}\right) = \min\left(1.00, 1.12\right) = 1.00$$

In updated distance matrix, the minimum value is 0.71 which is distance between cluster B and cluster A. Thus, we group cluster A and cluster B into a single cluster name (A, B). Now we update distance matrix as below:

	A, B	**C**	**D, F**	**E**
A, B	0			
C	4.95	0		
D, F	2.50	2.24	0	
E	3.54	1.41	1.00	0

$$d_{C\rightarrow(A,B)} = \min\left(d_{CA}, d_{CB}\right) = \min\left(5.66, 4.95\right) = 4.95$$

$$d_{(D,F)\mapsto(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}\right) = \min\left(3.61, 2.92, 3.20, 2.50\right) = 2.50$$

$$d_{E\rightarrow(A,B)} = \min\left(d_{EA}, d_{EB}\right) = \min\left(4.24, 3.54\right) = 3.54$$

In updated distance matrix, the minimum value is 1.00 which is distance between cluster E and cluster (D, F). Thus, we group cluster E and cluster (D, F) into a single cluster ((D, F), E). The updated distance matrix is given below:

	A, B	**C**	**(D, F), E**
A, B	0		
C	4.95	0	
(D, F), E	2.50	1.41	0

$$d_{((D,F),E)\mapsto(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}, d_{EA}, d_{EB}\right) = \min\left(3.61, 2.92, 3.20, 2.50, 4.24, 3.54\right) = 2.50$$

$$d_{((D,F),E)\mapsto C} = \min\left(d_{DC}, d_{FC}, d_{EC}\right) = \min\left(2.24, 2.50, 1.41\right) = 1.41$$

In updated distance matrix, the minimum value is 1.41 which is distance between cluster C and cluster ((D, F), E). Thus, we group them into a cluster (((D, F), E), C). The updated distance matrix is shown below:

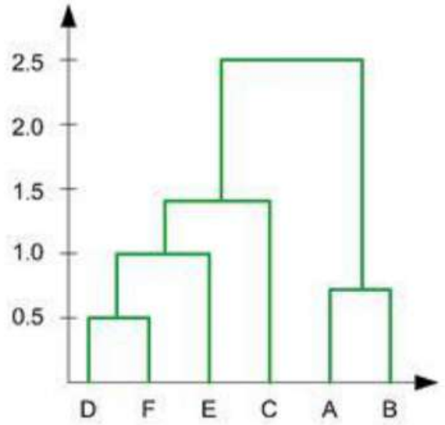
	A, B	**((D, F), E), C**
A, B	0	
((D, F), E), C	2.50	0

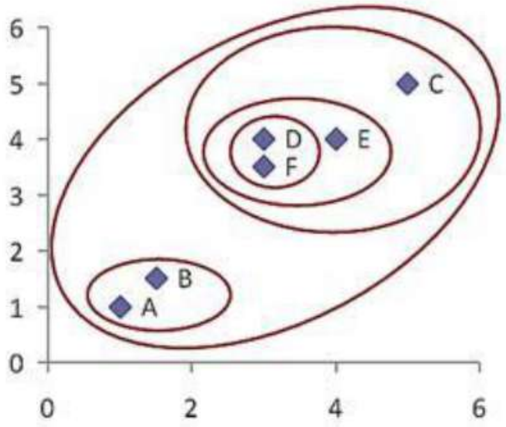
$$d_{(((D,F),E),C)\to(A,B)} = \min\left(3.61, 2.92, 3.20, 2.50, 4.24, 3.54, 5.66, 4.95\right) = 2.50$$

Required two clusters are: (A, B) and (((D, F), E), C).

If we merge the remaining two clusters, we will get only single cluster contain the whole 6 objects.

Using this information, we can now draw the final result of dendogram and clustering hierarchy into XY space:





Divisive Clustering Algorithm

1. Start with all data points in single cluster.
2. ***Repeat***
   - Choice of the cluster to be split.
   - Split of this cluster
3. ***Until*** only K cluster are created.

DBSCAN Clustering Algorithm

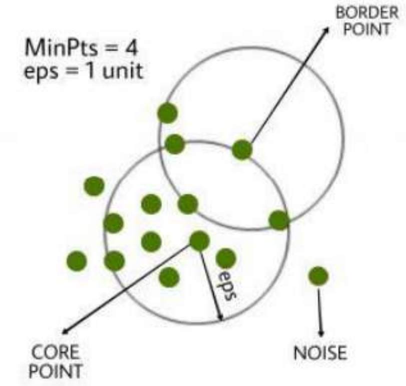
DBSCAN is density based clustering method. It stands for ***Density-based spatial clustering of applications with noise*** clustering method. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers. The main idea behind DBSCAN is that a point belongs to a cluster if it is close to many points from that cluster.

The DBSCAN algorithm uses two parameters:

- ***Epsilon (ε):*** The distance that specifies the neighborhoods. Two points are considered to be neighbors if the distance between them are less than or equal to $\boldsymbol{\varepsilon}$.
- ***minPts:*** The minimum number of points (a threshold) clustered together for a region to be considered dense.

Based on these two parameters, points are classified as core point, border point, or outlier:

- ***Core Point:*** Data point that has at least *minPts* number of points within *epsilon (ε)* distance.
- ***Border Point:*** Data point that has at least one core point within *epsilon (ε)* distance and lower than *minPts* number of points within *epsilon (ε)* distance from it.
- ***Noise or Outlier Point:*** Data point that has no core points within *epsilon (ε)* distance.



DBSCAN Algorithm Steps

Step 1: Label Core point and Noise point

- Select a random starting point, say $\boldsymbol{O}$.
- Identify neighborhood of this point $\boldsymbol{O}$ using the radius $\boldsymbol{\varepsilon}$.
- Count the number of points, say $\boldsymbol{k}$, in this neighborhood including point $\boldsymbol{O}$.
- If $\boldsymbol{k \geq minPts}$ then mark $\boldsymbol{O}$ as a core point.
- Else it will be marked as noise point.
- Select a new unvisited point and repeat the above steps.

Step 2: Check if noise point can become boundary point.

- If noise point is directly density reachable (i.e. within the boundary of radius $\boldsymbol{\varepsilon}$ from the core point), mark it as boundary points and it will form the part of the cluster.
- A point which is neither core point nor boundary point is marked as noise.

Examples

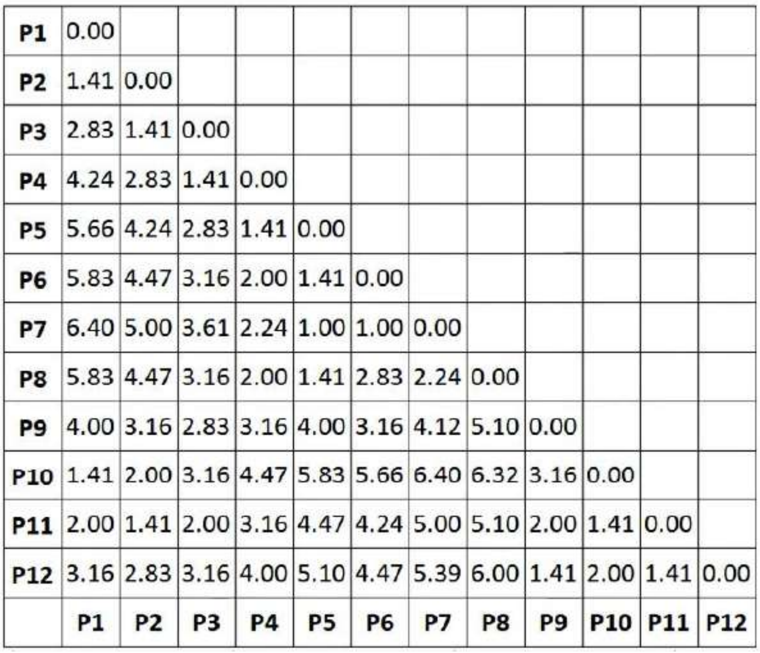
Q. Cluster the following data points $P1(3,7)$, $P2(4,6)$, $P3(5,5)$, $P4(6,4)$, $P5(7,3)$, $P6(6,2)$, $P7(7,2)$, $P8(8,4)$, $P9(3,3)$, $P10(2,6)$, $P11(3,5)$, $P12(2,4)$ using DBSCAN algorithm. Assume $eps(\varepsilon) = 1.9$ and $MinPts = 4$.

Solution:

To find the cluster by using DBSCAN we need to first calculate the distance among all pairs of given data point. Let us use *Euclidean distance* measure for distance calculation:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The distance matrix is:



P1	0.00											
P2	1.41	0.00										
P3	2.83	1.41	0.00									
P4	4.24	2.83	1.41	0.00								
P5	5.66	4.24	2.83	1.41	0.00							
P6	5.83	4.47	3.16	2.00	1.41	0.00						
P7	6.40	5.00	3.61	2.24	1.00	1.00	0.00					
P8	5.83	4.47	3.16	2.00	1.41	2.83	2.24	0.00				
P9	4.00	3.16	2.83	3.16	4.00	3.16	4.12	5.10	0.00			
P10	1.41	2.00	3.16	4.47	5.83	5.66	6.40	6.32	3.16	0.00		
P11	2.00	1.41	2.00	3.16	4.47	4.24	5.00	5.10	2.00	1.41	0.00	
P12	3.16	2.83	3.16	4.00	5.10	4.47	5.39	6.00	1.41	2.00	1.41	0.00
	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**

Point and their neighbours within the boundary of radius $\varepsilon = 1.9$ are given below:

$\boldsymbol{P1}: P2, P10$	$\boldsymbol{P2}: P1, P3, P11$	$\boldsymbol{P3}: P2, P4$	$\boldsymbol{P4}: P3, P5$
$\boldsymbol{P5}: P4, P6, P7, P8$	$\boldsymbol{P6}: P5, P7$	$\boldsymbol{P7}: P5, P6$	$\boldsymbol{P8}: P5$
$\boldsymbol{P9}: P12$	$\boldsymbol{P10}: P1, P11$	$\boldsymbol{P11}: P2, P10, P12$	$\boldsymbol{P12}: P9, P11$

Given, $minPts = 4$, Thus, Core points are: $P2, P5$ and $P11$

Outliers are: $P1, P3, P4, P6, P7, P8, P9, P10$ and $P12$

Check for direct density reachable condition for noise points (outliers). If density reachable condition is satisfied, convert noise to boundary point.

Points $P1, P3, P4, P6, P7, P8, P10$ and $P12$ become the boundary points.

Hence,
Core points are: $P2, P5$ and $P11$
Boundary points are: $P1, P3, P4, P6, P7, P8, P10$ and $P12$
Noise point: $P9$

Now constructing clusters:

$C_1 = \{P2, P1, P3, P11, P10, P12\} = \{P1, P2, P3, P10, P11, P12\}$

$C_2 = \{P5, P4, P6, P7, P8\} = \{P4, P5, P6, P7, P8\}$

Q. Perform DBSCAN on the given problem with $\varepsilon = 2$ and $minPt = 2$.

	X	***Y***
A	***2***	***10***
B	***2***	***5***
C	***8***	***4***
D	***5***	***8***
E	***7***	***5***
F	***6***	***4***
G	***1***	***2***
H	***4***	***9***

Solution:

To find the cluster by using DBSCAN we need to first calculate the distance among all pairs of given data point. Let us use *Euclidean distance* measure for distance calculation:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The distance matrix is:

A	0							
B	5	0						
C	8.49	6.08	0					
D	3.61	4.24	5	0				
E	7.07	5	1.41	3.61	0			
F	7.21	4.12	2	4.12	1.41	0		
G	8.06	3.16	7.28	7.21	6.71	5.39	0	
H	2.24	4.47	6.4	1.42	5	5.39	7.62	0
	A	***B***	***C***	***D***	***E***	***F***	***G***	***H***

Point and their neighbours within the boundary of radius $\varepsilon = 2$ are given below:

$\boldsymbol{A}$: {}	$\boldsymbol{B}$: {}	$\boldsymbol{C}: E, F$	$\boldsymbol{D}: H$
$\boldsymbol{E}: C, F$	$\boldsymbol{F}: C, E$	$\boldsymbol{G}$: {}	$\boldsymbol{H}: D$

Given, $minPt = 2$. Thus, Core points are: C, D, E, F, H

Outliers are: A, B, G

There are 2 clusters formed:

$C_1 = \{C, E, F\}$

$C_2 = \{D, H\}$

<u>Advantages of DBSCAN Algorithm</u>

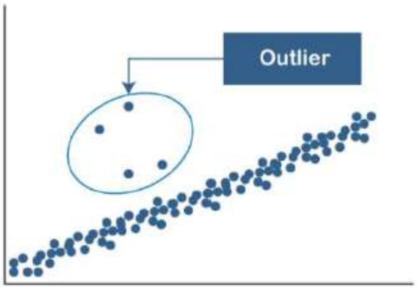
- Does not require a-priori specification of number of clusters.
- Able to identify noise data while clustering.
- DBSCAN algorithm is able to find arbitrarily size and arbitrarily shaped clusters.

<u>Disadvantages of DBSCAN Algorithm</u>

- Cannot cluster well with huge differences in densities.
- Does not work well in case of high dimensional data.
- Can be confused when there's a border point that belongs to two clusters.
- Choosing a meaningful *eps* value can be difficult if the data isn't well understood.

Outlier Analysis

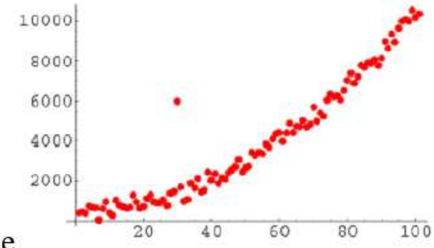
An ***outlier*** is a data point in a data set that is distant from all other observations. A data point that lies outside the overall distribution of the dataset. Or we can say, an outlier is something that behaves differently from the combination/collection of the data.



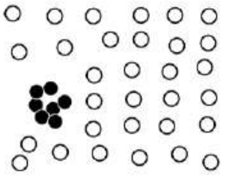
Outlier analysis is the process of identifying outliers, or abnormal observations, in a dataset. Also known as *outlier detection*, it's an important step in data analysis, as it removes erroneous or inaccurate observations which might otherwise skew conclusions. It has various applications in fraud detection, such as unusual usage of credit card or telecommunication services, Healthcare analysis for finding unusual responses to medical treatments, and also to identify the spending nature of the customers in marketing.

Types of Outliers

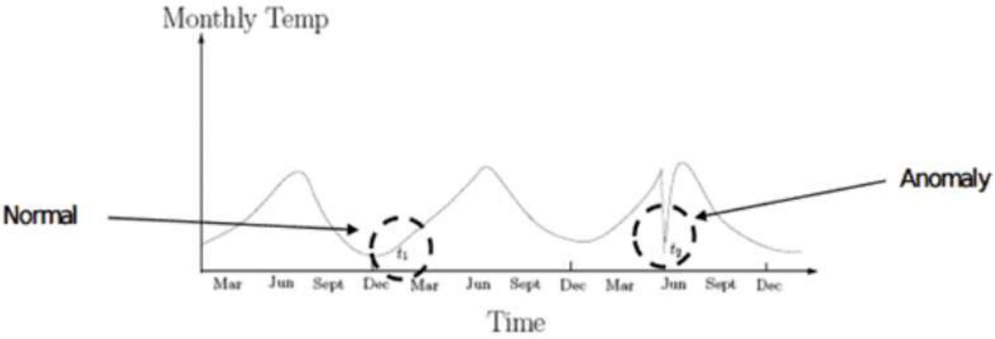
- ***Global Outliers (Point Outliers):*** If, in a given dataset, a data point strongly deviates from all the rest of the data points, it is known as a global outlier.



- ***Collective Outliers:*** If in a given dataset, some of the data points, as a whole, deviate significantly from the rest of the dataset, they may be termed as collective outliers. Here, the individual data objects may not be outliers, but when seen as a whole, they may behave as outliers.



- ***Contextual Outliers (Conditional Outliers):*** If in a given dataset, a data object deviates significantly from the other data points based on a specific context or condition only. A data point may be an outlier due to a certain condition and may show normal behavior under another condition. Contextual outliers are basically hard to spot if there was no background information.



Outlier Analysis Techniques

There are a variety of ways to find outliers. All these methods employ different approaches for finding values that are unusual compared to the rest of the dataset. Here we'll look at just a few of these techniques are as follows:

- ***Sorting:*** The method involves sorting the data according to their magnitude in any of the tools used for data manipulation. Observation of data could then lead towards identifying any objects that have a value of quite a higher range or a lower range. These objects could be treated as outliers. Let us consider an example of age dataset:

  Age: 1, 21, 25, 36, 43, 55, 56, 78, 398

  In the above dataset, we have sort the age dataset and get to know that 398 is an outlier. Sorting data method is most effective on the small dataset.

- ***Data Graphing:*** The technique involves the use of a graph to plot all the data points. This will allow the observer to visualize which data points are actually diverging from the other objects in the dataset. We can detect outliers with the help of graphical representation like Scatter plot and Boxplot.

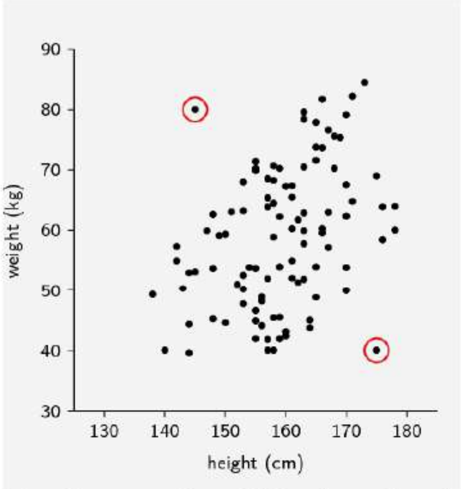
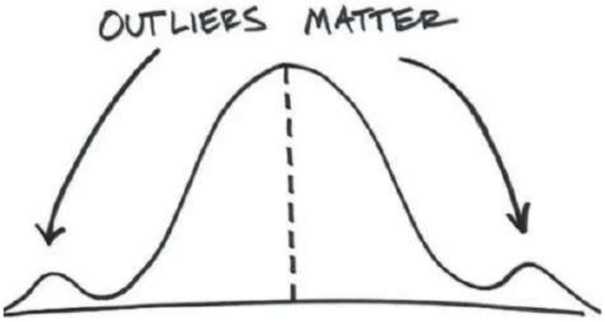


Fig: Scatter plot of Weight vs height

- ***Z-Score:*** The Z-Score is used to identify how much the data points are deviating from the mean of the sample through calculating the standard deviations of the points. A Gaussian distribution is assumed in this case. The outliers are the data points that are in the tails of the distribution and therefore far from the mean. By calculating the Z-score for each data point, it's easy to see which data points are placed far from the mean.

  Formula for Z score = (Observation — Mean)/Standard Deviation
  i.e. $z = (X — \mu) / \sigma$



- ***DBSCAN:*** The method is a clustering approach and is referred to as the Density-Based Spatial Clustering of Applications with Noise. Here, all data points are defined either as Core Points, Border Points or Noise Points. ***Core Points*** are data points that have at least MinPts neighboring data points within a distance ε. ***Border Points*** are neighbours of a Core Point within the distance ε but with less than MinPts neighbours within the distance ε. All other data points are ***Noise Points***, also identified as ***outliers***. Outlier detection thus depends on the required number of neighbours MinPts, the distance ε and the selected distance measure, like Euclidean or Manhattan.

Classification vs. Clustering

Classification	***Clustering***
Process of classifying the input instances based on their corresponding class labels.	Grouping the instances based on their similarity without the help of class labels
Classification is a type of supervised learning method.	Clustering is a kind of unsupervised learning method.
The number of classes is known.	The number of classes is unknown.
It uses a training dataset.	It does not use a training dataset.
In classification, there are labels for training data.	In clustering, there are no labels for training data.
Its objective is to find which class a new object belongs to form the set of predefined classes.	Its objective is to group a set of objects to find whether there is any relationship between them.
More complex as compared to clustering.	Less complex as compared to classification.
Popular algorithms for classification include Naive Bayes Classifier, Decision Trees, Support Vector Machines etc.	Popular algorithms used for clustering include K-Means, DBSCAN etc.

Please let me know if I missed anything or anything is incorrect.

poudeljayanta99@gmail.com