Unsupervised Learning Handbook

Introduction to Unsupervised Learning

The goal of unsupervised learning is to find hidden patterns in unlabeled data. As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

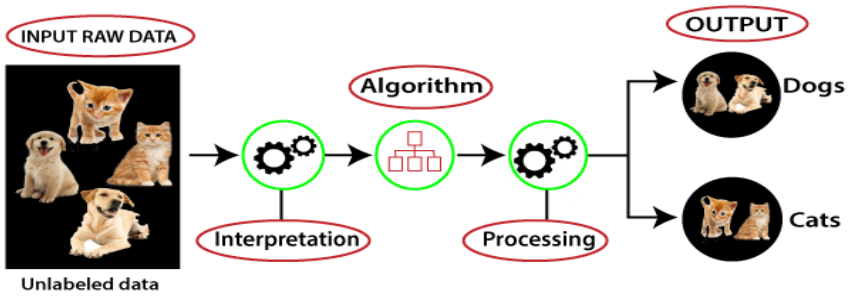


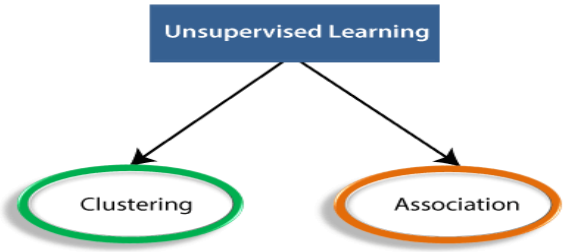
Image Source: Unsupervised Learning

Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and differences between the objects.

Types of Unsupervised Learning Algorithm:

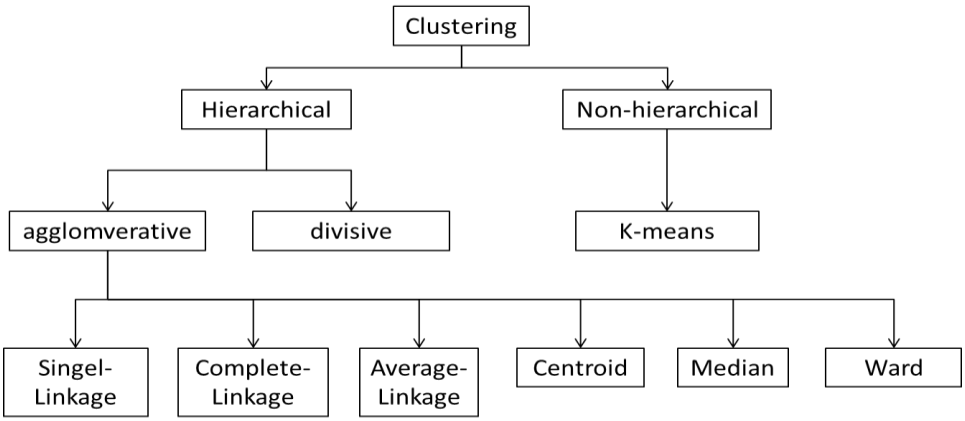
The unsupervised learning algorithm can be further categorized into two types of problems:



- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remain into a group and have less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occur together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis. For more details refer to the Recommendation System Handbook.

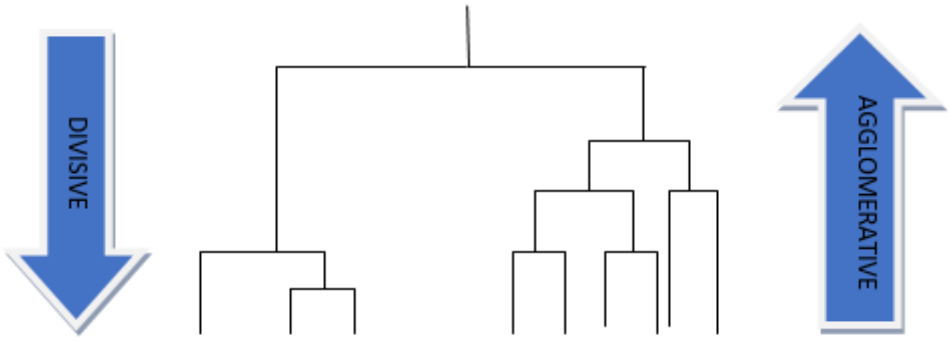
Clustering

Clustering is mainly divided as the below diagram



Hierarchical Clustering

Hierarchical Clustering analysis is an algorithm used to group the data points with similar properties. These groups are termed as clusters. As a result of hierarchical clustering, we get a set of clusters where these clusters are different from each other. Hierarchical Clustering is further classified as Agglomerative Clustering (involving decomposition of cluster using bottom-up strategy) and Divisive Clustering (involving decomposition of cluster using top-down strategy)



The above figure shows Agglomerative vs Divisive clustering

This clustering technique uses distance as a measure of similarity and hence observations which have smaller distance are considered as similar and vice-versa.

These are the main distance measures which are used:

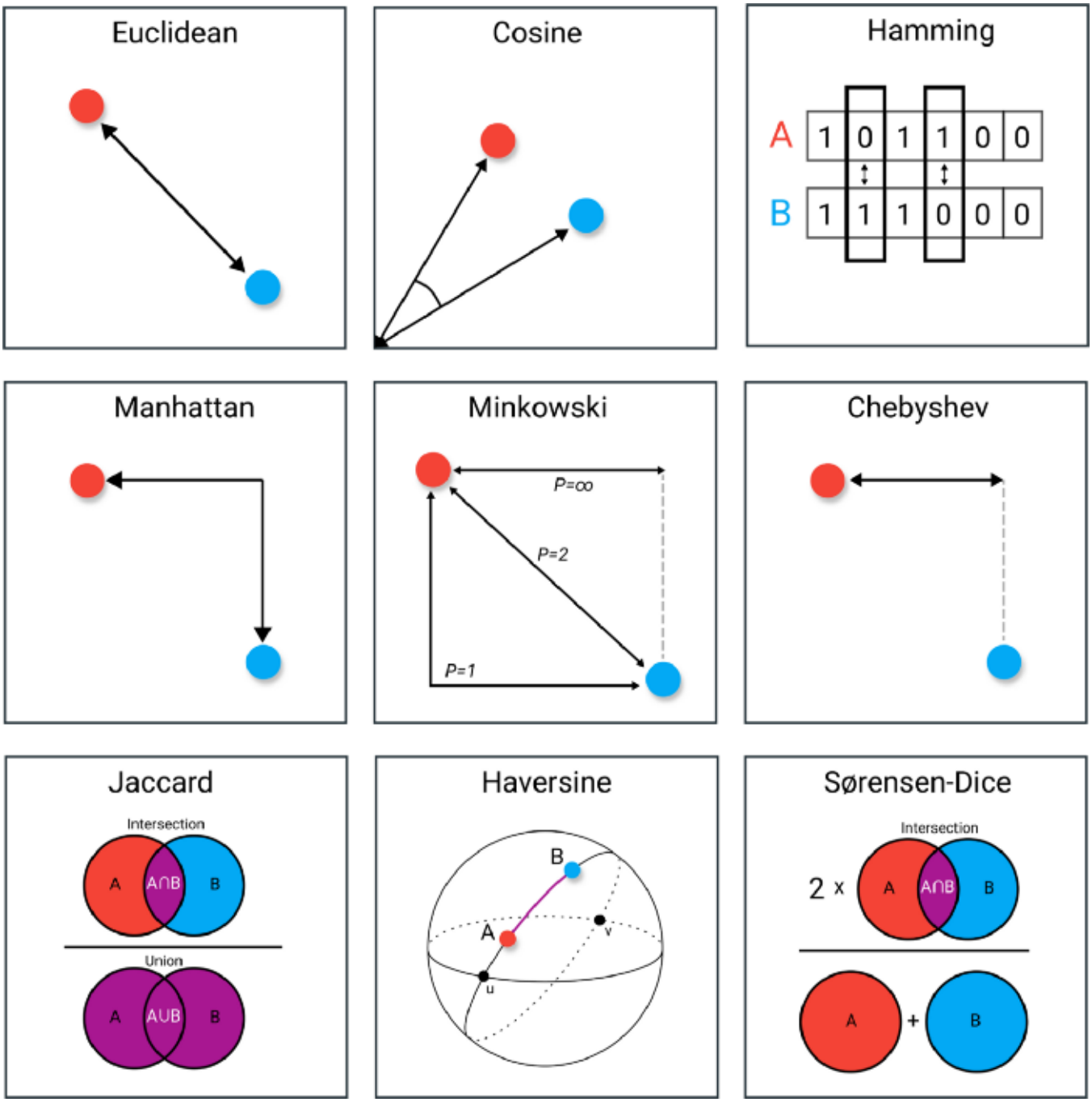


Image source: Distance measures

Let's understand 3 main types of distance measures:

- **Euclidean distance:** Euclidean distance is used when calculating the distance between two rows of data that have numerical values, such a floating point or integer values. Most instance-based learners use Euclidean distance.

$$D(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2)}$$

- **Manhattan distance:** Manhattan distance also called the Taxicab distance or the City Block distance, calculates the distance between two real-valued vectors. It is perhaps more useful to vectors that describe objects on a uniform grid, like a chessboard or city blocks. The taxicab name for the measure refers to the intuition for what the measure calculates: the shortest path that a taxicab would take between city blocks (coordinates on the grid).

$$D(x, y) = \sum_{i=1}^{k} |x_i - y_i|$$

- **Minkowski distance:** Minkowski calculates the distance between two real-valued vectors. It is a generalization of the Euclidean and Manhattan distance measures and adds a parameter, called the "order" or "p ", that allows different distance measures to be calculated.
  When p is set to 1, the calculation is the same as the Manhattan distance. When p is set to 2, it is the same as the Euclidean distance.

$$D(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

source: Distance Measures

Hierarchical clustering further divided into two groups:

- **Agglomerative Clustering:** The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, named *dendrogram*.

- **Divisive Clustering:** The inverse of agglomerative clustering is *divisive clustering* it works in a "top-down" strategy. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster (see figure below).

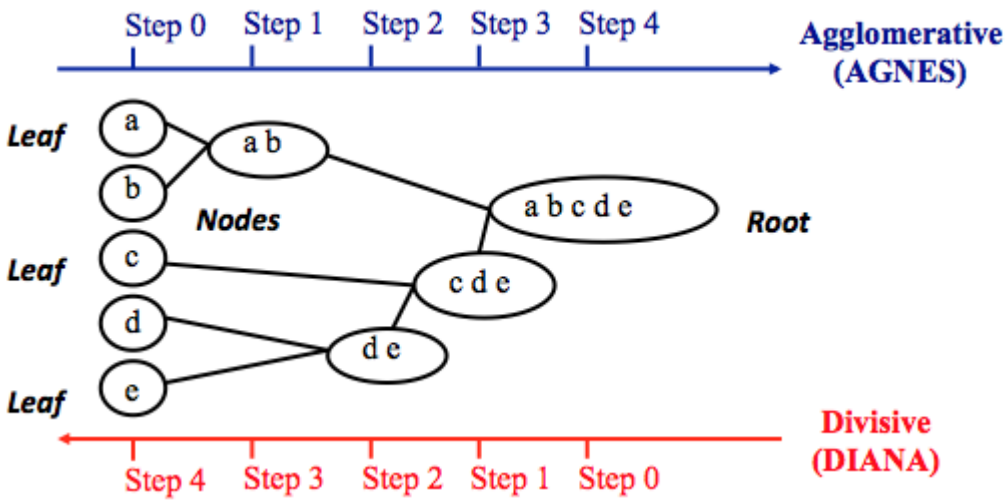


Image source: Hierarchical Clustering

Note: Agglomerative clustering is good at identifying small clusters. Divisive clustering is good at identifying large clusters.

There are difference ways to measure the distance between two clusters

There are several ways to measure the distance between in order to decide the rules for clustering, and they are often called Linkage Methods.

Some of the popular linkage methods are:

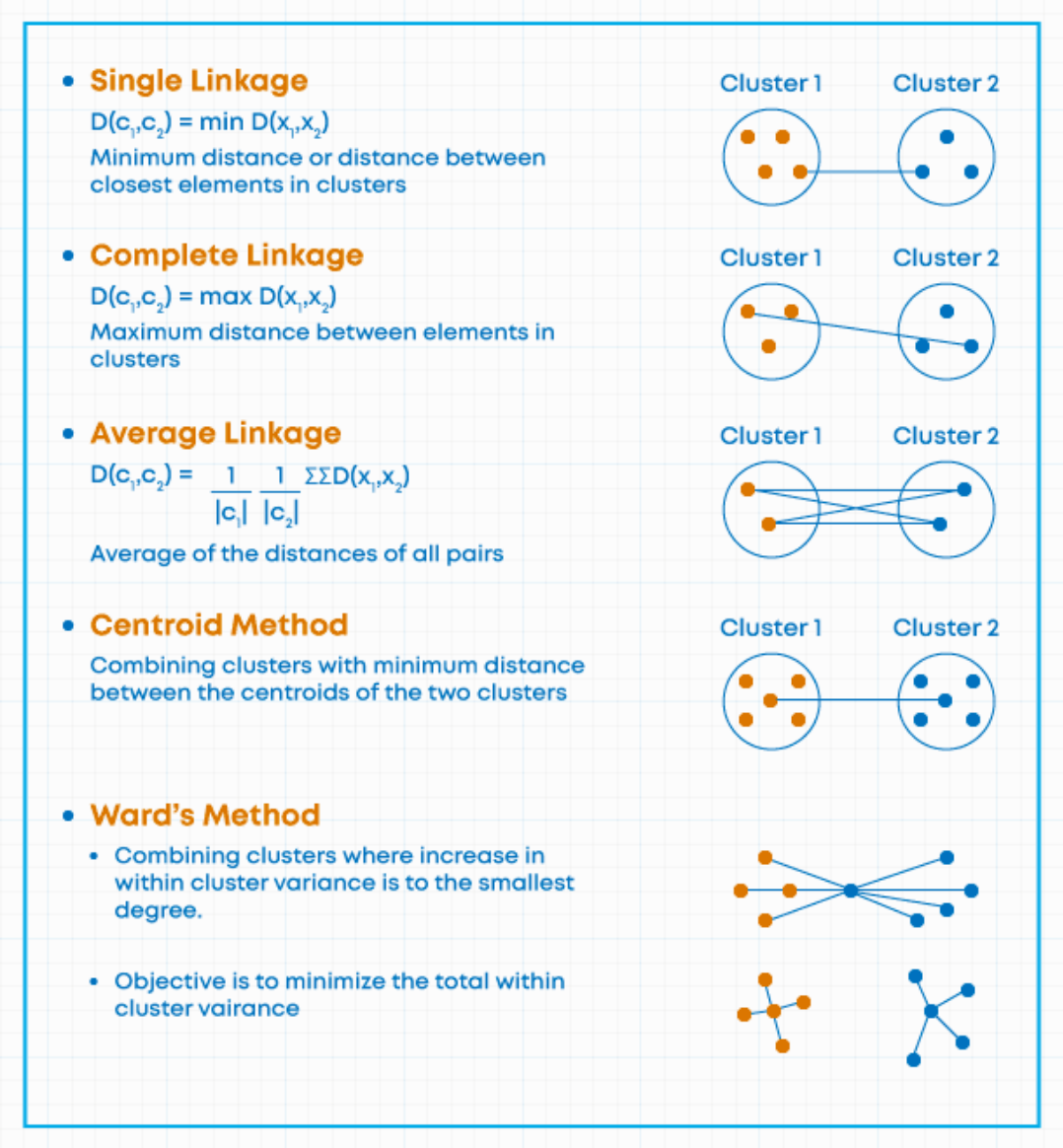
- Simple Linkage
- Complete Linkage
- Average Linkage
- Centroid Linkage
- Ward's Linkage

Some of the other linkage methods are:

- Strong Linkage
- Flexible linkage
- Simple Average

The Linkage method's choice depends on you, and you can apply any of them according to the type of problem, and different linkage methods lead to different clusters.

Below is the comparison image, which shows all the linkage methods.



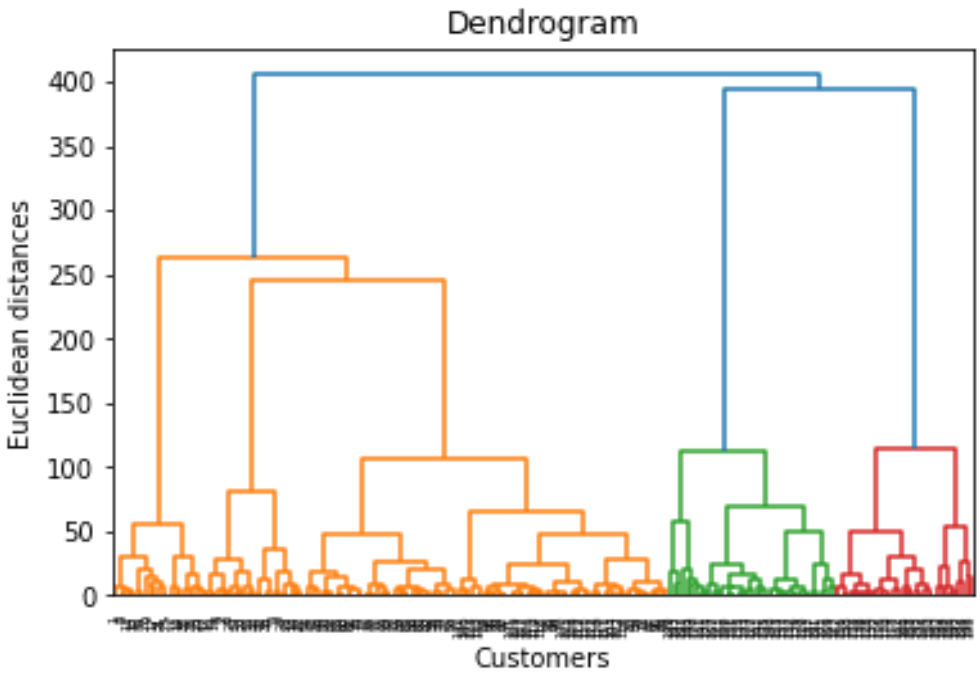
Source: Linkages

What is Dendrogram

A Dendrogram is a diagram that represents the hierarchical relationship between objects. The Dendrogram is used to display the distance between each pair of sequentially merged objects.

These are commonly used in studying hierarchical clusters before deciding the number of clusters significant to the dataset.

The distance at which the two clusters combine is referred to as the dendrogram distance. The primary use of a dendrogram is to work out the best way to allocate objects to clusters.



The key point to interpreting or implementing a dendrogram is to focus on the closest objects in the dataset.

Applications of Hierarchical clustering:

Hierarchical clustering can be used in various fields of applied sciences and scientific research such as:

1. Bioinformatics — In this field, clustering DNA sequences and classifying them with similar biological characteristics and study the relationships among them
2. Healthcare / Medical research — In this field, clustering helps to classify the patients based on their symptoms, pathological tests scores, CT Scans, MRI scans, X-ray reports, etc. and study the application of specific lines of treatment for various clusters and standardize the treatment procedures for different clusters.
3. Internet search engines — Here clustering techniques can be used to group the search results based on the similarity of the results and hence can drastically improve the performance of the search.
4. Image classification — using the extracted features from the images, clustering can be useful to classify the images and group the similar images in the clusters which can represent a specific class or species
5. Judicial Practices — using the clustering techniques, it can help the judges to take decisions on the basis of clustering of the cases on the basis of their similarity.

Non-Hierarchical Clustering

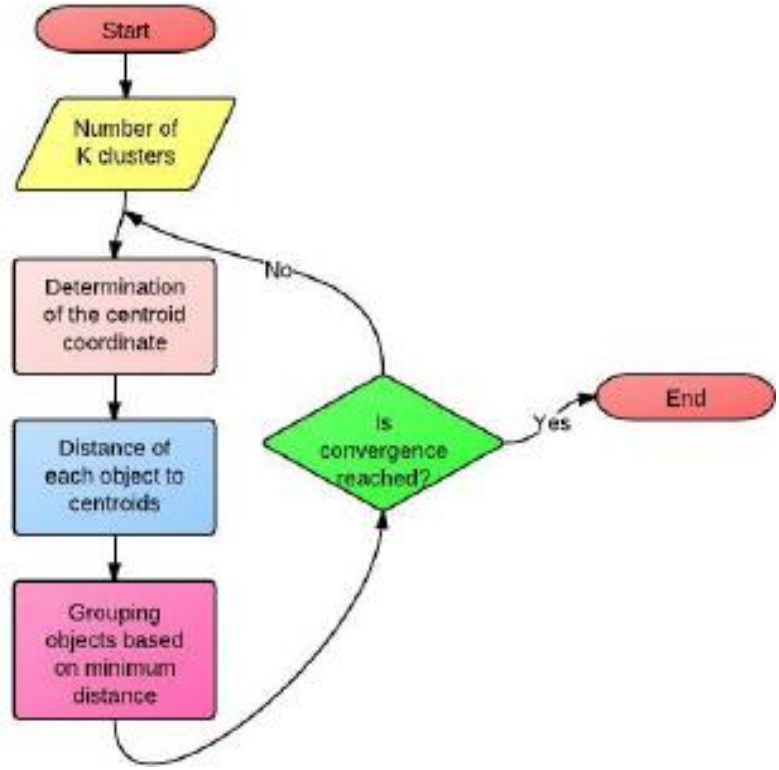
Non-Hierarchical Clustering involves formation of new clusters by merging or splitting the clusters instead of following a hierarchical order.

This technique groups the data in order to maximize or minimize some evaluation criteria. K means clustering is an effective way of non-hierarchical clustering. In this method the partitions are made such that non-overlapping groups have no hierarchical relationships between themselves.

K-Means Clustering

It is a method that calculates the Euclidean distance to split observations into k clusters in which each observation is attributed to the cluster with the nearest mean (cluster centroid).

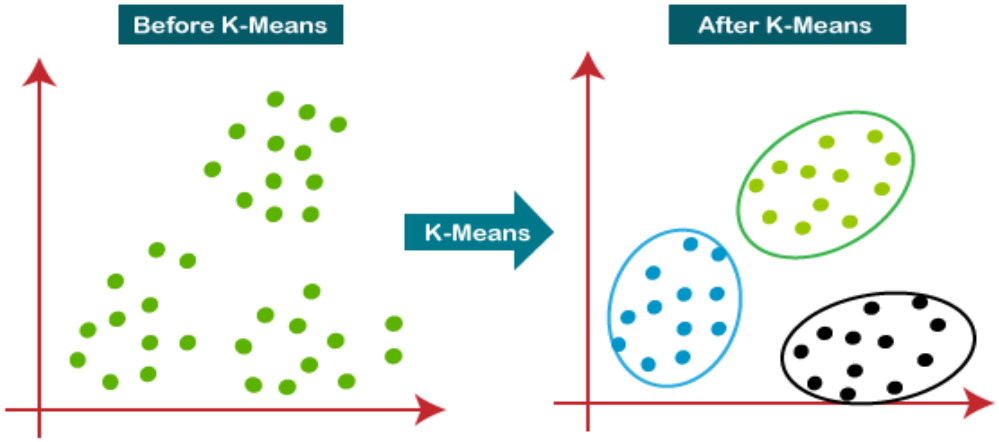
Flowchart to understand how K-Means algorithm works



Source: K-means-algorithm

How does the K-Means cluster algorithm work?

1. Select the number of clusters (K)
2. Randomly select a number of data points that matches the number of clusters
3. Measure the distances between each point to its initial cluster
4. Assign each datapoint to its nearest initial cluster
5. Repeat the calculations for each point
6. Calculate the mean of each cluster
7. Assign the mean as the new cluster centroid
8. Measure each point to the new cluster centroid
9. Redefine clusters and assign the new mean as the next cluster centroid
10. Repeat process until convergence
11. Select the parameters with the smallest SSE



source: K Means Clustering

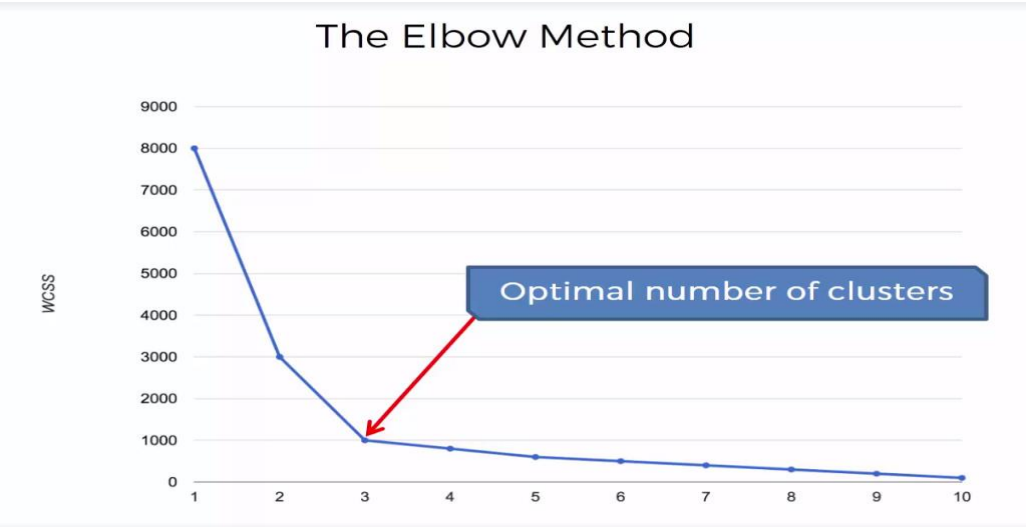
How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters. Mainly used techniques are:

1. Elbow Method
2. Silhouette Method

Elbow Method:

In Elbow method we Calculate the **Within-Cluster-Sum of Squared** Errors (WSS) for **different values of k**, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an **elbow.**



Source: Elbow Method

The curve looks like an elbow. In the above plot, the elbow is at k=3 (i.e. Sum of squared distances falls suddenly) indicating the optimal k for this dataset is 3.

Silhouette Method

The silhouette value measures how similar a point is to its own cluster (intra cluster distance) compared to other clusters (inter cluster distance). Its value ranges from -1 to 1.

Silhouette Score = (b-a)/max(a,b)

where

a= average intra-cluster distance i.e the average distance between each point within a cluster.

b= average inter-cluster distance i.e the average distance between all clusters.

A score of 1 denotes that the data point is very compact within the cluster to which it belongs and far away from the other clusters, very good value. The worst value is -1. Values near 0 denote overlapping clusters.

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i) \end{cases}$$

Source: Silhouette-Coefficient

Note: In k-means clustering, the number of clusters that you want to divide your data points into i.e., the value of K has to be pre-determined whereas in Hierarchical clustering data is automatically formed into a tree shape form (dendrogram).

But beware: *k-means* uses numerical distances, so it could consider close two really distant objects that merely have been assigned two close numbers.

The solution lies in the *k-modes* algorithm. KModes clustering is one of the unsupervised Machine Learning algorithms that is used to cluster categorical variables.

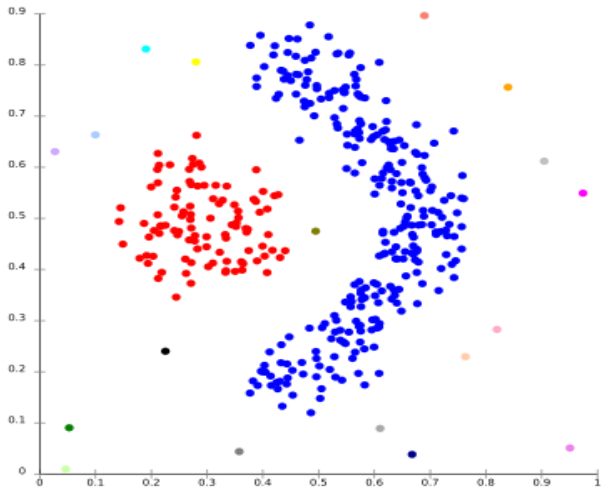
k-modes is an extension of *k-means*. Instead of distances it uses *dissimilarities* (that is, quantification of the total mismatches between two objects: the smaller this number, the more similar the two objects). And instead of means, it uses *modes.* A mode is a vector of elements that minimizes the dissimilarities between the vector itself and each object of the data. We will have as many modes as the number of clusters we required, since they act as centroids.

For numerical *and* categorical data, another extension of these algorithms exists, basically combining *k-means* and *k-modes*. It is called *k-prototypes*.

Density-Based Methods:

These methods consider the clusters as the dense region having some similarities and differences from the lower dense region of the space. These methods have good accuracy and the ability to merge two clusters.

Example *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)*, *OPTICS (Ordering Points to Identify Clustering Structure)*, etc.



Source: Density Based Clustering

Necessary Libraries

In [ ]:
```
# For data cleaning and analysis
import pandas as pd
# For some basic mathematical operations
import numpy as np
```

In [ ]:
```
# For visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [ ]:
```
# Scale the data
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
# Hierarchical Clustering
from sklearn.cluster import AgglomerativeClustering
# For finding cophenet correlation, dendrogram and linkage matrix
from scipy.cluster.hierarchy import cophenet, dendrogram, linkage
#Pairwise distribution between data points
from scipy.spatial.distance import pdist
# fcluster function to cluster the data
from scipy.cluster.hierarchy import fcluster
```

In [ ]:
```
# K-Means Clustering
from sklearn.cluster import KMeans
```

Principal Component Analysis (PCA)

Principal component analysis (PCA) is the process of computing the principal components. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible.

In data analysis, the first principal component of a set of p variables are the derived variable formed as a linear combination of the original variables that explains the most variance. The second principal component explains the most variance in what is left once the effect of the first component is removed, and we may proceed through p iterations until all the variance is explained. PCA is most commonly used when many of the variables are highly correlated with each other and it is desirable to reduce their number to an independent set.

Principal component analysis can be broken down into five steps

1. Standardize the continuous variables
2. Compute the covariance matrix to identify correlations
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
4. Create a feature vector to decide which principal components to keep
5. Recast the data along the principal components axes or Reduce data dimensions.

Mechanics of Principal Component Analysis:

1. Begins by standardizing the data. Data on all the dimensions are subtracted from their means and divided by their standard deviation to shift all the data points to the origin.
2. Generate the covariance matrix/correlation matrix for all the dimensions.
    - Matrix is two dimensional.
    - Correlation on the other hand measures both the strength and direction of the linear relationship between two variables. Correlation is a function of Covariance.
    - Covariance is when two variables vary with each other, whereas Correlation is when the change in one variable results in the change of another variable.
    - Variance and Covariance: is measured with the dimensions and Covariance is among the dimensions

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

where n is the number of samples (e.g. the number of people) and $\bar{x}$ is the mean of the random variable x (represented as a vector). The covariance $\sigma(x, y)$ of two random variables x and y is given by

$$\sigma(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

with n samples. The variance σ_x^2 of a random variable x can be also expressed as the covariance with itself by $\sigma(x, x)$.

source: covariance-matrix

- Let's consider a two-dimensional case, Covariance matrix for two dimensions is given by.

$$C = \begin{pmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{pmatrix}$$

σ(x,x) = variance of (x) ← individual dimensions

σ(x,y) = Covariance of (y,x) ← 2 dimensions

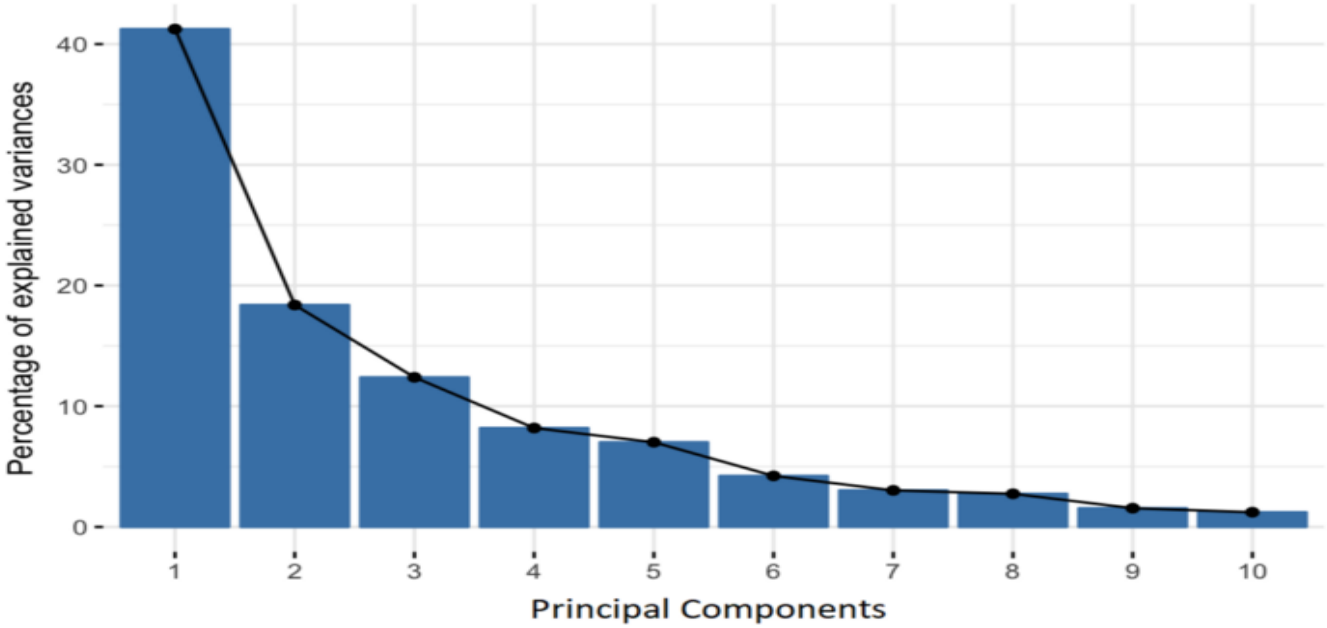
- **What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?**
  It's actually the sign of the covariance that matters :
  - if positive then : the two variables increase or decrease together (correlated)
  - if negative then : One increases when the other decreases (Inversely correlated)

3. Perform Eigen decomposition, that is compute Eigen vectors which are the principal components and the corresponding Eigen values which are the magnitudes of the variance captured.
   An eigenvector is a nonzero vector that changes at most by a scalar factor when that linear transformation is applied to it. The corresponding eigenvalue is the factor by which the eigenvector is scaled.

   Let A be a square matrix (in our case the covariance matrix), v a vector and λ a scalar that satisfies Av = λv, then λ is called eigenvalue associated with eigenvector v of A.

```
In [ ]: eig_vals, eig_vecs = np.linalg.eig(cov_matrix)
```

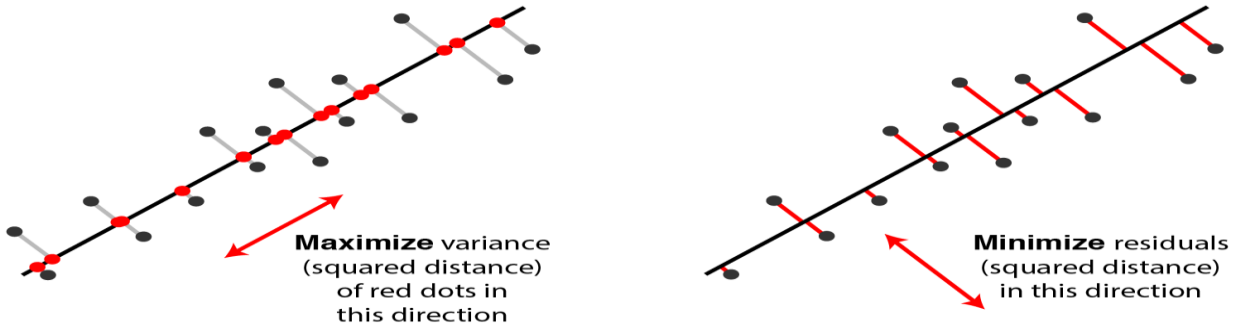
- Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the ***principal components*** of the data. Before getting to the explanation of these concepts, let's first understand what do we mean by principal components.
- Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until having something like shown in the scree plot below.



- Organizing information in principal components this way, will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

4. Eigenvectors of the Covariance matrix are actually the directions of the axes where there is the most variance (most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each Principal Component.

By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.



Source: PCA

5.The axes rotation is done such that the new dimensions capture max variance in the data points and also reduce total error(residuals) of representation.

Let's calculate from sklearn library

```
import numpy as np
import pandas as pd
A = np.matrix([[1,2,3,4],
               [5,5,6,7],
               [1,4,2,3],
               [5,3,2,1],
               [8,1,2,2]])

df = pd.DataFrame(A,columns  = ['f1','f2','f3','f4'])
df_std  = (df - df.mean()) / (df.std())
n_components = 2
from sklearn.decomposition import PCA
pca = PCA(n_components=n_components)
principalComponents = pca.fit_transform(df_std)
principalDf = pd.DataFrame(data=principalComponents,columns=['nf'+str(i+1) for i in range(n_components)])
print(principalDf)
```

```
        nf1       nf2
0 -0.014003  0.755975
1  2.556534 -0.780432
2  0.051480  1.253135
3 -1.014150  0.000239
4 -1.579861 -1.228917
```

Assumptions and limitations of PCA

1. PCA assumes a correlation between features.
2. PCA is sensitive to the scale of the features.
3. PCA is not robust against outliers.
4. The algorithm is not well suited to capturing non-linear relationships.

Advantages and Disadvantages of PCA

Advantages	Disadvantages
PCA reduces the curse of dimensionality. PCA lower the dimensions of the training dataset, which prevents the predictive algorithms from overfitting.	Even Though PCA covers maximum variance amid data features, sometimes it may skip a bit of information in comparison to the actual list of features.
Principal components are independent of each other, so removes correlated features.	Implementing PCA over datasets leads to transforming actual features in principal components that are linear combinations of actual features, therefore principle components are difficult to read or interpret as compared to actual features.
Reduction of noise since the maximum variation basis is chosen and so the small variations in the background are ignored automatically.	Data must be standardized before implementing PCA, else optimal PCA's can't be found.

********** HAPPY LEARNING **********