Symbiosis Statistical Institute, Pune

A project Report on

HIERARCHICHAL CLUSTERING AND DENSITY BASED CLUSTERING

Presenters:

JOY ANGELIN - 21060641022

SNEHA M - 21060641049

INTRODUCTION

The main objective of this project is to learn Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

What is Supervised Learning data?

If you're learning a task under supervision, someone is present judging whether you're getting the right answer. Similarly, in supervised learning, that means having a full set of labelled data while training an algorithm.

Fully labelled means that each example in the training dataset is tagged with the answer the algorithm should come up with on its own. So, a labelled dataset of flower images would tell the model which photos were of roses, daisies and daffodils. When shown a new image, the model compares it to the training examples to predict the correct label.

What Is Unsupervised Learning data?

Clean, perfectly labelled datasets aren't easy to come by. And sometimes, researchers are asking the algorithm questions they don't know the answer to. That's where unsupervised learning comes in.

In unsupervised learning, a deep learning model is handed a dataset without explicit instructions on what to do with it. The training dataset is a collection of examples without a specific desired outcome or correct answer. The neural network then attempts to automatically find structure in the data by extracting useful features and analysing its structure.

What is clustering?

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**.

Clustering was originally used by anthropologists aiming to explain origin of human beings. Later was adapted by psychologist, intelligence and others.

Types of clustering?

Since there are many types of clustering. Few of those we mentioned were: -

- Connectivity-based Clustering (Hierarchical clustering)
- Centroids-based Clustering (Partitioning methods)
- Distribution-based Clustering.
- Density-based Clustering (Model-based methods)
- Fuzzy Clustering.
- Constraint-based (Supervised Clustering)
- Hierarchical Clustering

Real life examples of Clustering: -

1. Marketing and Sales

Personalization and targeting in marketing is big business.

This is achieved by looking at specific characteristics of a person and sharing campaigns with them that have been successful with other similar people.

What the problem is: If you are a business trying to get the best return on your marketing investment, it is crucial that you target people in the right way. If you get it wrong, you risk not making any sales, or worse, damaging your customer trust.

How clustering works: Clustering algorithms are able to group together people with similar traits and likelihood to purchase. Once you have the groups, you can run tests on each group with different marketing copy that will help you better target your messaging to them in the future.

2.spam filter

You know the junk folder in your email inbox? It is the place where emails that have been identified as spam by the algorithm.

Many machine learning courses, such as Andrew Ng's famed Coursera course, use the spam filter as an example of unsupervised learning and clustering.

What the problem is: Spam emails are at best an annoying part of modern-day marketing techniques, and at worst, an example of people phishing for your personal data. To avoid getting these emails in your main inbox, email companies use algorithms. The purpose of these algorithms is to flag an email as spam correctly or not.

How clustering works: K-Means clustering techniques have proven to be an effective way of identifying spam. The way that it works is by looking at the different sections of the email (header, sender, and content). The data is then grouped together.
These groups can then be classified to identify which are spam. Including clustering in the classification process improves the accuracy of the filter to 97%. This is excellent news for people who want to be sure they're not missing out on your favourite newsletters and offers.

3. Identifying fraudulent or criminal activity

In this scenario, we are going to focus on fraudulent taxi driver behaviour. However, the technique has been used in multiple scenarios.

What is the problem: You need to look into fraudulent driving activity? The challenge is how do you identify what is true and which is false?

How clustering works: By analysing the GPS logs, the algorithm is able to group similar behaviours. Based on the characteristics of the groups you are then able to classify them into those that are real and which are fraudulent.

What is hierarchical clustering?

Hierarchical clustering*,* also known as *hierarchical cluster analysis,* is an algorithm that groups similar objects into groups called *clusters*. The endpoint is a set of clusters*,* where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

How hierarchical clustering works: -

Hierarchical clustering starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps: (1) identify the two clusters that are closest together, and (2) merge the two most similar clusters. This iterative process continues until all the clusters are merged together.

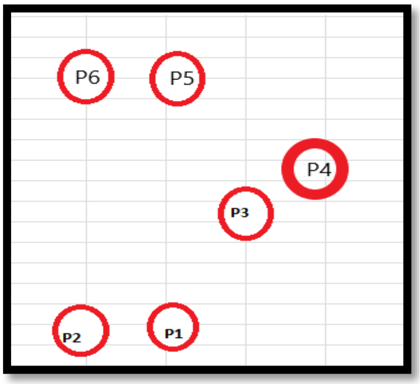
There are 2 types of hierarchical clustering: -

- AGGLOMERATIVE CLUSTERING –
- This is a "bottom-up" approach: each observation starts in its own cluster.
- **Pairs** of clusters are merged as one moves up the hierarchy.

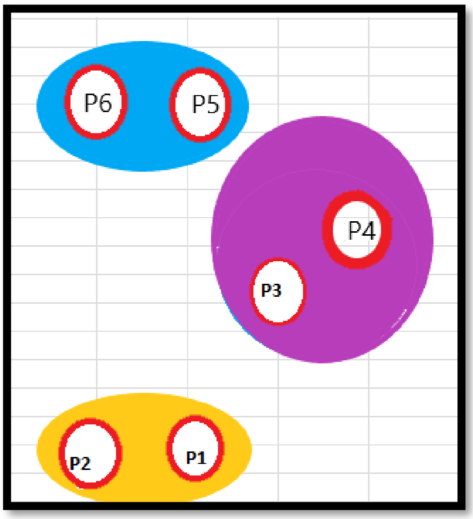
- DIVISIVE CLUSTERING-
- This is a "top-down" approach: all observations start in one cluster.
- **Splits** are performed recursively as one moves down the hierarchy.

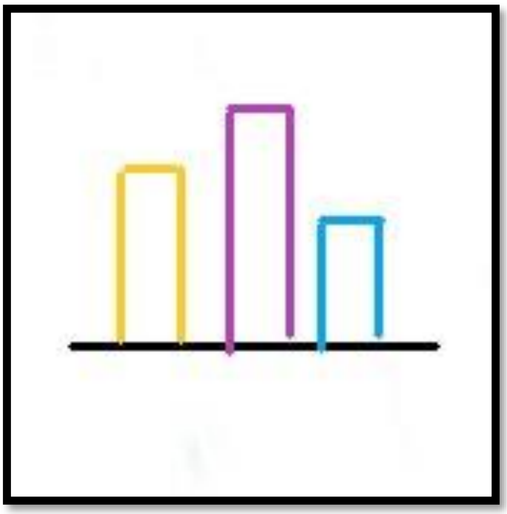
How to make a Dendrogram: -

- Each Data Point is a cluster of its own
- We try to find the least distance between two clusters.

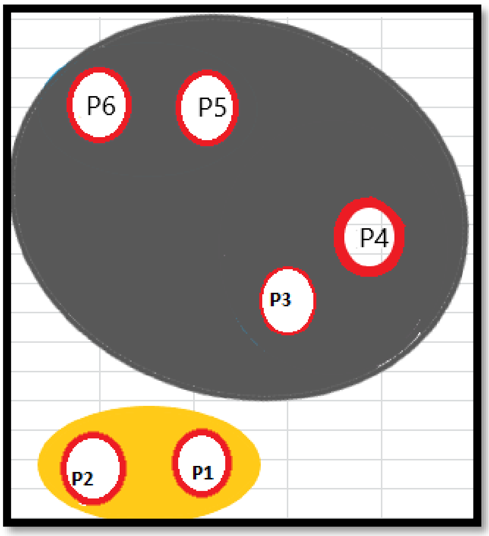


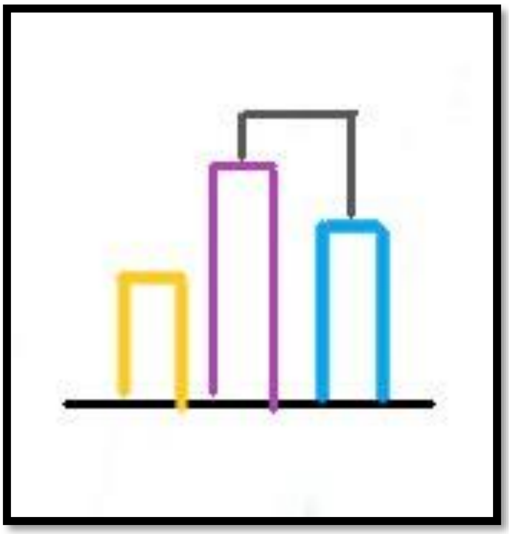
- The two nearest clusters are merged together.



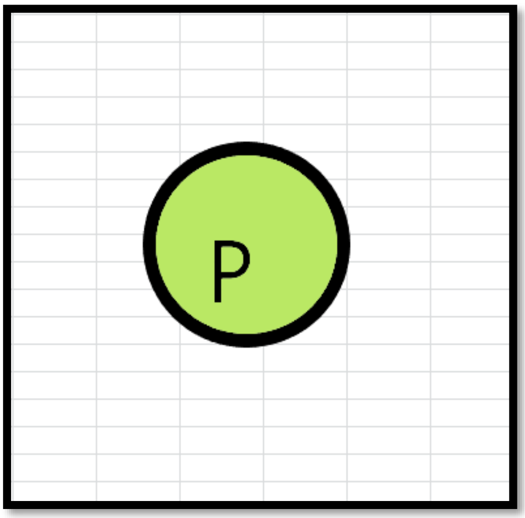


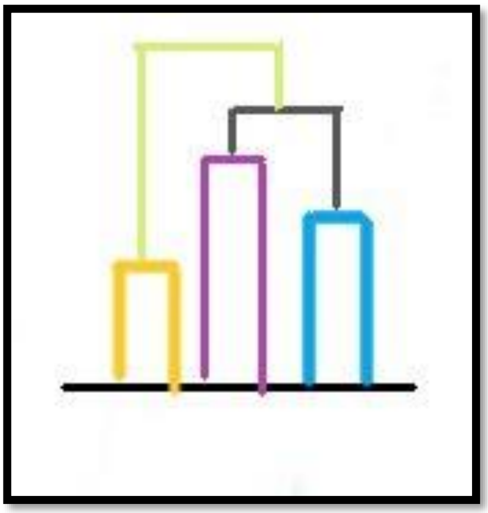
- The two nearest clusters are merged together.





- We terminate to when we are left with only one cluster.



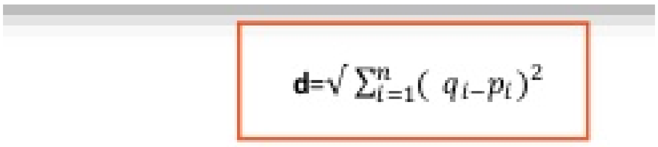


How to measure distance between points?

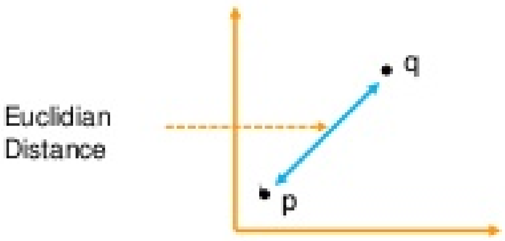
There are 4 ways in which distance can be measured, they are: -

1) **Euclidean Distance Measure-**

- It is most used distance measure
- The Euclidean distance is the ordinary straight line
- It is the distance between two points in Euclidean space

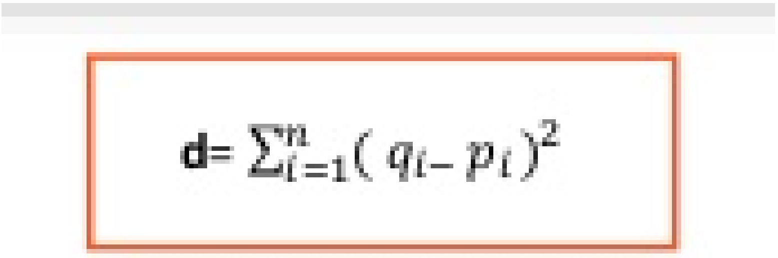


$$d=\sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}$$



2) **Squared Euclidean Distance Measure-**

- Euclidean Squared distance metric is faster than clustering with the regular Euclidean distance.
- This method tends to create clusters of small size.
- The Euclidean squared distance metric uses the same equation as the Euclidian distance metric but does not take square root.



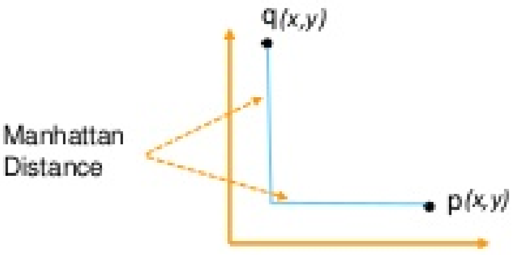
$$d= \sum_{i=1}^{n}(q_i- p_i)^2$$

3) Manhattan Distance Measure-

The Manhattan distance is a simple sum of the horizontal and vertical components or the distance between two points measured along axis at right angles.

Manhattan distance is usually preferred when there is high dimensionality in the data

$$d= \sum_{i=1}^{n} | q_x - p_x | + | q_y - p_y |$$

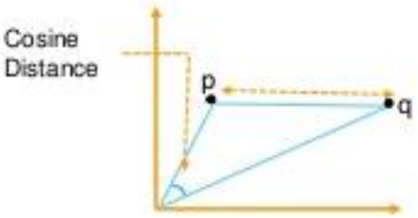


4) Cosine Distance Measure-

The cosine distance similarity measures the angle between two vectors.

Cosine **distance** metric is mainly used to find the amount of similarity between two data points.

$$d= \frac{\sum_{i=0}^{n-1} q_i - p_x}{\sum_{i=0}^{n-1} (q_i)^2 \times \sum_{i=0}^{n-1} (p_i)^2}$$



What is divisive clustering?

The **divisive clustering** algorithm is a top-down **clustering** approach, initially, all the points in the dataset belong to one **cluster** and split is performed recursively as one moves down the hierarchy.

- For each split compute cluster sum of squares.

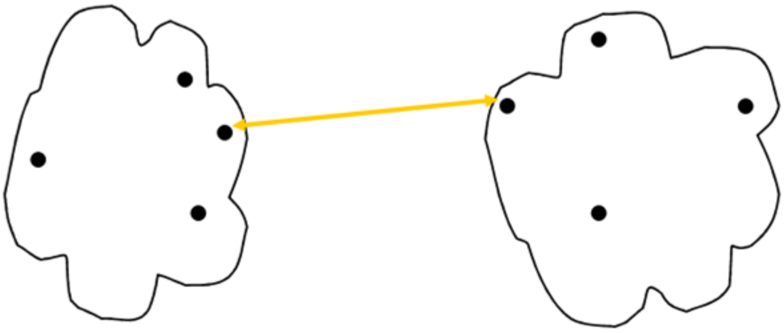
$$B_{j12} = n_1(\bar{x}_{j1} - \bar{x}_j)^2 + n_2(\bar{x}_{j2} - \bar{x}_j)^2$$

Measure for the distance between two clusters: -

1. **Single linkage: -**

Can handle non elliptical shapes.

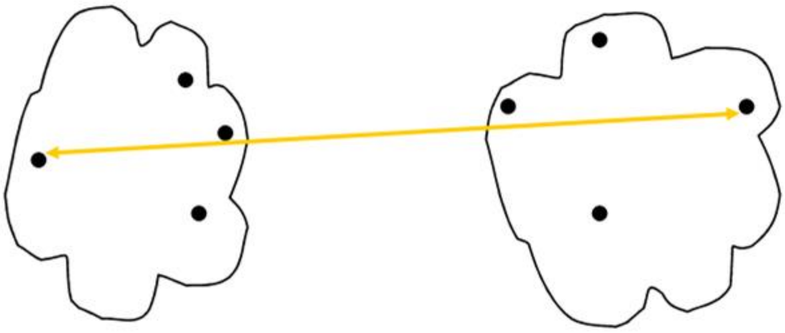
Sensitive to noise and outliers



2. **Complete linkage: -**

Less susceptible to noise and outliers

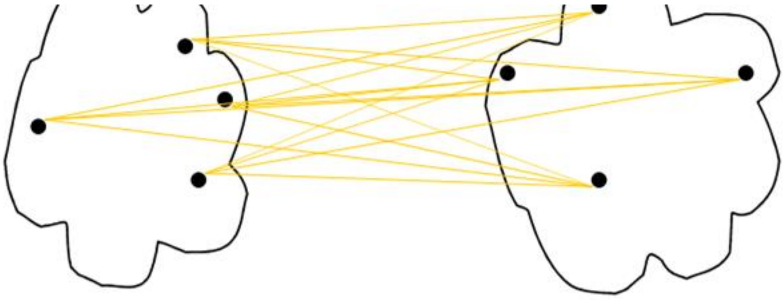
Tends to break large clusters and biased towards globular clusters



3. **Average linkage: -**

Less susceptible to noise and outliers

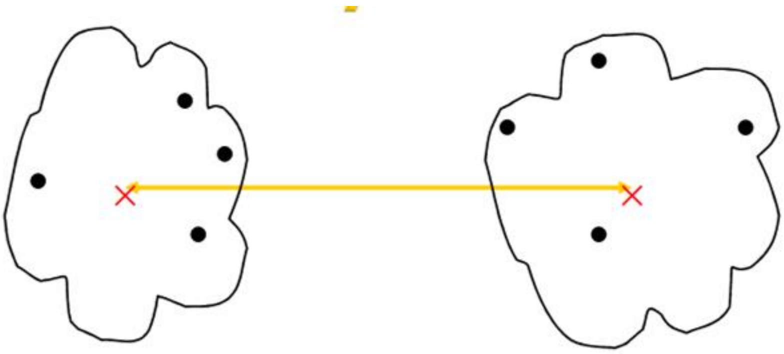
Biased towards globular clusters



4. **Centroid linkage: -**

Does well in separating clusters if there is any noise between the clusters.

Biased towards globular clusters



How to perform hierarchical clustering?

Steps are: -

- IMPORT THE DATA SET
- CREATE A SCATTER PLOT
- NORMALIZE THE DATA
- CALCULATE EUCLIDEAN DISTANCE
- CREATE A DENDOGRAM

The data set we choose was of an Indian oil organization who needs to know its sales in various citied of India and cluster the cities based on the sales.

R Demo 1: -

```
mydata <- read.csv("C:\\Users\\Admin\\Downloads\\mldata (1).csv")

str(mydata)

head(mydata)



names(mydata)[1]=("City")

mydata
```

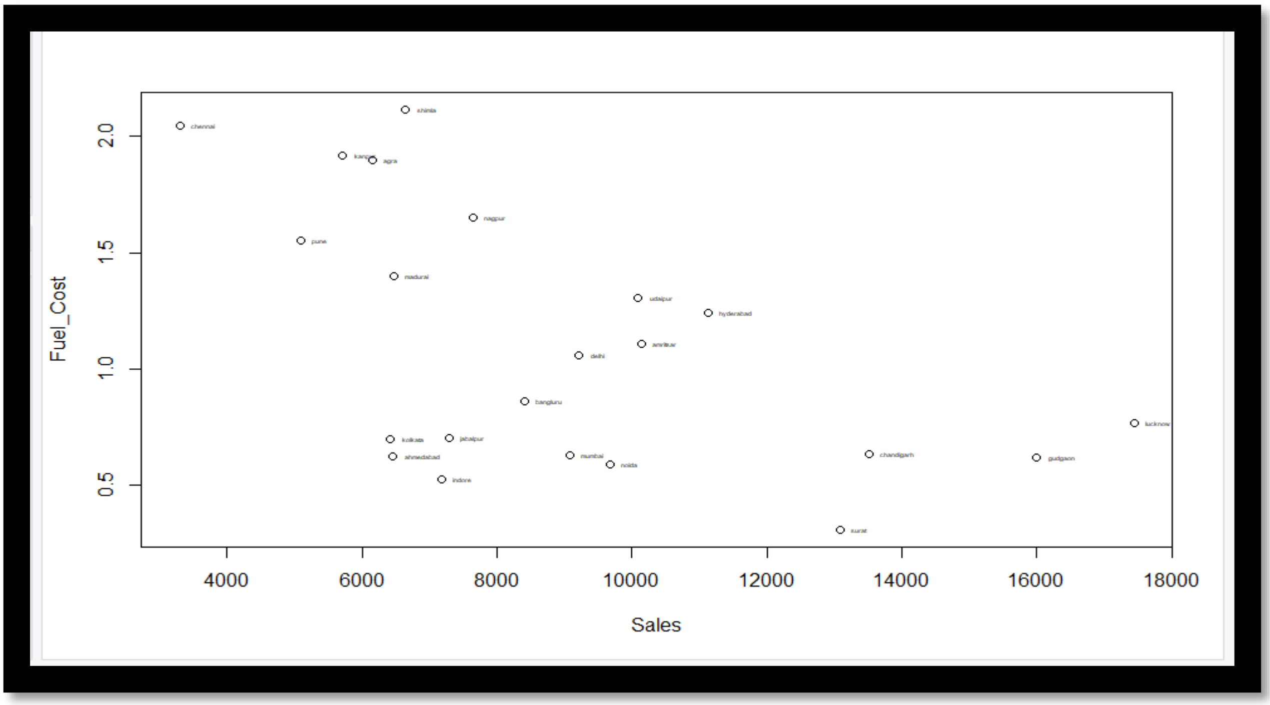
CITY	Fixed_charge	RoR	Cost	Load	D Demand	Sales	Nuclear
mumbai	1.06	9.2	151	54.4	1.6	9077	0
pune	0.89	10.3	202	57.9	2.2	5088	25.3
delhi	1.43	15.4	113	53	3.4	9212	0
kolkata	1.02	11.2	168	56	0.3	6423	34.3
chennai	1.49	8.8	192	51.2	1	3300	15.6
hyderabad	1.32	13.5	111	60	-2.2	11127	22.5
nagpur	1.22	12.2	175	67.6	2.2	7642	0

surat	1.1	9.2	245	57	3.3	13082	0
bangluru	1.34	13	168	60.4	7.2	8406	0
ahmedabad	1.12	12.4	197	53	2.7	6455	39.2
lucknow	0.75	7.5	173	51.5	6.5	17441	0
agra	1.13	10.9	178	62	3.7	6154	0
indore	1.15	12.7	199	53.7	6.4	7179	50.2
noida	1.09	12	96	49.8	1.4	9673	0
madurai	0.96	7.6	164	62.2	-0.1	6468	0.9
gudgaon	1.16	9.9	252	56	9.2	15991	0
kanpur	0.76	6.4	136	61.9	9	5714	8.3
amritsar	1.05	12.6	150	56.7	2.7	10140	0
chandigarh	1.16	11.7	104	54	-2.1	13507	0
jabalpur	1.2	11.8	148	59.9	3.5	7287	41.1
shimla	1.04	8.6	204	61	3.5	6650	0
udaipur	1.07	9.3	174	54.3	5.9	10093	26.6

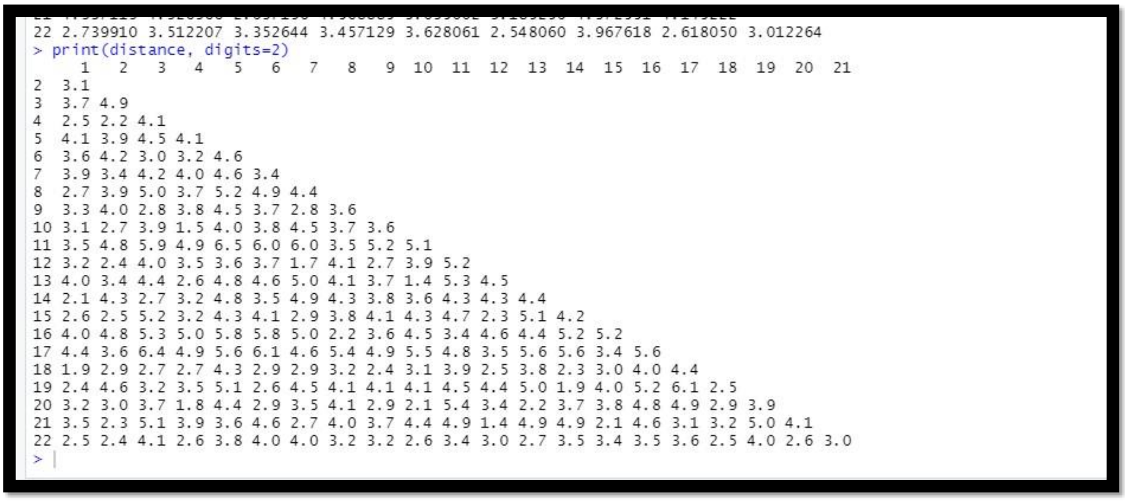
```
#scatter plot

plot(Fuel_Cost~Sales, mydata)

with(mydata,text(Fuel_Cost~Sales, labels = City, pos = 4, cex = .3))


plot(RoR~Sales, mydata)

with(mydata,text(RoR~Sales, labels = ï..CITY, pos = 4, cex = .3))


#normalization

z <- mydata[,-c(1)]

m <- apply(z,2,mean)
```

```
s <- apply(z,2,sd)
z <- scale(z,m,s)
head(z)

#calculate the Euclidean Distance
distance <- dist(z)
distance
print(distance, digits=2)

#clustering dendogram
hc.l <- hclust(distance)
plot(hc.l)
plot(hc.l, labels= mydata$City, hang =-1)

#clustering dendogram average
hc.l <- hclust(distance, method = "average")
plot(hc.l, labels= mydata$City, hang =-1)
```

scatter plot of fuel cost and sales

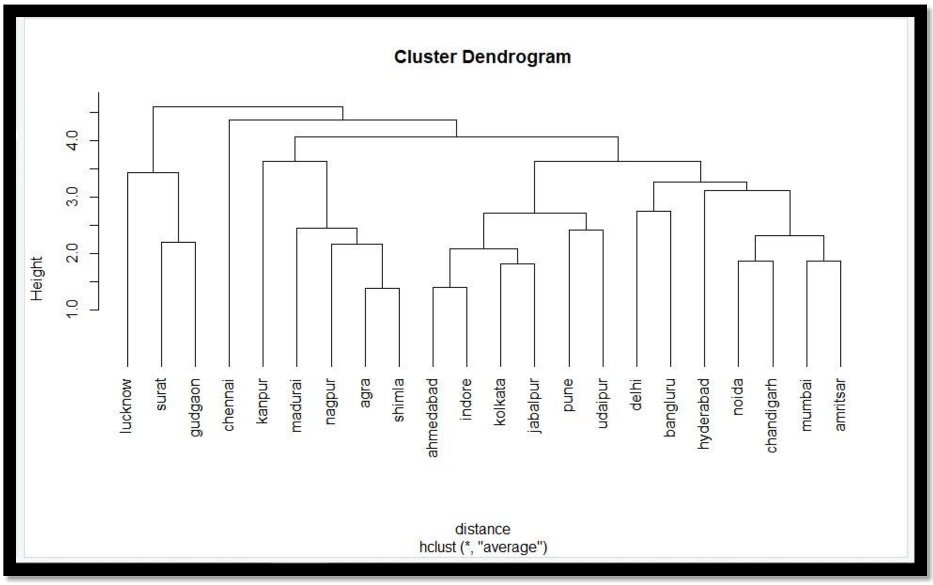


Distance matrix

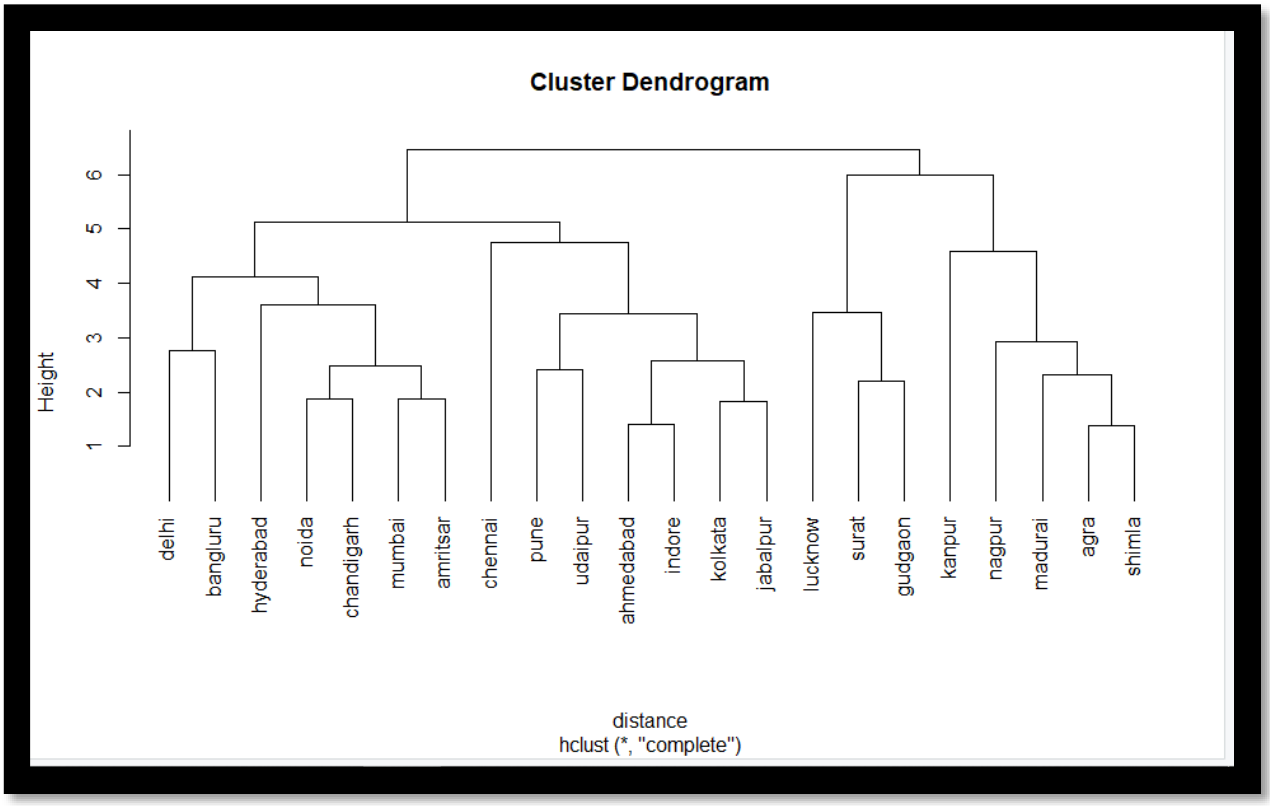


```
22 2.739910 3.512207 3.352644 3.457129 3.628061 2.548060 3.967618 2.618050 3.012264
> print(distance, digits=2)
    1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
2  3.1
3  3.7 4.9
4  2.5 2.2 4.1
5  4.1 3.9 4.5 4.1
6  3.6 4.2 3.0 3.2 4.6
7  3.9 3.4 4.2 4.0 4.6 3.4
8  2.7 3.9 5.0 3.7 5.2 4.9 4.4
9  3.3 4.0 2.8 3.8 4.5 3.7 2.8 3.6
10 3.1 2.7 3.9 1.5 4.0 3.8 4.5 3.7 3.6
11 3.5 4.8 5.9 4.9 6.5 6.0 6.0 3.5 5.2 5.1
12 3.2 2.4 4.0 3.5 3.6 3.7 1.7 4.1 2.7 3.9 5.2
13 4.0 3.4 4.4 2.6 4.8 4.6 5.0 4.1 3.7 1.4 5.3 4.5
14 2.1 4.3 2.7 3.2 4.8 3.5 4.9 4.3 3.8 3.6 4.3 4.3 4.4
15 2.6 2.5 5.2 3.2 4.3 4.1 2.9 3.8 4.1 4.3 4.7 2.3 5.1 4.2
16 4.0 4.8 5.3 5.0 5.8 5.8 5.0 2.2 3.6 4.5 3.4 4.6 4.4 5.2 5.2
17 4.4 3.6 6.4 4.9 5.6 6.1 4.6 5.4 4.9 5.5 4.8 3.5 5.6 5.6 3.4 5.6
18 1.9 2.9 2.7 2.7 4.3 2.9 2.9 3.2 2.4 3.1 3.9 2.5 3.8 2.3 3.0 4.0 4.4
19 2.4 4.6 3.2 3.5 5.1 2.6 4.5 4.1 4.1 4.1 4.5 4.4 5.0 1.9 4.0 5.2 6.1 2.5
20 3.2 3.0 3.7 1.8 4.4 2.9 3.5 4.1 2.9 2.1 5.4 3.4 2.2 3.7 3.8 4.8 4.9 2.9 3.9
21 3.5 2.3 5.1 3.9 3.6 4.6 2.7 4.0 3.7 4.4 4.9 1.4 4.9 4.9 2.1 4.6 3.1 3.2 5.0 4.1
22 2.5 2.4 4.1 2.6 3.8 4.0 4.0 3.2 3.2 2.6 3.4 3.0 2.7 3.5 3.4 3.5 3.6 2.5 4.0 2.6 3.0
> |
```

Dendrogram using average method

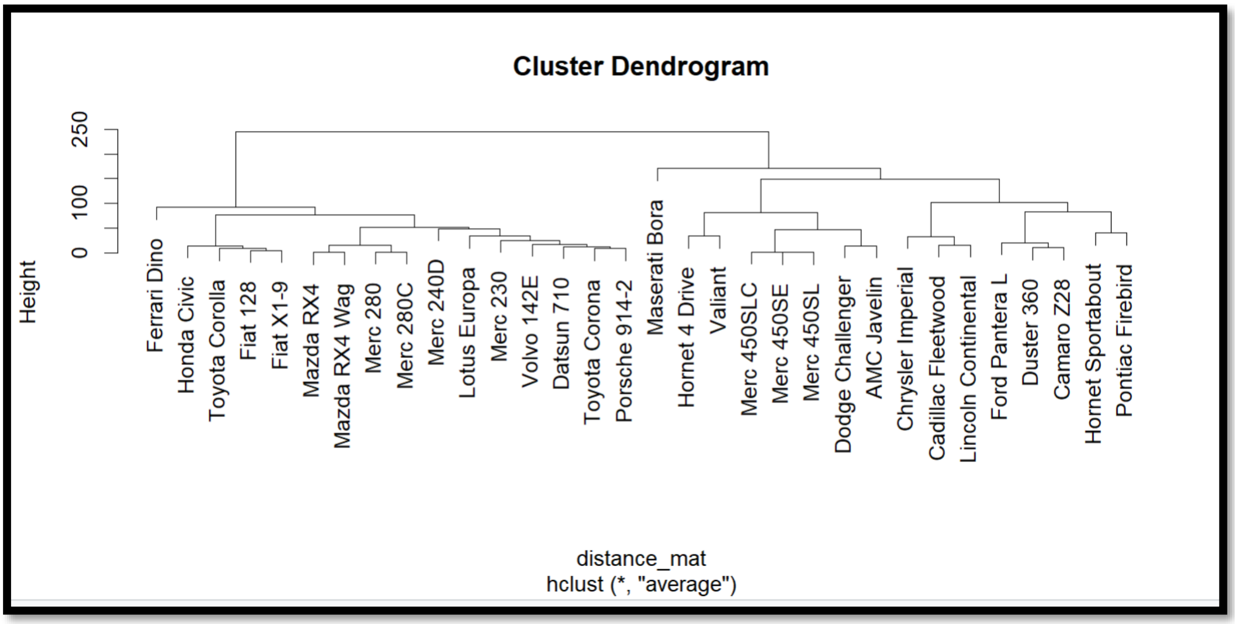


Dendrogram using complete linkage method

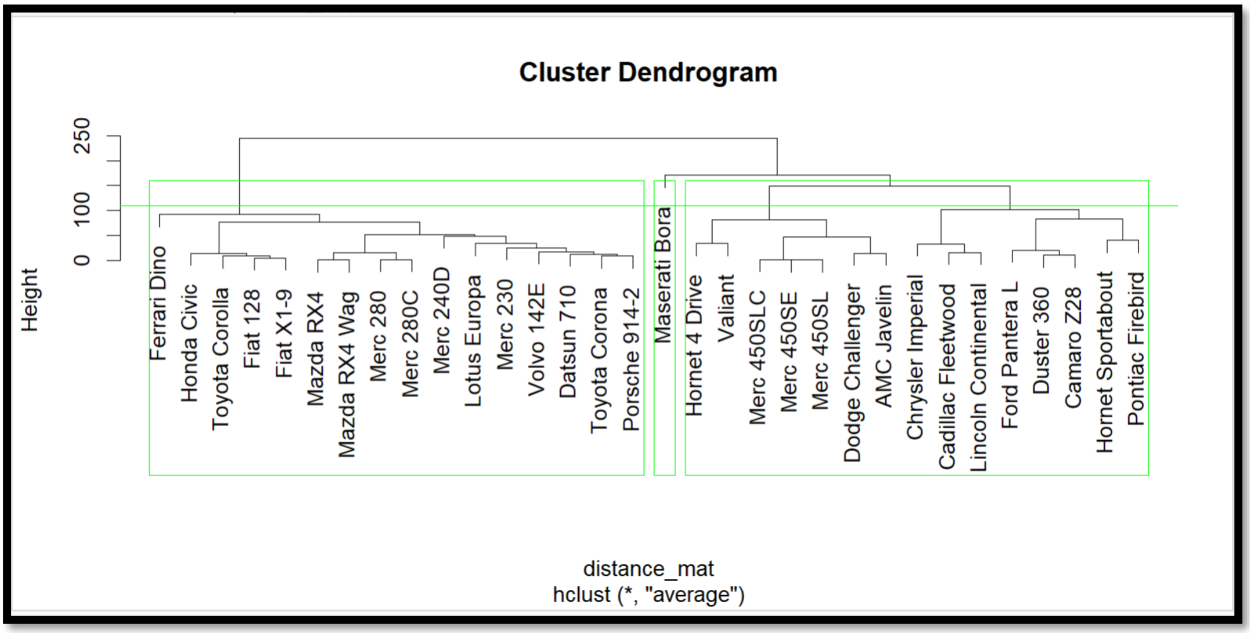


R Demo 2: -

```
#hierarchical clustering on mtcars dataset


# Installing the package

install.packages("dplyr")


# Loading package

library(dplyr)


# Summary of dataset in package

head(mtcars)


# Finding distance matrix

distance_mat <- dist(mtcars, method = 'euclidean')

distance_mat


# Fitting Hierarchical clustering Model to training dataset

Hierar_cl <- hclust(distance_mat, method = "average")

Hierar_cl


# Plotting dendrogram

plot(Hierar_cl)
```



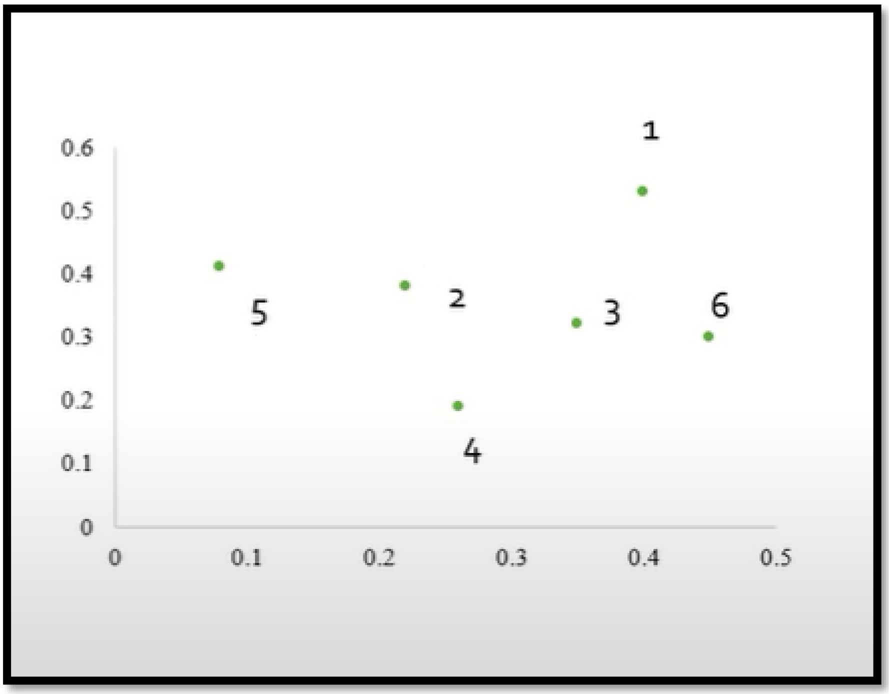
```
# Choosing no. of clusters

# Cutting tree by height

abline(h = 110, col = "green")


# Cutting tree by no. of clusters

fit <- cutree(Hierar_cl, k = 3 )

fit


table(fit)

rect.hclust(Hierar_cl, k = 3, border = "green"
```



How to calculate distance matrix?

- Find the clusters using single link technique.
- Use Euclidean distance

	X	Y
P1	0.4	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30



Euclidean Distance: -

Distance [P1 , P2] = [(x , y), (a , b)] = V $(x-a)^2$ + $(y-b)^2$

- Using distances we get a 'DISTANCE MATRIX'
- It is a Hollow Matrix.

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0

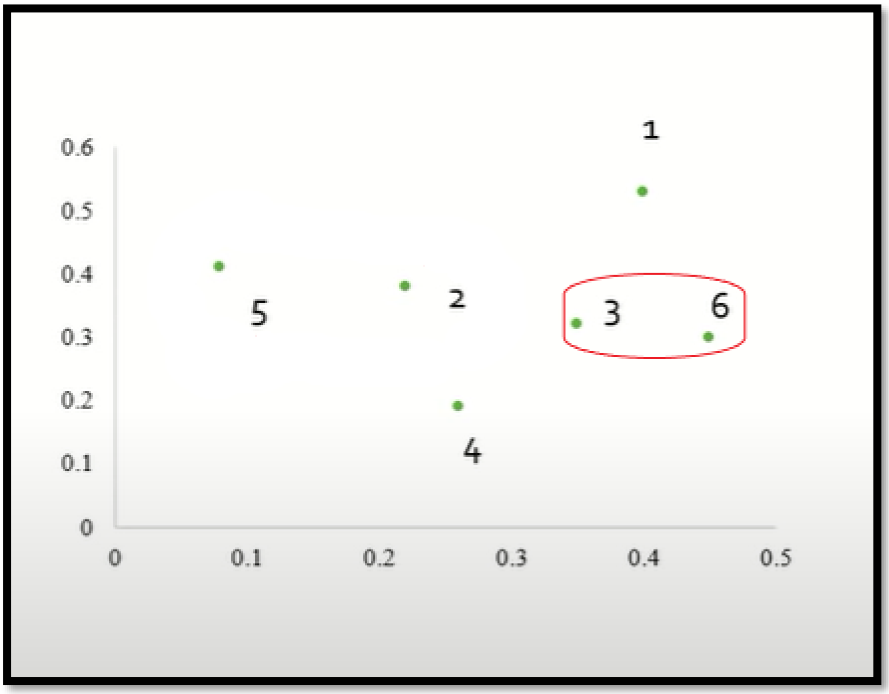
To update the distance matrix

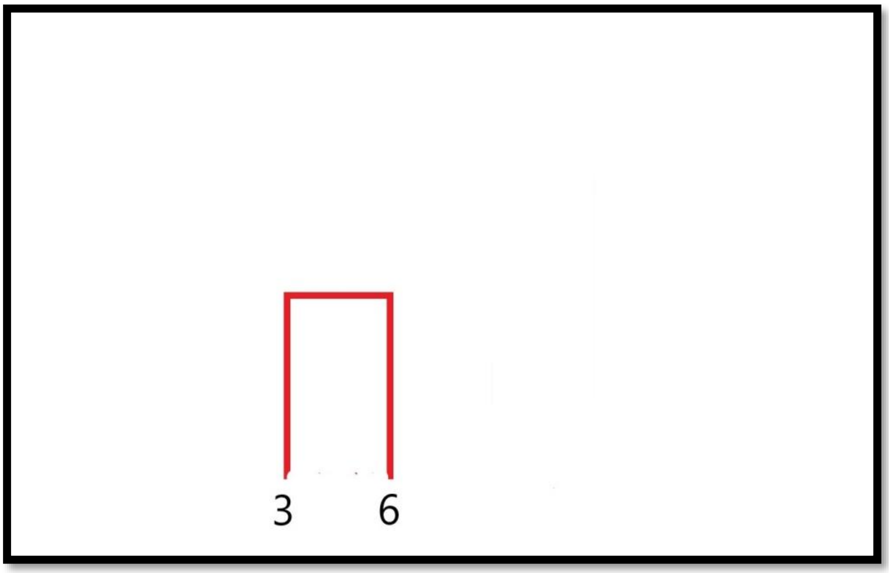
MIN [dist (P3 , P6) , point]

= MIN [dist (P3,point) , (P6,point)]

- This is new Distance

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0

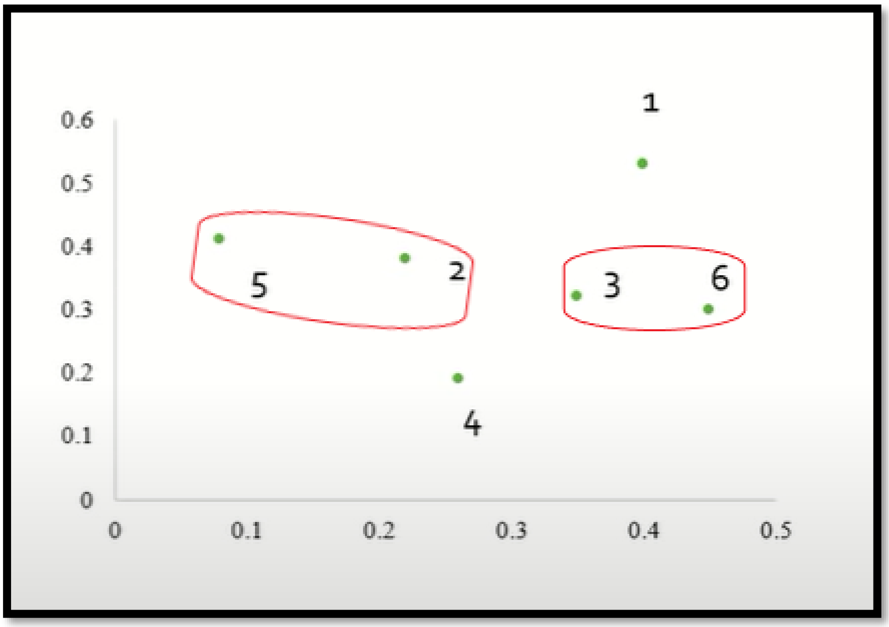


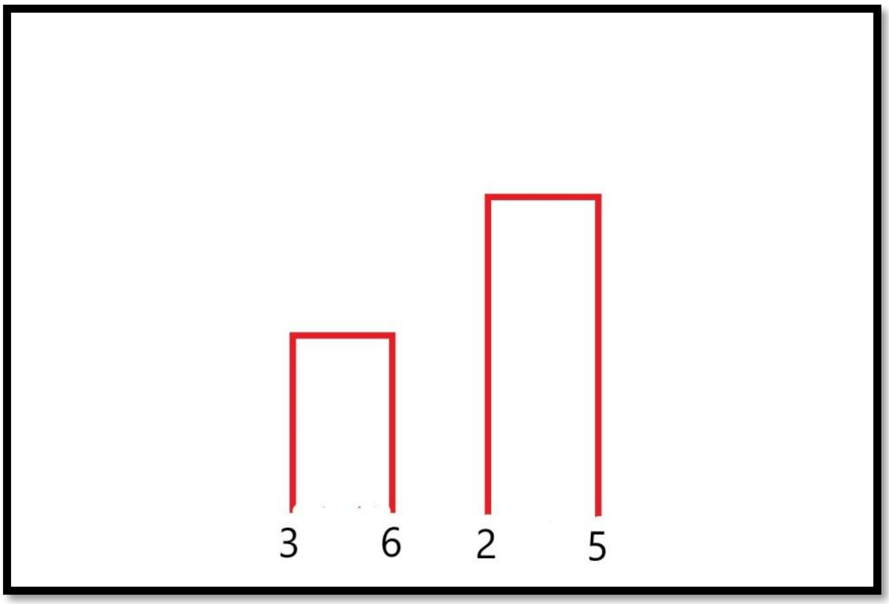


New distance matrix

	P1	P2	P3,P6	P4	P5
P1	0				
P2	0.23	0			
P3,P6	0.22	0.15	0		
P4	0.37	0.20	0.15	0	
P5	0.34	0.14	0.28	0.29	0

	P1	P2	P3,P6	P4	P5
P1	0				
P2	0.23	0			
P3,P6	0.22	0.15	0		
P4	0.37	0.20	0.15	0	
P5	0.34	0.14	0.28	0.29	0





And the process keeps on going

To update the distance matrix

MIN [dist (P2 , P5) , point]

= MIN [dist (P2,point) , (P5,point)]

Applications: -

1. **Charting Evolution through Phylogenetic Trees**

Answering the questions How can we relate different species together, Are giant pandas closer to bears or racoons? Nowadays, we can use DNA sequencing and hierarchical clustering to find the phylogenetic tree of animal evolution:

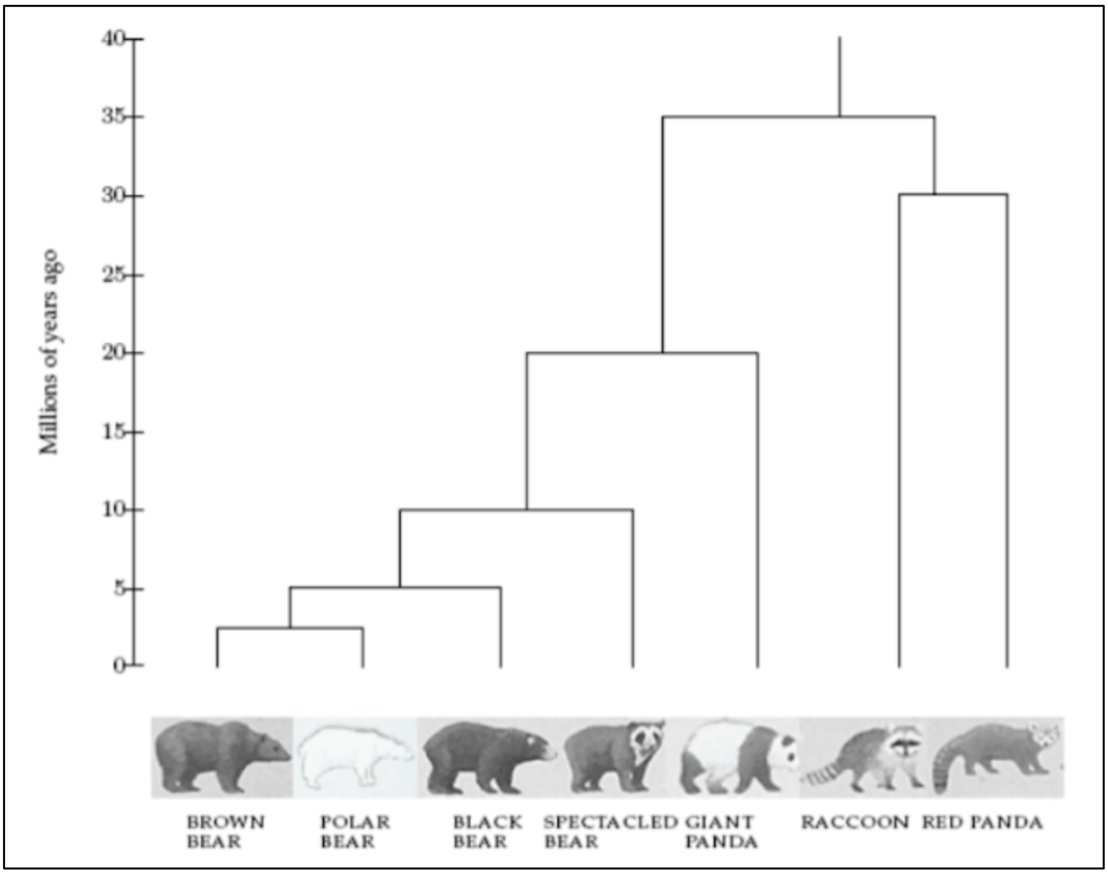
Generate the DNA sequences

Calculate the edit distance between all sequences.

Calculate the DNA similarities based on the edit distances.

Construct the phylogenetic tree.

As a result of this experiment, the researchers were able to place the giant pandas closer to bears.



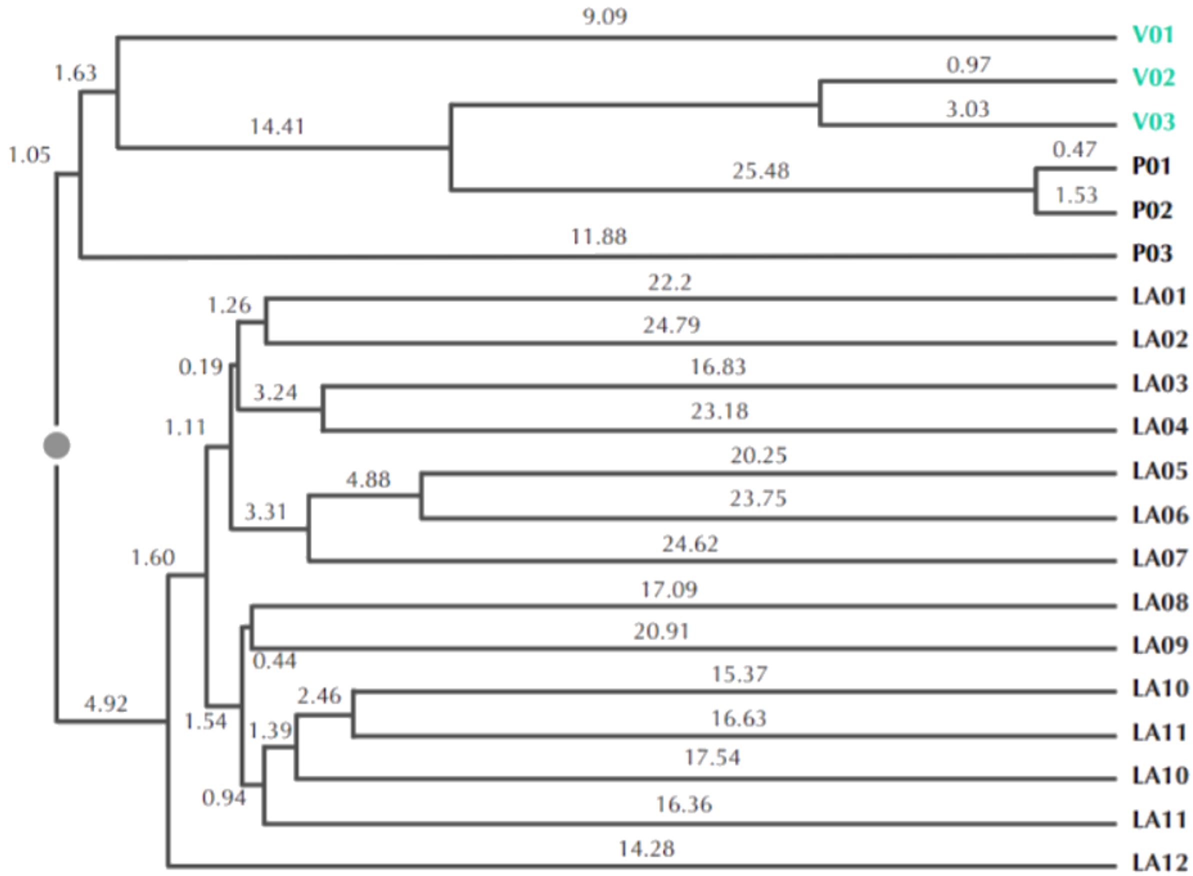
2. **Tracking Viruses through Phylogenetic Tree**

Can we find where a viral outbreak originated?

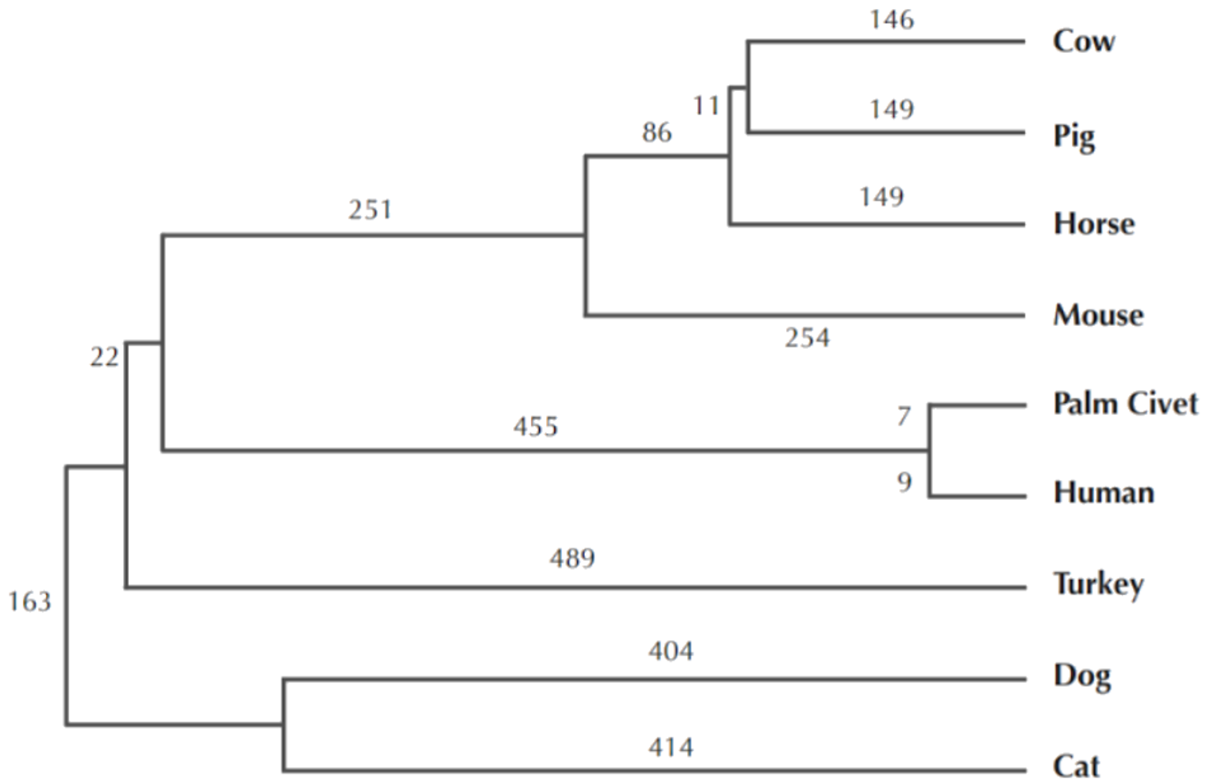
Tracing these outbreaks to their source can give scientists additional data as to why and how the outbreak began, potentially saving lives.

Viruses such as HIV have high mutation rates, which means the similarity of the DNA sequence of the same virus depends on the time since it was transmitted. This can be used to trace paths of transmission.

This method was used as evidence in a court case, wherein the victim's strand of HIV was found to be more similar to the accused patient's strand, compared to a control group.



A similar study was also done for finding the animal that gave the humans the SARS virus:


146
Cow
11
149
Pig
86
149
Horse
251
Mouse
254
22
Palm Civet
7
455
9
Human
489
Turkey
163
404
Dog
414
Cat

Pros and cons: -

Pros-

Easy to understand and implement and gives best results in some cases

No prior information about the number of clusters required

Produces an order of objects, which may be informative for display and better visualization.

Cons-

Dendrogram is commonly misinterpreted

It rarely provides the best solution

Poorly works with mixed data types

Does not work with missing data

Does not work well on very large data sets

No objective function is directly minimized

Once a decision is made to combine two clusters, it cannot be undone

May not give best results in all cases

Density Based Clustering: -

Density-Based Clustering identifies distinctive groups/clusters in the data, based on the idea that a cluster in a data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density. Density-Based Clustering identifies distinctive groups/clusters in the data, based on the idea that a cluster in a data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density.

The DBSCAN algorithm uses two parameters:

MINPTS: It is the minimum number of points (a threshold) clustered together, for a region to be considered dense.

EPS (ε): The distance that specifies the neighbourhoods. Two points are considered to be neighbours if the distance between them is less than or equal to eps.

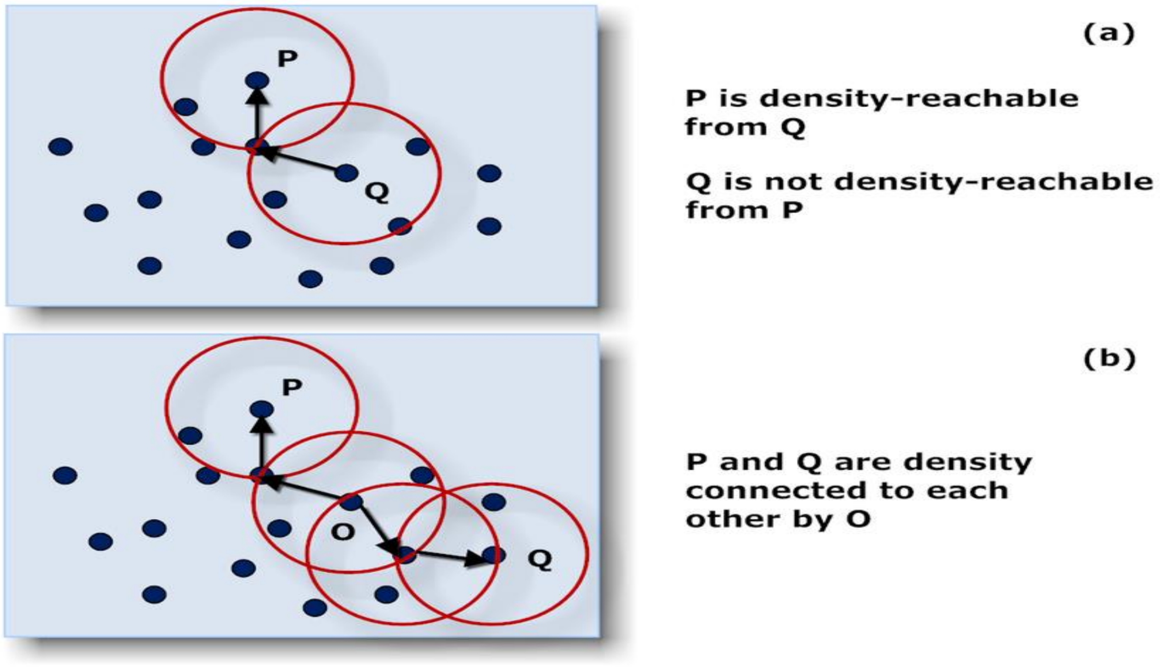
These parameters can be understood if we explore two concepts called:

DENSITY REACHABILITY:

A point "p" is said to be density reachable from a point "q" if point "p" is within ε distance from point "q" and "q" has a sufficient number of points in its neighborhood which are within distance ε.

DENSITY CONNECTIVITY:

A point "p" and "q" are said to be density connected if there exists a point "o" which has a sufficient number of points in its neighbors and both the points "p" and "q" are within the ε distance from "o".



(a)

P is density-reachable from Q

Q is not density-reachable from P

(b)

P and Q are density connected to each other by O

This is a chaining process. So, if "q" is neighbor of "o", "o" is neighbor of "s", "s" is a neighbor of "t" which in turn is neighbor of "p" implies that "q" is neighbor of "p".

There are three types of points after the DBSCAN clustering is complete:

CORE POINT: A data point is considered to be a core point if it has a minimum number of neighbouring data points (min_pts) at an epsilon distance from it. (These min_pts include the original data points also.)

BORDER POINT: A data point that has less than the minimum number of neighboring data points needed but has at least one core point in the neighborhood.

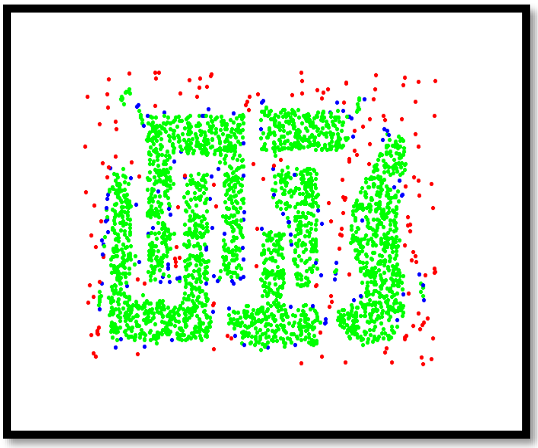
NOISE POINT: A data point that is not a core point or a border point is considered noise or an outlier.

DBSCAN DATASET

CORE

BORDER

NOISE



THE DBSCAN ALGORITHM IS AS FOLLOWS:

- Randomly select a point p.

  Retrieve all the points that are density reachable from p with regard to the Maximum radius of the neighborhood (EPS) and the minimum number of points within the eps neighborhood (Min Pts).

- If the number of points in the neighborhood is more than Min Pts then p is a core point. We assign a new cluster for the core point p.

- Find all its density connected points and assign them to the same cluster as the core point. If p is not a core point, then mark it as a noise/outlier and move to the next point.

- Continue the process until all the points have been processed and visited.

R DEMO:

```
# Loading data
data(iris)

# Structure
str(iris)

# Installing Packages
install.packages("fpc")

# Loading package
library(fpc)

# Remove label form dataset
iris_1 <- iris[-5]

# Fitting DBScan clustering Model
# to training dataset
set.seed(220)  # Setting seed
Dbscan_cl <- dbscan(iris_1, eps = 0.45, MinPts = 5)
Dbscan_cl

# Checking cluster
Dbscan_cl$cluster

# Table
```

```
table(Dbscan_cl$cluster, iris$Species)


# Plotting Cluster

plot(Dbscan_cl, iris_1, main = "DBScan")

plot(Dbscan_cl, iris_1$Petal.Width, iris_1$Sepal.Length, main = "Petal Width vs Sepal Length")
```

Applications of dbscan clustering:

1. Images of satellite
2. Crystallography of x-ray
3. Anomaly detection in temperature data

Pros and Cons:

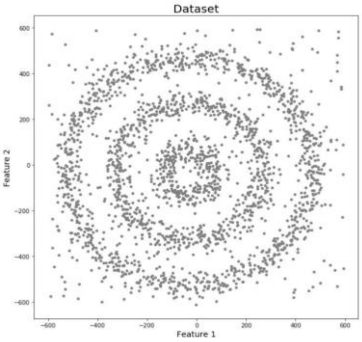
Pros:

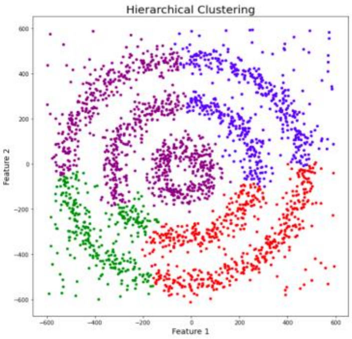
- Does not require one to specify the number of clusters in the data a priori, as opposed to k-means.
- Can find arbitrarily or non-linear shaped clusters.
- Can even find a cluster completely surrounded by (but not connected to) a different cluster.
- Requires just two parameters and is mostly insensitive to the ordering of the points in the database.
- Has a notion of noise, and is robust to outliers.

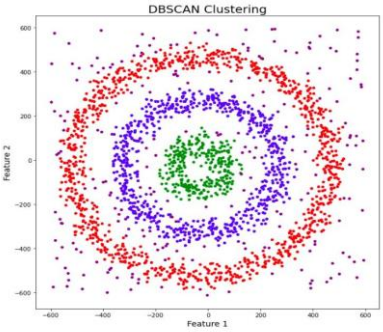
Cons:

- Fails in case of high varying density clusters. Cannot cluster data-sets with large differences in densities well.
- Hence the minPts- eps combination cannot be chosen appropriately for all clusters.
- Border points that are reachable from more than one cluster can be part of either cluster, depending on the order in which the data are processed.
- Choosing a meaningful eps value can be difficult if the data isn't well understood.
- DBSCAN is not entirely deterministic.

COMPARISON







The DBSCAN Clustering has identified the outliers which is in purple color unlike Hierarchical clustering which has failed to identify the outliers.

Hierarchical clustering can put together clusters that seem close, but no information about other points is considered. Density-based methods only look at a small neighborhood of nearby points and similarly fail to consider the full dataset.

Links-

https://www.displayr.com/strengths-weaknesses-hierarchical-clustering/

https://www.youtube.com/watch?v=9U4h6pZw6f8&t=2213s

https://www.youtube.com/watch?v=RdT7bhm1M3E