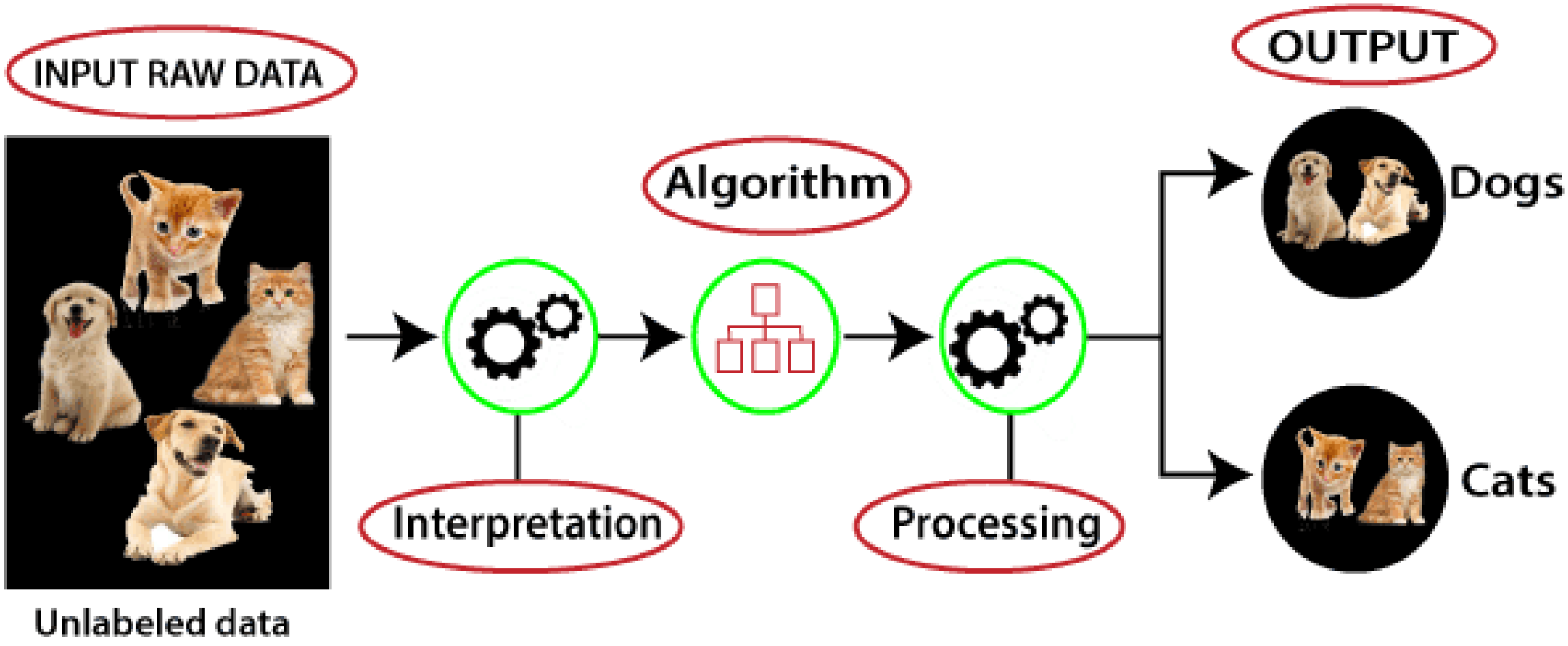
Machine Learning

Samatrix Consulting Pvt Ltd

Unsupervised Learning


INPUT RAW DATA
Unlabeled data
Interpretation
Algorithm
Processing
OUTPUT
Dogs
Cats

Machine Learning Algorithms *(sample)*

	Unsupervised	Supervised

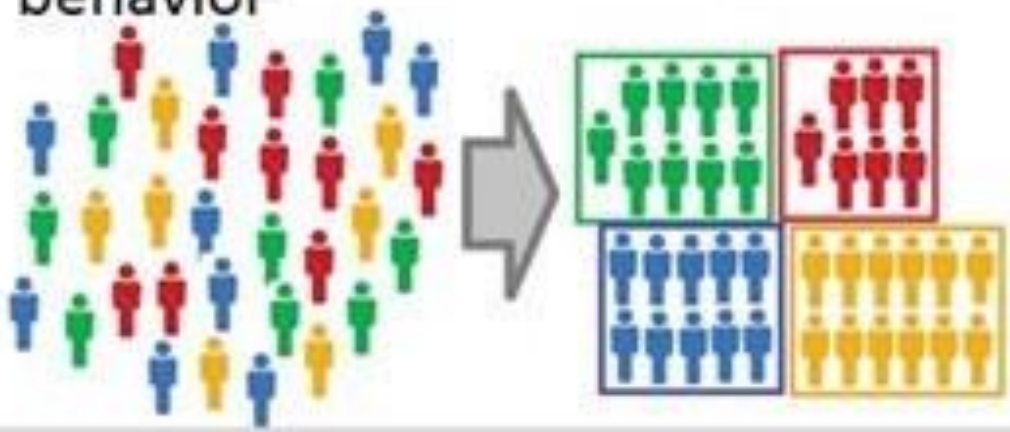
Continuous	• Clustering & Dimensionality Reduction
	  ○ SVD
	  ○ PCA
	  ○ K-means
		• Regression
		  ○ Linear
		  ○ Polynomial
		• Decision Trees
		• Random Forests

Categorical	• Association Analysis
	  ○ Apriori
	  ○ FP-Growth
	• Hidden Markov Model
		• Classification
		  ○ KNN
		  ○ Trees
		  ○ Logistic Regression
		  ○ Naive-Bayes
		  ○ SVM

Unsupervised Learning

Clustering

Grouping customers by purchasing behavior



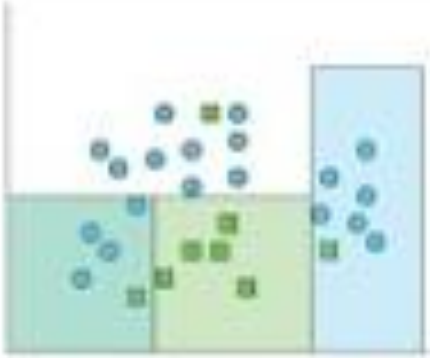
Association

People that buy X tend to buy Y

People that buy A+B tend to buy C

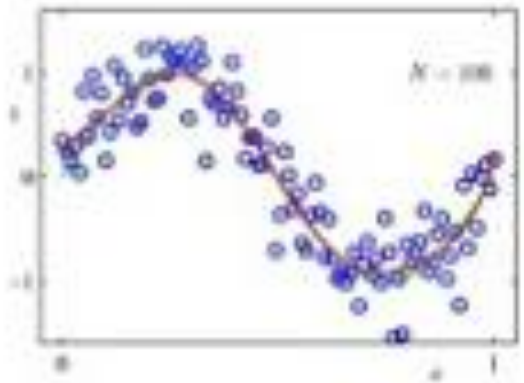
Supervised learning

Classification



Learns a method for predicting the instance class from pre-labeled (classified) instances

Regression



An attempt to predict a continuous attribute

Unsupervised Learning

Clustering

Finds "natural" grouping of instances given un-labeled data

Association Rules

Method for discovering interesting relations between variables in large DBs

Unsupervised Learning

- During this course, we have been studying about supervised learning methods such as regression and classification.
- In the supervised learning, we have a training data set which contains n observations and p features $X_1, X_2, \dots, X_p$.
- We also have an associated response Y for each of those n observations.
- Our goal is to predict Y using $X_1, X_2, \dots, X_p$

Unsupervised Learning

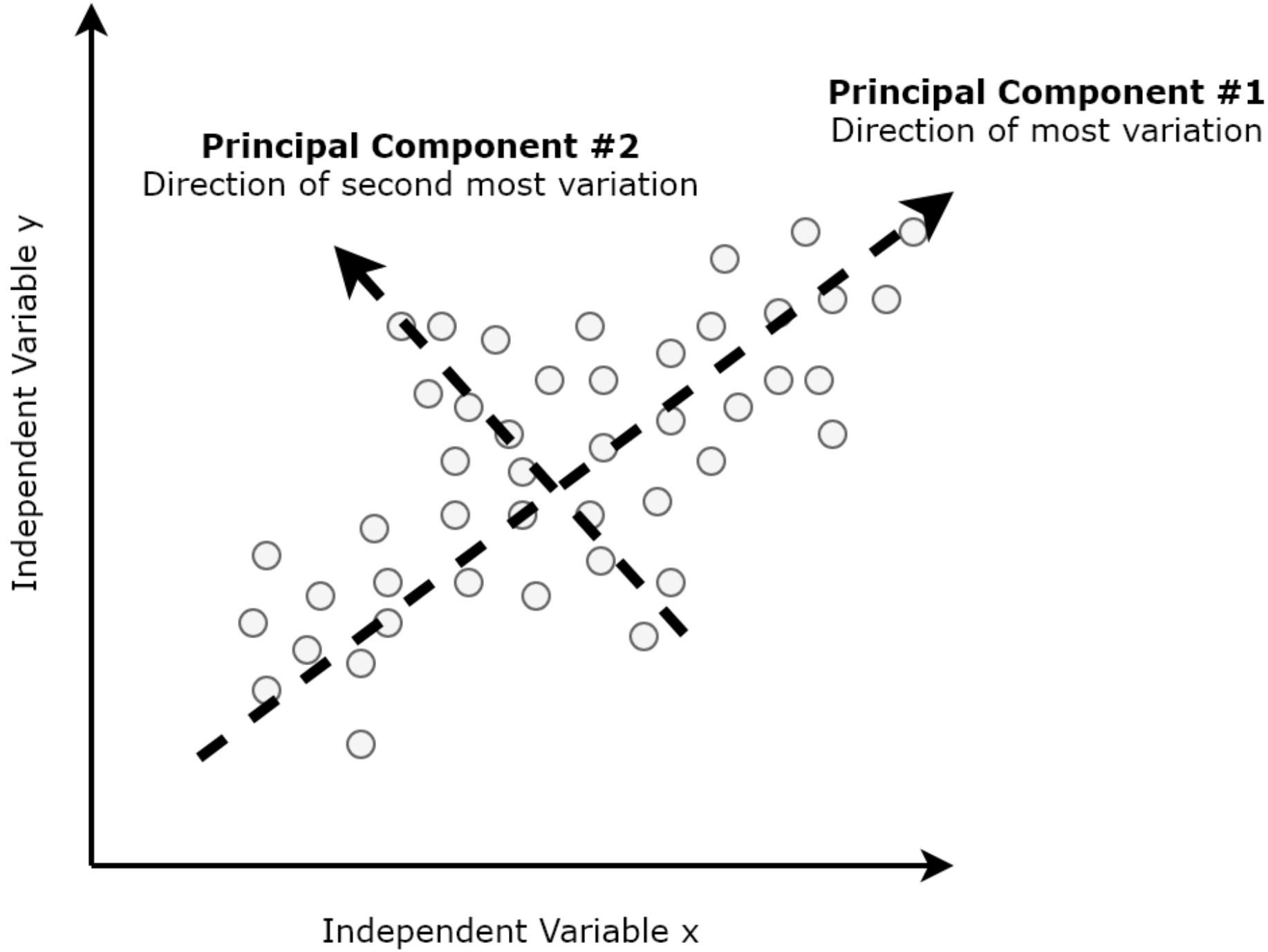
- In this chapter, we will focus on **unsupervised learning**.
- In this case, we only have a data set which contains n observations and p features $X_1, X_2, \ldots, X_p$.
- But we are not interested in predictions as we do not have any associated response variable Y.
- We are interested in discovering interesting facts about the features $X_1, X_2, \ldots, X_p$.
- In this chapter we will learn about principal components analysis and clustering.
- Principal component analysis is used for data visualization or data preprocessing. The clustering is used for discovering subgroups in the data.

Unsupervised Learning - Challenges

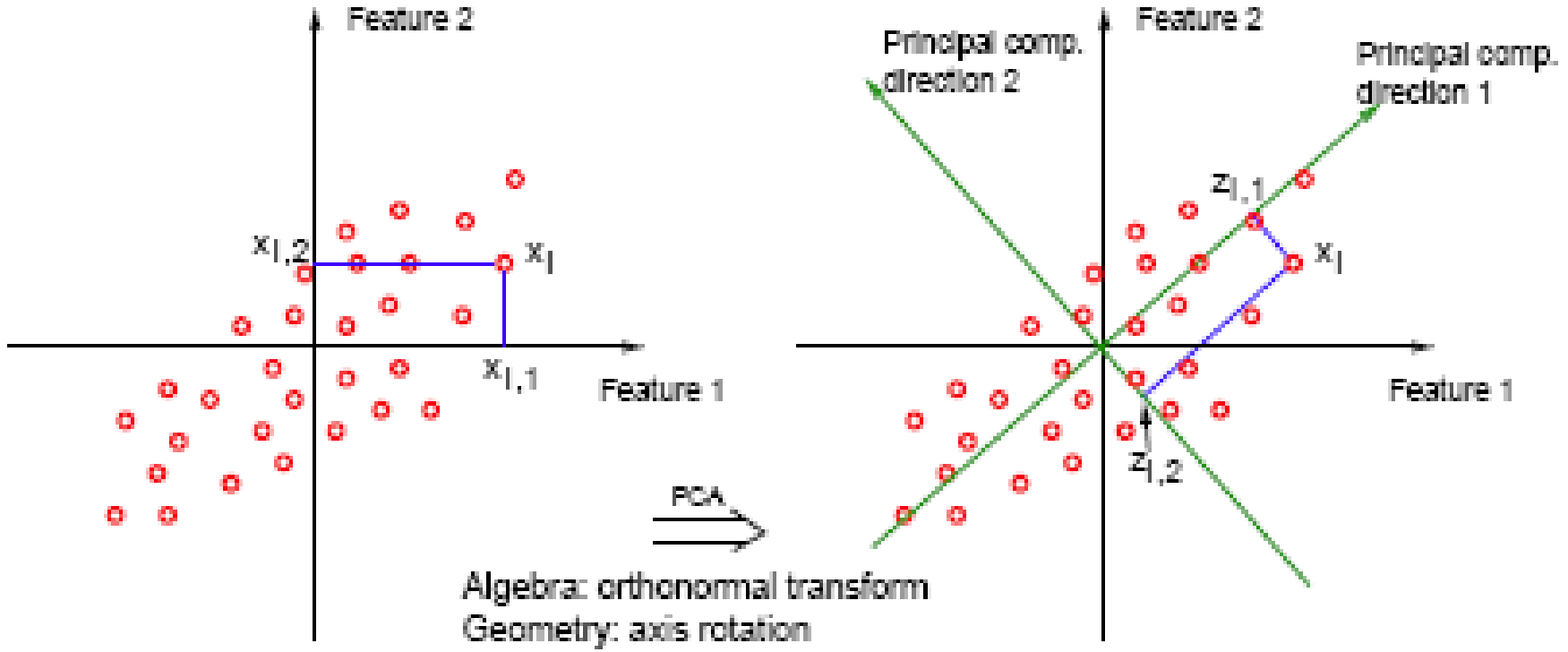
- Unsupervised learning is more challenging when compared to supervised learning.
- Generally, the goal of the unsupervised learning is not the prediction of a response.
- However, it is often use as a part of an exploratory data analysis.
- Furthermore, accessing the results that we get from unsupervised learning methods, is often hard because we do not have any universally accepted mechanism to perform the cross validation or validate the results on an independent data set.
- In the case of the supervised learning techniques, we can check the results by comparing the predicted response Y value with the observed response Y value by using the data that was not used in fitting the model.
- In the case of unsupervised learning, we do not have any mechanism to check our work because we do not have any response Y value.

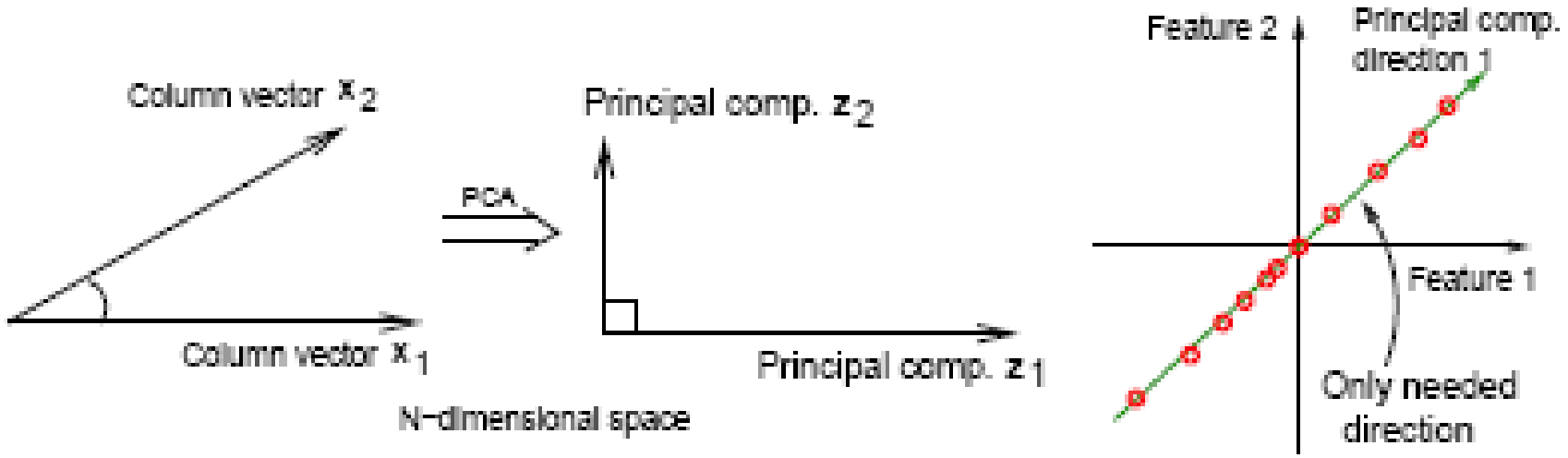
Unsupervised Learning – Use Cases

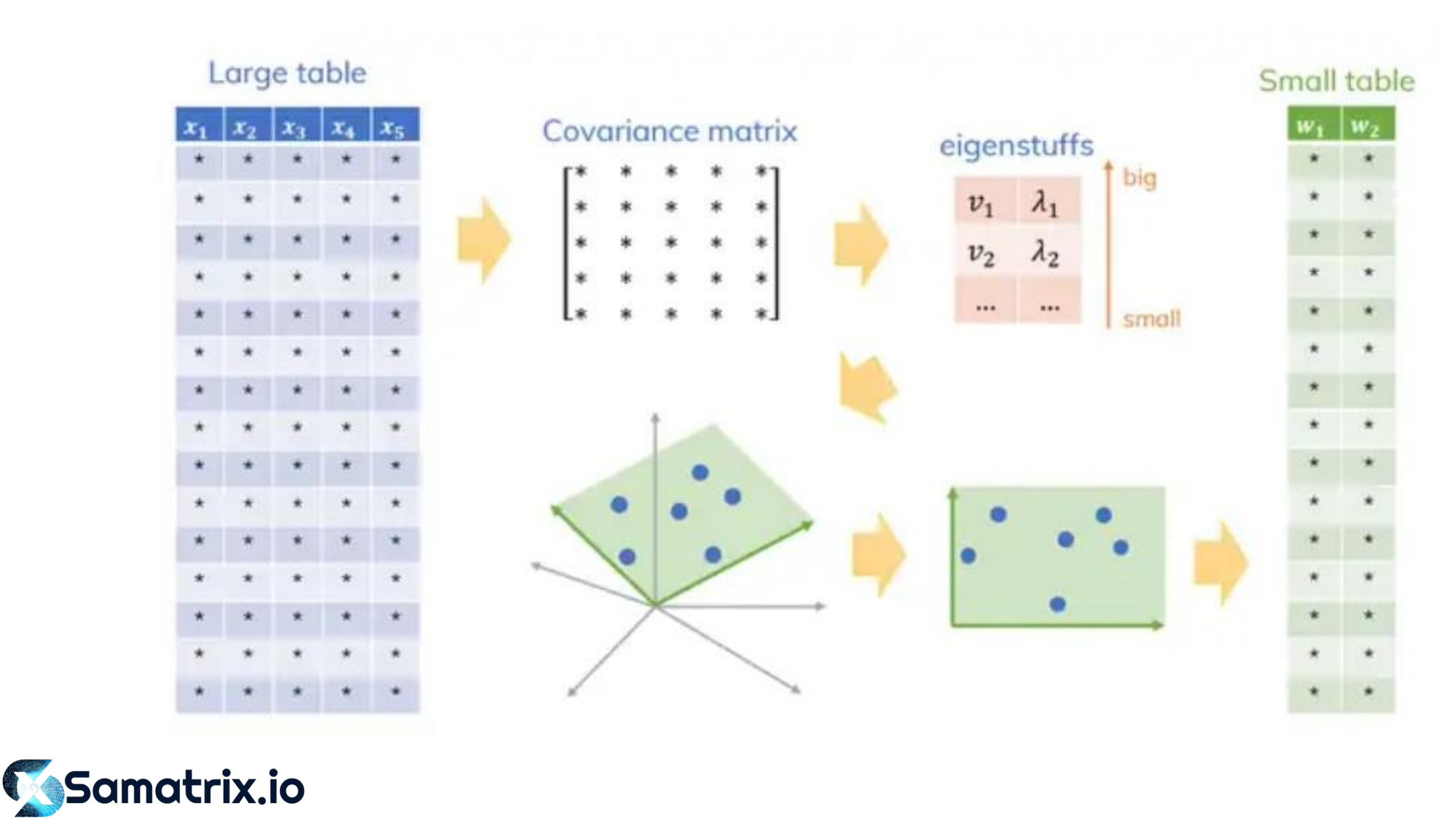
- Unsupervised learning techniques have been gaining importance in a number of fields.
- Online shopping sites have been using recommender system that identify groups of customers with similar browsing and purchase histories.
- It also identifies the items that are of particular interest to the shoppers within each group.
- Based on the purchase history the customers in a particular group, the recommender system can show the items to the individual customer.
- The search engine can show different search results to a particular person based on the click histories of other people who have similar search patterns.


Principal Component #1
Direction of most variation
Principal Component #2
Direction of second most variation
Independent Variable y
Independent Variable x

Samatrix.io


Feature 2
$x_{I,2}$
x_I
$x_{I,1}$
Feature 1
PCA
Principal comp. direction 2
Principal comp. direction 1
$z_{I,1}$
$z_{I,2}$
Algebra: orthonormal transform
Geometry: axis rotation


Column vector x_2
Column vector x_1
PCA
Principal comp. z_2
Principal comp. z_1
N-dimensional space
Feature 2
Principal comp. direction 1
Feature 1
Only needed direction


Large table
x_1 x_2 x_3 x_4 x_5
Covariance matrix
eigenstuffs
v_1 λ_1
v_2 λ_2
... ...
big
small
Small table
w_1 w_2

Principal Component Analysis

- We have already studied the principal component analysis in the context of principal component regression.
- When our data set contains a large number of correlated variables, we use principal components to summarize the data set by using a smaller number of representative variables that can explain the most variability in the original set collectively.
- The principal components are the directions in the feature space along with the variability in the data is high.

Principal Component Analysis

- The process by which we can compute the principal components and subsequently use them to understand the data is known as **Principal Component Analysis** (PCA).
- PCA is an unsupervised learning approach because we only have a set of features $X_1, X_2, \ldots, X_p$ and we do not have any associated response variable Y.
- We can also use PCA to visualization of the observations or visualization of the variables.

Principal Components

- PCA provides a mechanism to find a low-dimensional representation of a data set that contains the maximum possible variation.
- If our data set has p variables, it lives in p-dimensional space.
- All the variables are not equally important for machine learning models or data analysis problems.
- Such variables are dependent on other variables and do not add any new information to the process.
- Moreover, due to linear dependence, the regression matrix may become singular and we would not get a unique solution of the matrix.

Principal Components

- The objective of PCA is to find a smaller number of dimensions that are more useful.
- We can measure the usefulness by the amount that the observations vary along each dimension.
- Each of these dimensions are called principal components and they are linear combination of the p features.
- All the principal components are independent of each other and hence orthogonal to one another.

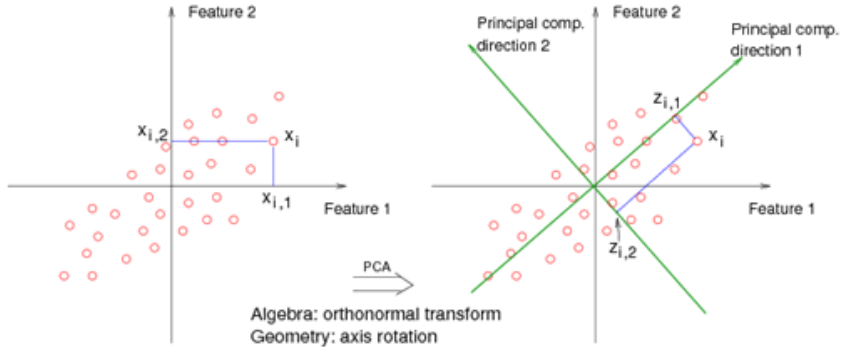
Principal Components

- In the linear algebraic terms, the PCA projects the data along the direction where the data varies the most.
- The first direction is represented by $\mathbf{v}_1$, which corresponds to the largest eigenvalue λ_1.
- The second direction is represented by $\mathbf{v}_2$, which corresponds to the second largest eigenvalue λ_2.

Principal Components

- The difference between linear dependence and linear independence has been illustrated geometrically in figure – 1.
- In the left-hand side panel, column vector $\mathbf{x}_1$ and $\mathbf{x}_2$ are linearly dependent. Due to which both the vectors are not orthogonal (angle $\neq 90^o$) to each other.
- We can also say that both the vectors are correlated with each other.
- The right-hand side panel shows two principal components $\mathbf{z}_1$ and $\mathbf{z}_2$ that are orthogonal (angle $=90^o$) to each other.
- Hence, they are linearly independent and uncorrelated (correlation $= 0$)

Principal Components



- The left-hand panel of figure – 2 illustrated several observations in red circle. These observations are aligned to two coordinates.
- One of the observations, x_i, has coordinates $(x_{i,1}, x_{i,2})$.
- Similarly, we can represent all the observations by their projected values on the two original coordinates.
- In the right-hand panel we have added two principal components in green.
- The principal components are represented by two new coordinates. In the new coordinate system, each observation, x_i, will new coordinates $(z_{i,1}, z_{i,2})$.
- The new coordinates will depend upon original vector and the relationship between the original coordinates and rotated coordinates.

Principal Components

- Suppose, we have set of features $X_1, X_2, \ldots, X_p$. The first principal component of a set of features is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

- That has maximum variance. In the statement above, the normalized means that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- The elements $\phi_{11}, \ldots, \phi_{p1}$ are called the loadings of the first principal.

Collect PCA model data

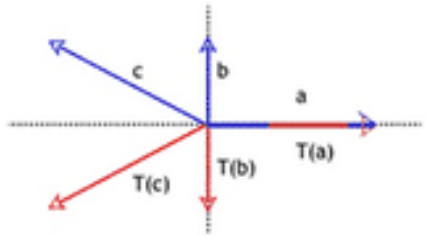
Standardize variables

$$z = \frac{x - \mu}{\sigma}$$

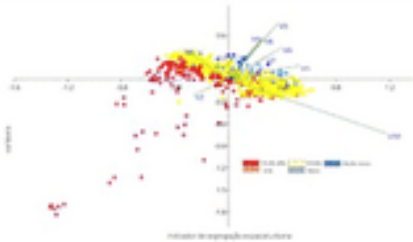
Create correlation matrix

	VE2	VE3	VED2	VH2	VH3
VE2	1,00	0,61	- 0,34	- 0,81	- 0,24
VE3	0,61	1,00	- 0,71	- 0,56	- 0,51
VED2	- 0,34	- 0,71	1,00	0,32	0,45
VH2	- 0,81	- 0,56	0,32	1,00	0,23
VH3	- 0,24	- 0,51	0,45	0,23	1,00

Calculate eigenvalues / eigenvectors



Extract principal components



Calculate scores (indicator)

Census tracts	Urban Inequality
411520005060003	0,24
411520005080006	0,34
411520005080007	0,18
411520005080010	0,32

Validate PCA model

$$KMO = \frac{\sum\sum_{i \neq j} r_{ji}^2}{\sum\sum_{i \neq j} r_{ji}^2 + \sum\sum_{i \neq j} a_{ji}^2}$$

Map indicator

Samatrix.io

Steps for PCA

Step 1 - Standardization

- We standardize the range of the continuous initial variable so that each variable contributes equally to the analysis.
- PCA is very sensitive regarding the variances of the initial variables.
- The variables with larges range will dominate over the variables with small range.
- For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1.
- Mathematically, we can do this by subtracting the mean and dividing by the standard deviation of each variable.

$$z = \frac{value - mean}{standard\ deviation}$$

Step 2 – Covariance Matrix Computation

- We need to understand how the variables of the input data set are varying from the mean with respect to each other.
- In other words, the aim is to see whether there is any relationship between them.
- A high correlation between the variables indicates that they have redundant information.
- In order to identify such correlation, we need to compute the covariance matrix.
- The classic approach to PCA involves the eigen decomposition on the covariance matrix $\sum$. It is a $d \times d$ matrix where d is number of dimensions.

$$cov\,(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

Step 2 – Covariance Matrix Computation

- Let our data set X be the score of five students for three subjects
- In Matrix format

$$marks = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix}$$

Student	Math	English	Art
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

Step 2 – Covariance Matrix Computation

In [**1**]: **import numpy as np**

In [**2**]: marks = np.array([[90,90,60,60,30],[60,90,60,60,30], [90,30,60,90,30]])

- The mean Matrix would be

In [**3**]: mean_marks=np.mean(marks, axis= 1)

In [**4**]: mean_marks
Out[**4**]: array([66., 60., 60.])

Step 2 – Covariance Matrix Computation

- The covariance Matrix would be

```
In [5]: CovMat=np.cov(marks, bias=True)

In [6]: CovMat
Out[6]:
array([[504., 360., 180.],
       [360., 360.,   0.],
       [180.,   0., 720.]])
```

Step 2 – Covariance Matrix Computation

- The variance score for each test is shown along the diagonal.
- The Art test has highest variance (720) whereas English has smallest (360).
- Hence Art score has more variability than the English test.
- The covariance between Math and English is positive (360) whereas the covariance between Math and Art is also positive (180).
- The Covariance between English and Art is zero that shows no relationship between English and Art

Step 3 – Compute Eigenvalue and Eigenvector

- In order to determine the principal components of the data, we need to compute the eigenvalues and eigenvectors from covariance matrix

```
In [7]: eig_val, eig_vec = np.linalg.eig(CovMat)

In [8]: eig_val
Out[8]: array([ 44.81966028, 910.06995304, 629.11038668])

In [9]: eig_vec
Out[9]:
array([[ 0.6487899 , -0.65580225, -0.3859988 ],
       [-0.74104991, -0.4291978 , -0.51636642],
       [-0.17296443, -0.62105769,  0.7644414 ]])
```

Step 4 – Sort Eigenvalue Choose k Eigenvector

- The eigenvectors will form the basis of new feature space but they only define the directions and all of them have unit length.
- To decide which eigenvector(s), we need to drop to get lower dimensional subspace, we should review the corresponding eigenvalues of the eigenvectors.
- The eigenvector corresponding to the lowest eigenvalue bears the lowest information about the distribution of the data and we can drop them.

Step 4 – Sort Eigenvalue Choose k Eigenvector

- We need to rank the eigenvalues from the highest to the lowest and choose the top k eigenvalues and eigenvectors.

```
In [10]: eig_pairs = [(np.abs(eig_val[i]), eig_vec[:,i]) for i in range(len(eig_val))]

In [11]: eig_pairs.sort(key=lambda x: x[0], reverse=True)

In [12]: for i in eig_pairs:
   ...:     print(i[0])
910.0699530410367
629.1103866763253
44.81966028263878
```

Step 4 – Sort Eigenvalue Choose k Eigenvector

- The corresponding eigenvectors are defined as weights

```
In [13]: matrix_w = np.hstack((eig_pairs[0][1].reshape(3,1), eig_pairs[1][1].reshape(3,1)))


In [14]: print('Matrix W:\n', matrix_w)
Matrix W:
 [[-0.65580225 -0.3859988 ]
 [-0.4291978  -0.51636642]
 [-0.62105769  0.7644414 ]]
```

Step 5 – Transform the value in new subspace

- In the last step, we can filter top 2 eigenvectors and transform our sample into new subspace using $\mathbf{y} = w^T \mathbf{x}$

```
In [13]: transformed = matrix_w.T.dot(marks-mean_marks.reshape(3,1))

In [14]: transformed
Out[14]:
array([[-34.37098481,  -9.98345733,   3.93481353, -14.69691716,
         55.11654576],
       [ 13.66927088, -47.68820559,   2.31599277,  25.24923474,
         6.45370719]])
```

Calculate using sklearn

Alternatively, we can directly use sklearn to calculate the values

```
In [17]: from sklearn.decomposition import PCA as sklearnPCA

In [18]: sklearn_pca = sklearnPCA(n_components=2)

In [19]: sklearn_transf = sklearn_pca.fit_transform(marks.T)
```

Calculate using sklearn

```
In [20]: sklearn_transf
Out[20]:
array([[-34.37098481, -13.66927088],
       [ -9.98345733,  47.68820559],
       [  3.93481353,  -2.31599277],
       [-14.69691716, -25.24923474],
       [ 55.11654576,  -6.45370719]])
```

Uniqueness of Principal Components

Each principal component eigenvector is unique, up to a sign flip.

The two different software packages will yield the same eigenvectors; however, the signs of the eigenvectors may differ.

The eigenvector specifies the direction in p-dimensional space.

But flipping the sign has not affect as the direction does not change.

```
In [21]: sklearn_pca.components_

Out[21]:

array([[-0.65580225, -0.4291978 , -0.62105769],

       [ 0.3859988 ,  0.51636642, -0.7644414 ]])
```

Uniqueness of Principal Components

- If we compare the eigenvectors following two different methods.
- Even though the first eigenvector is same in value as well as sign but the second eigenvector is same in value but sign flips.
- In both the cases the value of principal components is same.

Proportion of Variance Explained

- In the previous section, we performed PCA on a three-dimensional data set and projected the data onto the first two principal component vectors to obtain a two-dimensional view of the data.
- This two-dimensional representation of the three-dimensional data successfully captures the major patterns in the data.
- The question arises how much information in a given data is lost by projecting the observations onto the first few principal components?
- How much information is not included in the first few principal components?
- We are interested in proportion of variance explained (PVE) by each principal component.

Proportion of Variance Explained

- We can find out PVE of mth principal component by dividing variance explained by mth principal component and total variance present in the data set.
- The PVE of each principal component is a positive quantity.
- To compute the cumulative PVE of first M principal components, we can sum the first M PVEs.
- The total PVE of all the principal components is one.

Proportion of Variance Explained

We can find out the variance explained using eigenvalue (from step 8)

```
In [22]: eig_val[::-1].sort()

In [23]: eig_val
Out[23]: array([910.06995304, 629.11038668,  44.81966028])

In [24]: eig_val/eig_val.sum()
Out[24]: array([0.57453911, 0.39716565, 0.02829524])
```

We can also find the value explained using sklearn for the top 2 principal components

```
In [25]: sklearn_pca.explained_variance_ratio_
Out[25]: array([0.57453911, 0.39716565])
```

Proportion of Variance Explained

Cumulative variance explained is

```
In [26]: sklearn_pca.explained_variance_ratio_.cumsum()
Out[26]: array([0.57453911, 0.97170476])
```

We can see that in this case, the first 2 components are able to explain 97.17% variance.

Deciding the number of components

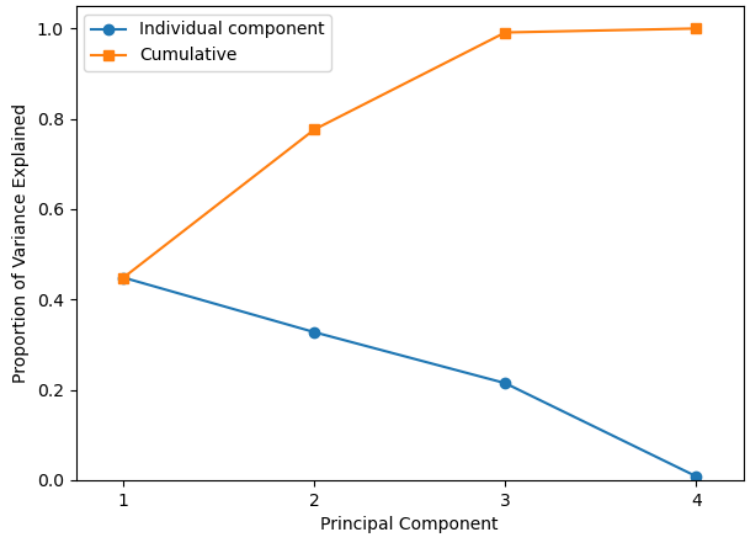
- A $n \times p$ data matrix has $\min(n-1, p)$ distinct principal components. But we may not be interested in all of them.
- We may be interested in the first few of them.
- What should be the smallest number of principal components that we need to get a good understanding of the data.
- However, there is no single answer to this question.
- We have added one more column stating the marks obtained in Science test and fit the data to PCA model.
- We have plotted the explained variance ratio and cumulative variance ratio.

Deciding the number of components

```
In [27]: marks1 = np.array([[90,90,60,60,30],[60,90,60,60,30],[90,30,60,90,30],[
   ...: 90,60,30,60,90]])

In [28]: sklearn_pca = sklearnPCA(n_components=4)

In [29]: sklearn_transf = sklearn_pca.fit_transform(marks1.T)
```

Deciding the number of components

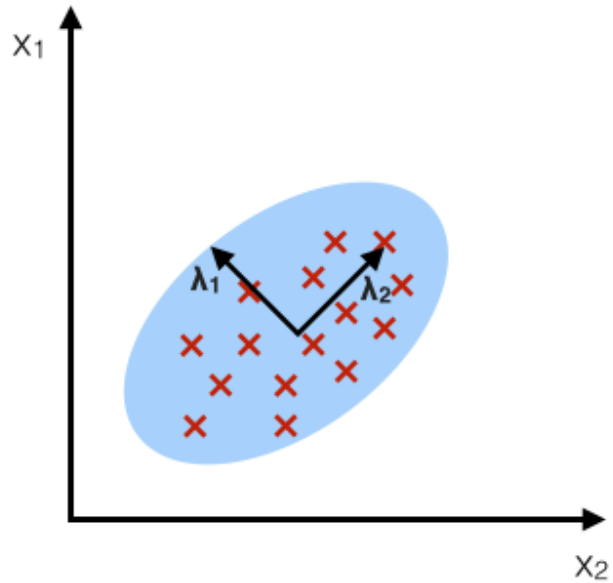
```
In [30]: from matplotlib import pyplot as plt

In [32]: plt.figure(figsize=(7,5))
   ...: plt.plot([1,2,3,4], sklearn_pca.explained_variance_ratio_, '-o', label=
   ...: 'Individual component')
   ...: plt.plot([1,2,3,4], np.cumsum(sklearn_pca.explained_variance_ratio_), '
   ...: -s', label='Cumulative')
   ...: plt.ylabel('Proportion of Variance Explained')
   ...: plt.xlabel('Principal Component')
   ...: plt.xlim(0.75,4.25)
   ...: plt.ylim(0,1.05)
   ...: plt.xticks([1,2,3,4])
   ...: plt.legend(loc=2)
   ...: plt.show()
```



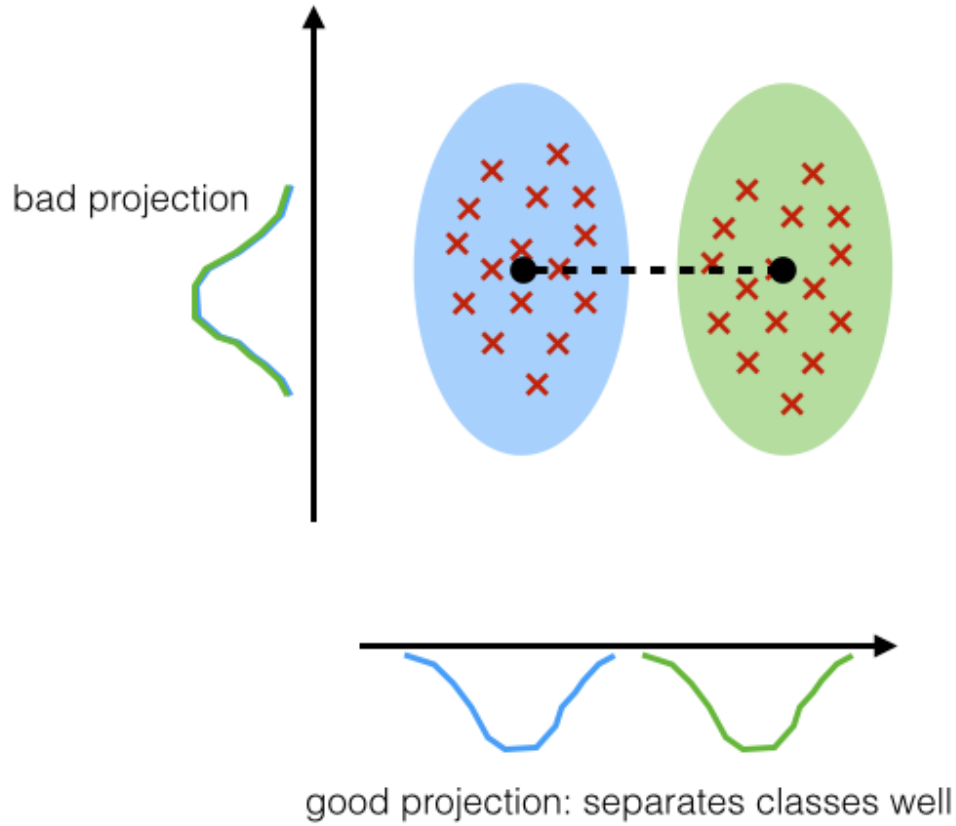
Deciding the number of components

- We generally decide on the number of principal components required by examining a screen plot such as illustrated in Figure – 4.
- We need the smallest number of principal components that can explain a sizable amount of variation in the data.
- We can do so by eyeballing the screen plot and looking for a point at which the proportion of variance explained by each subsequent principal component drops off.
- This is often referred to as an elbow in the screen plot.
- By inspection of Figure – 4, one might conclude a fair amount of variance has been explained by the first three principal components and there is an elbow after the third component.
- The fourth principal component explains a very small amount of variance. Hence, it is worthless.

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



Samatrix.io

Features	Principal Component Analysis	Linear Discriminant Analysis
Method of learning	Unsupervised	Supervised
Focus	Its searches for the direction that has the largest variations	Maximizes ratio between class variation and within-class variation
Computation for large dataset	Requires fewer computations	Requires more computation than PCA for a large dataset
Discrimination between classes	Deals without paying any particular attention to the class structure	Directly deals with discrimination between classes
Well distributed classes in a small dataset	PCA is less superior to LDA	LDA is more superior to PCA

Clustering Methods

- Non – Hierarchical Clustering
- Hierarchical Clustering
- Gaussian Mixture Models
- Density Based Clustering

Clustering

- The techniques used for finding subgroups, or clusters, in a data set are known as **clustering**.
- By using the clustering techniques, we can group the observation together so that the observations within each group are quite similar to each other, whereas the observations in different groups are different from each other.
- However, we need to define how to define whether two or more observations are similar or different.
- This requires domain-specific knowledge and knowledge about the data that is being studied.

Clustering

- Fundamentally, all clustering methods use the same approach i.e.

  first we calculate similarities and then we use it to cluster the data points into groups or batches.

- E.g. K-Means (distance between points)
- DBSCAN (distance between nearest points)
- Gaussian mixtures (Mahalanobis distance to centers)
- Spectral clustering (graph distance), etc.

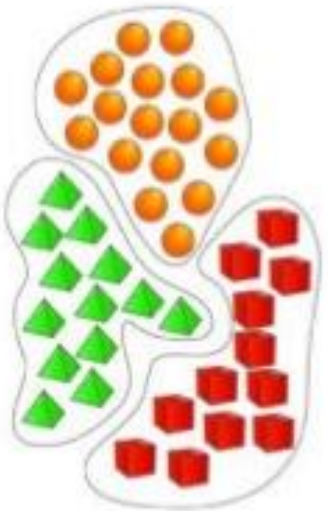
Clustering

- The objective of both clustering and PCA is to simplify the data via a small number of summaries even though their mechanisms are different.
- The objective of PCA is to find a low-dimensional representation of the observations that can explain a good fraction of the variance.
- The objective of clustering is to find a homogeneous subgroup among the observations

Clustering

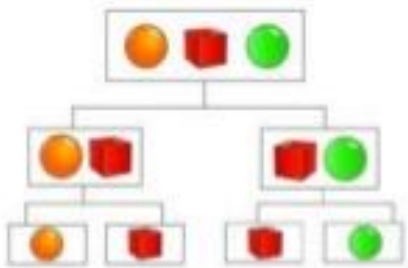
- Clustering is a popular technique.
- A number of clustering methods are available.
- The two best-known clustering approaches are **K – means clustering** and **hierarchical clustering**.
- In K – means clustering, we partition the observations into a pre-specified number of clusters.
- In hierarchical clustering, we do not know about the number of clusters, we prepare a tree-like visual representation of the observations, called a **dendrogram**.
- The dendrogram helps us view the clustering obtained at once for each possible number of clusters from 1 to p.

Types of Clusters



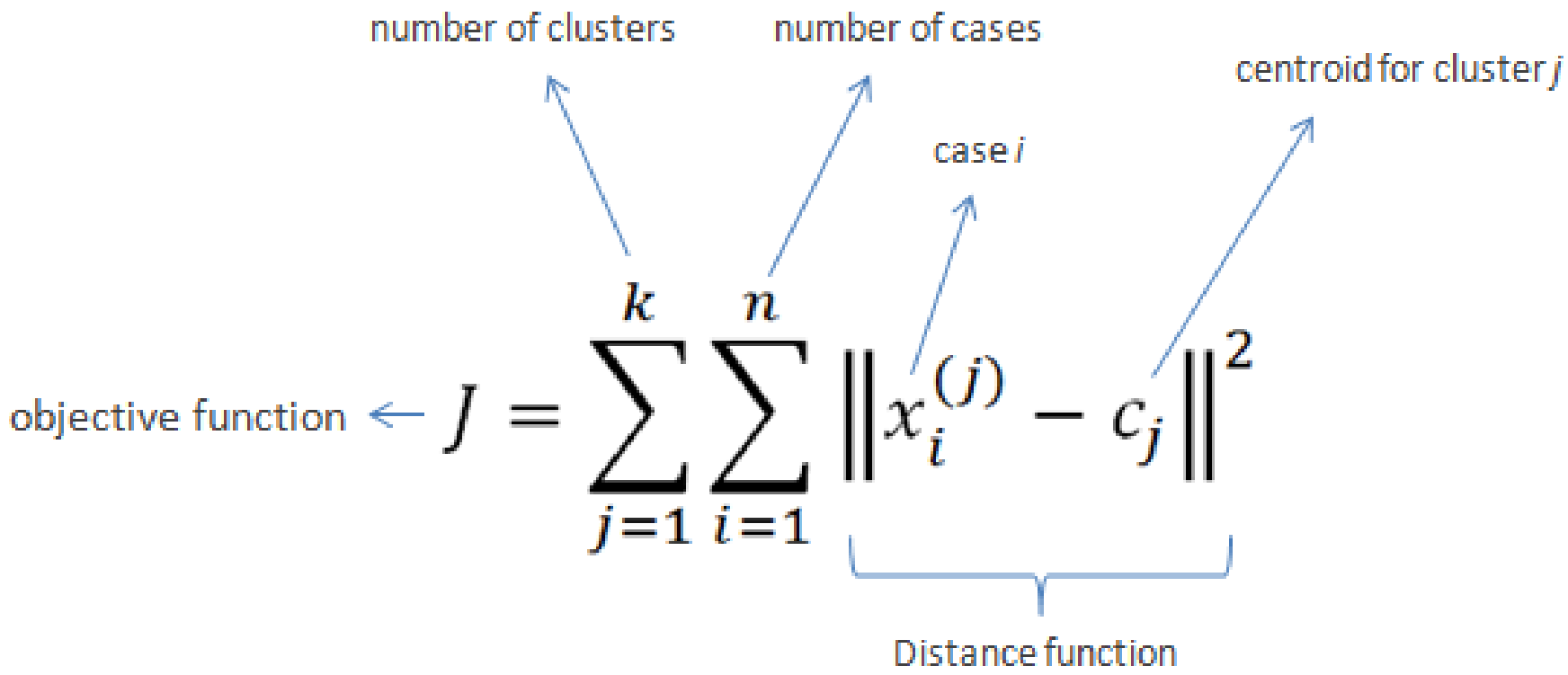
- ***Partitional* clustering or non-hierarchical :** A division of objects into non-overlapping subsets (clusters) such that each object is in exactly one cluster
- The non-hierarchical methods divide a dataset of N objects into M clusters.
- **K-means clustering**, a non-hierarchical technique, is the most commonly used one in business analytics

- ***Hierarchical* clustering:** A set of nested clusters organized as a hierarchical tree
- The hierarchical methods produce a set of nested clusters in which each pair of objects or clusters is progressively nested in a larger cluster until only one cluster remains
- **CHAID tree** is most widely used in business analytics



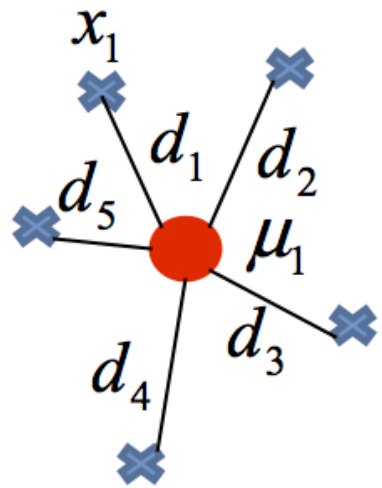
Non-Hierarchical Clustering

K-Means Clustering

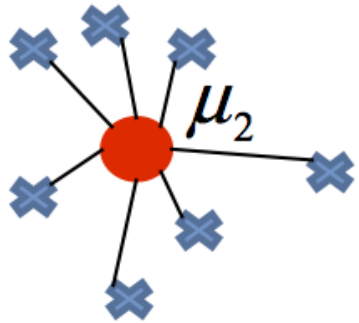

number of clusters
number of cases
centroid for cluster j
case i
objective function
$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$
Distance function

Centroid

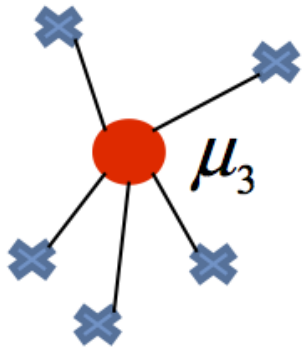
Sample



$$\sum_{x_j \in S_2} d_j^2$$



$$\sum_{x_j \in S_1} d_j^2 = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2$$



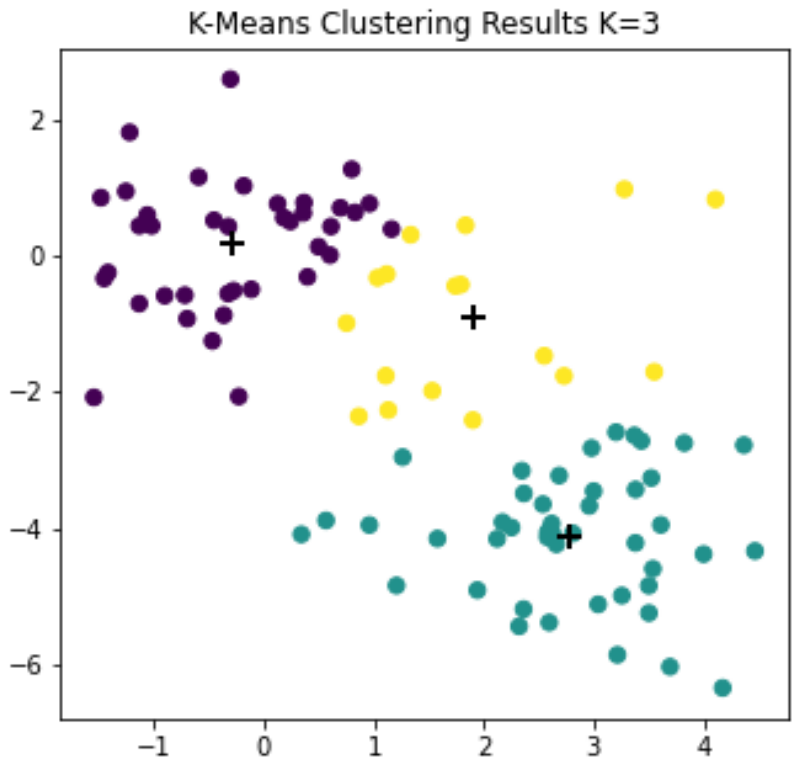
$$\sum_{x_j \in S_3} d_j^2$$

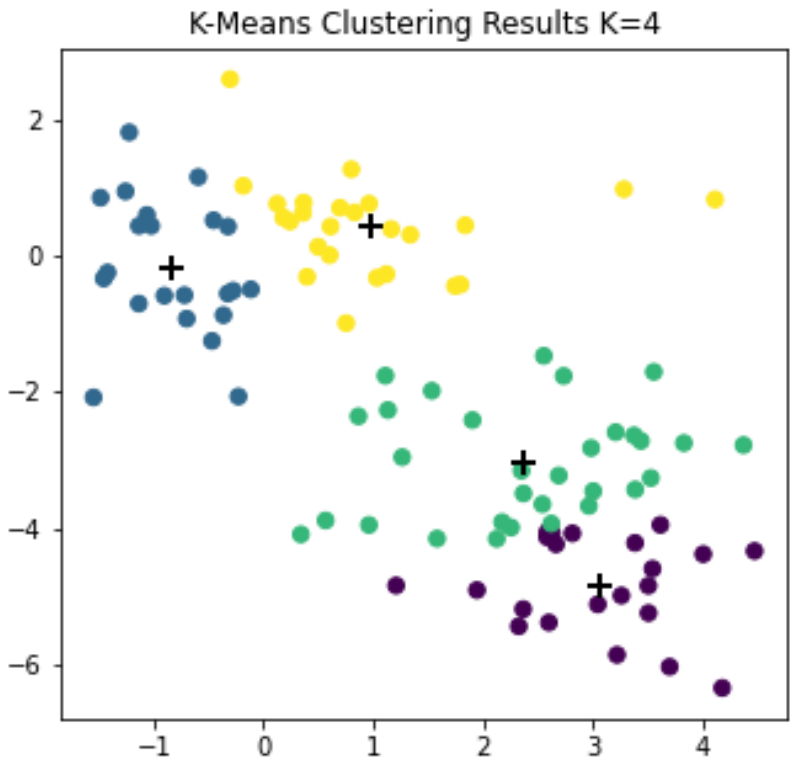
$$\min_S E(\mu_i) = \sum_{x_j \in S_1} d_j^2 + \sum_{x_j \in S_2} d_j^2 + \sum_{x_j \in S_3} d_j^2$$

K- Means Clustering

- K – means clustering is used to partition the data set into K distinct, non-overlapping clusters.
- We need to specify the desired number of clusters K, in order to perform K – means clustering.
- In Figure – 5, we have illustrated the results that we obtained after performing K – means clustering on a simulated example that consists of 200 observations in two dimensions.
- We used three different values of K. The color of each observation shows the cluster to which the observation was assigned as a result of applying the K – means clustering algorithm.

K- Means Clustering





Samatrix.io

K- Means Clustering

- The clusters follow the following two properties
  - Each observation should belong to at least one of the K clusters.
  - The clusters do not overlap each other. No observation should belong to more than one cluster.
- For a good clustering, within-cluster variation should be as small as possible.
- The within-cluster variation measures the amount by which the observations within a cluster differ from each other.
- The objective is to partition the observations into K clusters in such a way that the sum of within-cluster variation for all the clusters is as small as possible.

K- Means Clustering

- In other words, the within-cluster variation for the kth cluster is the sum of all of the pairwise squared Euclidean distances between the observations in the kth cluster, divided by the total number of observations in the kth cluster.

K- Means Clustering Python Example

```
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(2)
X = np.random.standard_normal((100,2))
X[:50,0] = X[:50,0]+3
X[:50,1] = X[:50,1]-4
km1 = KMeans(n_clusters=2, n_init=20)
km1.fit(X)
 np.random.seed(4)
km2 = KMeans(n_clusters=3, n_init=20)
km2.fit(X)
```

K- Means Clustering Python Example

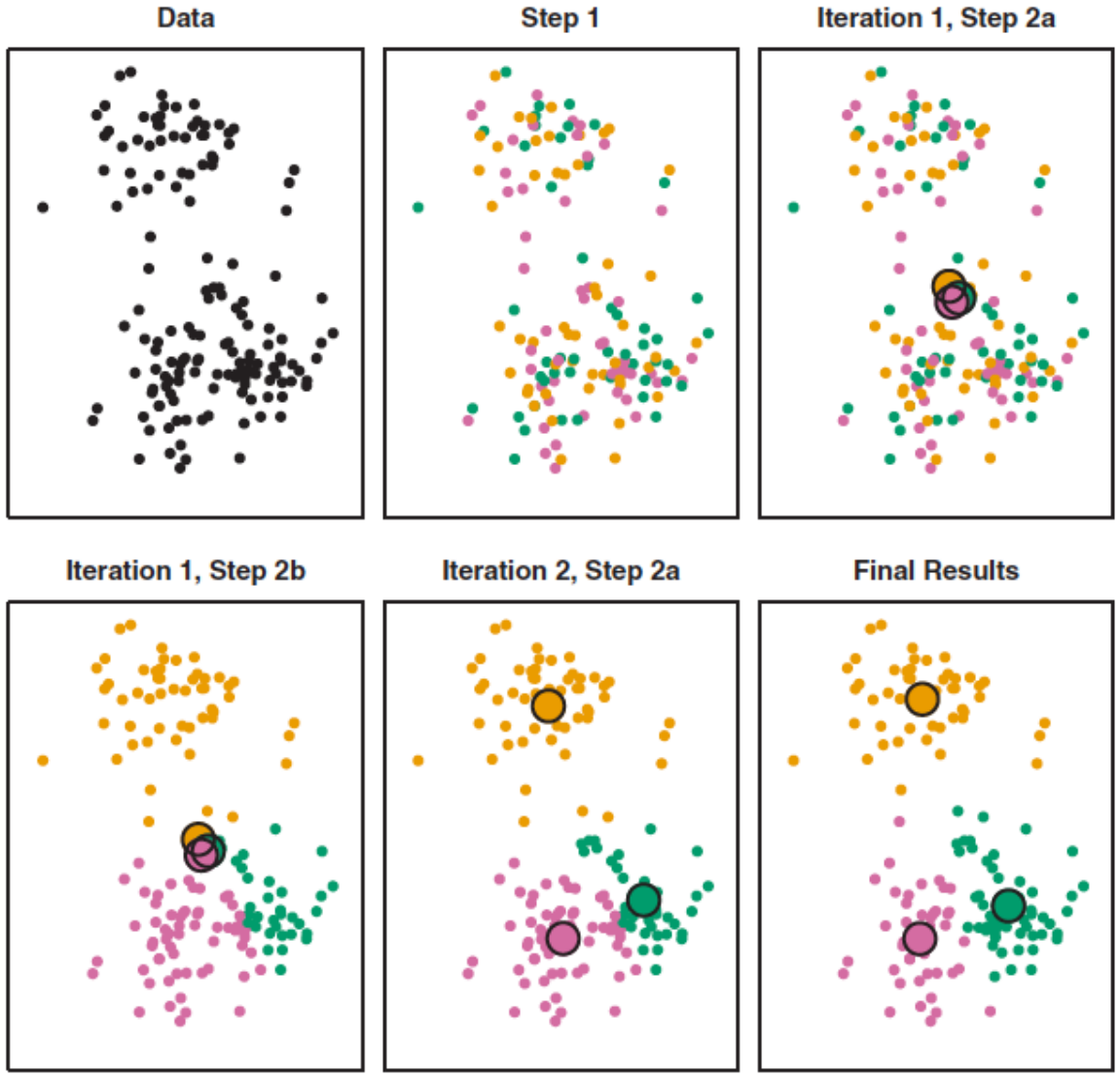
```
np.random.seed(6)
km3 = KMeans(n_clusters=4, n_init=20)
km3.fit(X)
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(18,5))
ax1.scatter(X[:,0], X[:,1], s=40, c=km1.labels_)
ax1.set_title('K-Means Clustering Results K=2')
ax1.scatter(km1.cluster_centers_[:,0], km1.cluster_centers_[:,1], marker='+', s=100, c='k', linewidth=2)
ax2.scatter(X[:,0], X[:,1], s=40, c=km2.labels_)
ax2.set_title('K-Means Clustering Results K=3')
ax2.scatter(km2.cluster_centers_[:,0], km2.cluster_centers_[:,1], marker='+', s=100, c='k', linewidth=2);
ax3.scatter(X[:,0], X[:,1], s=40, c=km3.labels_)
ax3.set_title('K-Means Clustering Results K=4')
ax3.scatter(km3.cluster_centers_[:,0], km3.cluster_centers_[:,1], marker='+', s=100, c='k', linewidth=2);
```

K- Means Clustering Algorithm

The algorithm of K- Means clustering is

1. Initial cluster assignment: In this step, we randomly assign a number from 1 to K to each observation
2. Iterate until the cluster assignments stop changing:
   a. Computer the cluster centroid for each of the K clusters
   b. Assign each observation to the cluster such that the Euclidean distance between the cluster and the centroid is minimum.

K- Means Clustering Algorithm



Samatrix.io

Limitation

- Predetermined number of clusters
- It always tries to create the clusters of the same size.

Hierarchical Clustering

Hierarchical Clustering

- Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis or HCA**.
- In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.
- When compared with K- means clustering, hierarchical clustering does not require a particular choice of K.

Two Approaches:

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.

2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach**

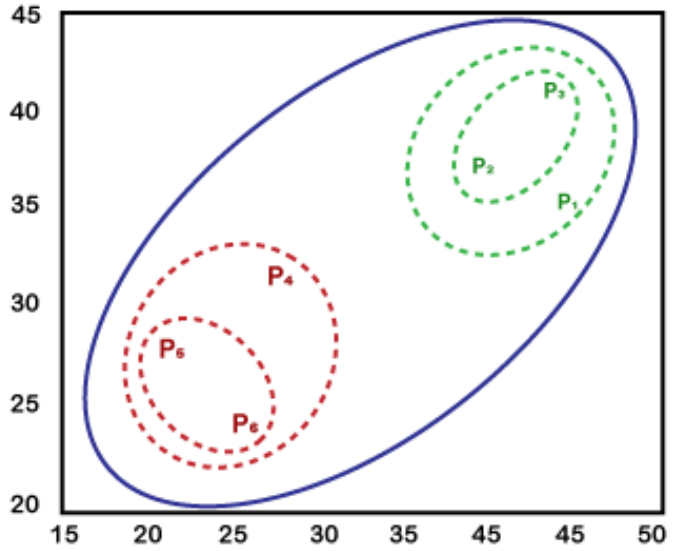
Agglomerative Hierarchical clustering

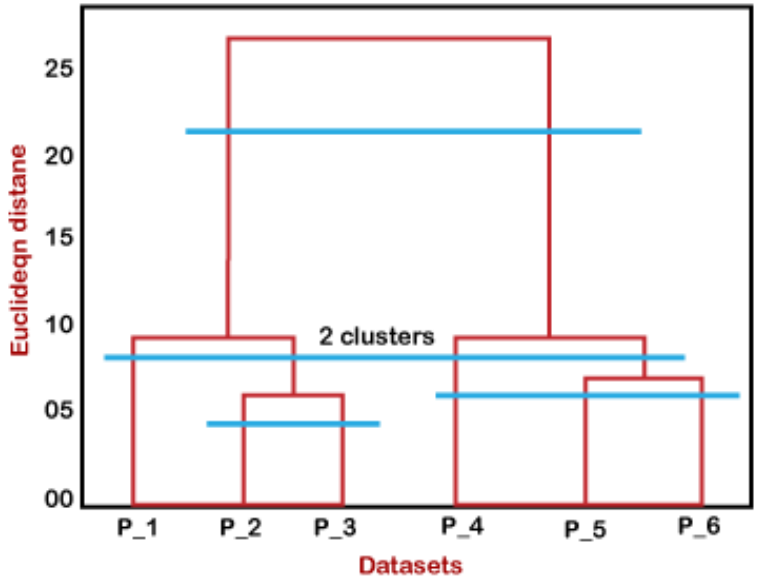
- The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the bottom-up approach.
- It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

- **This hierarchy of clusters is represented in the form of the dendrogram.**

Dendrogram

- The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs.
  - **Y-axis - shows the Euclidean distances between the data points**
  - **X-axis - shows all the data points of the given dataset**

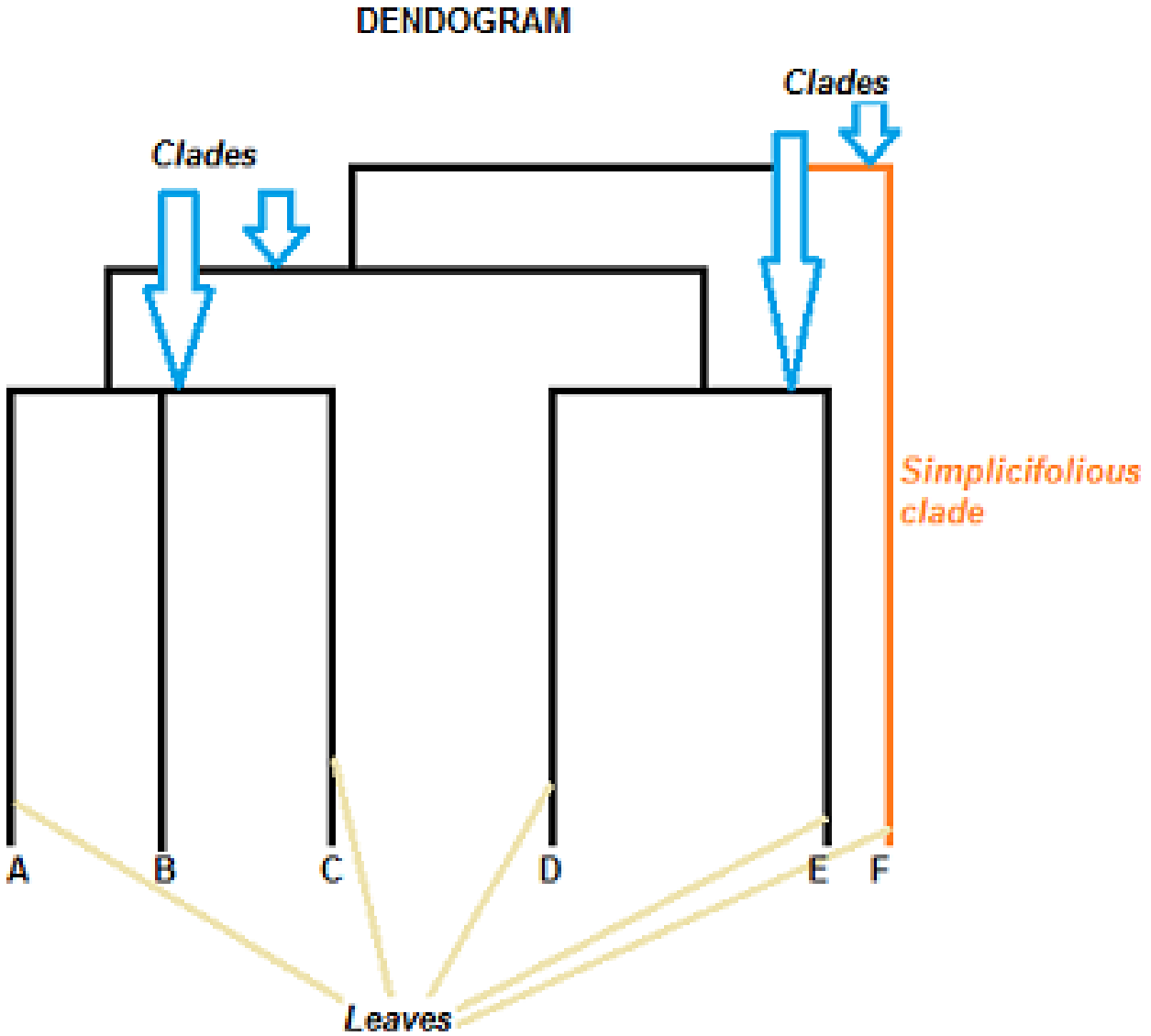




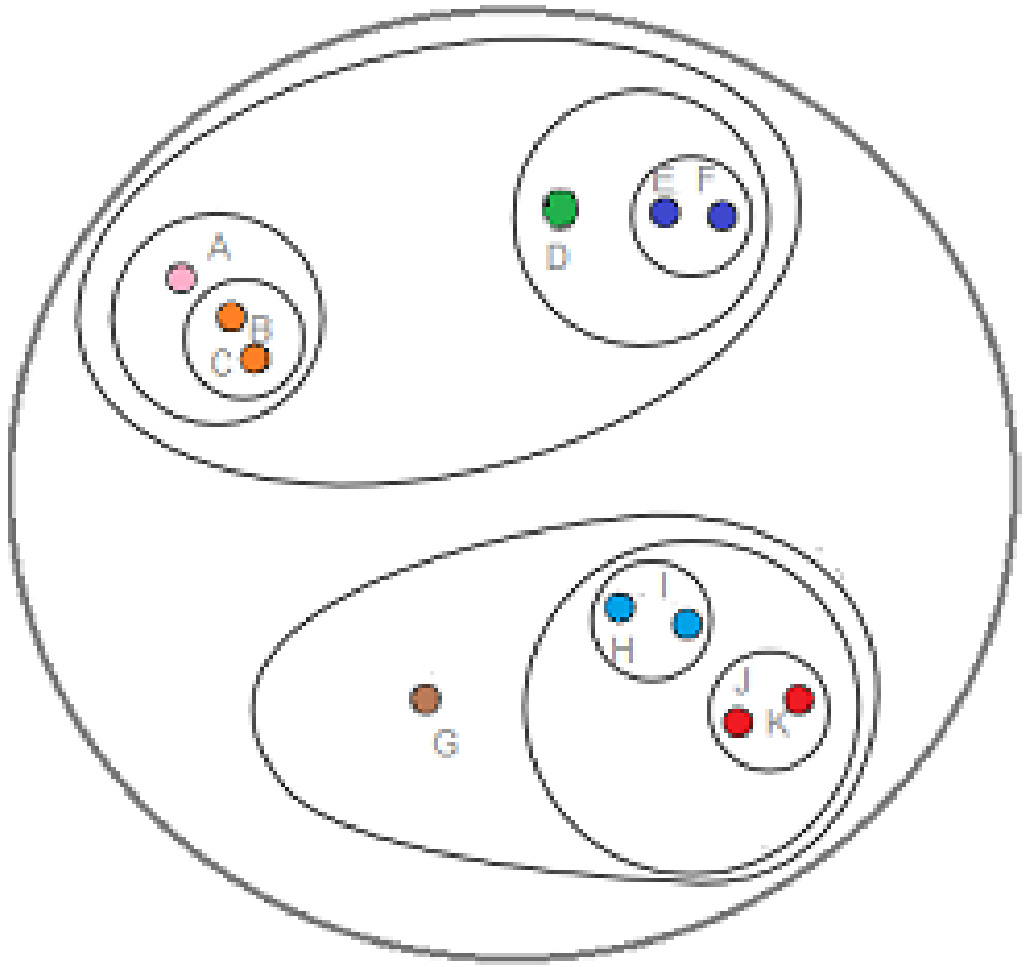
Parts of Dendrogram

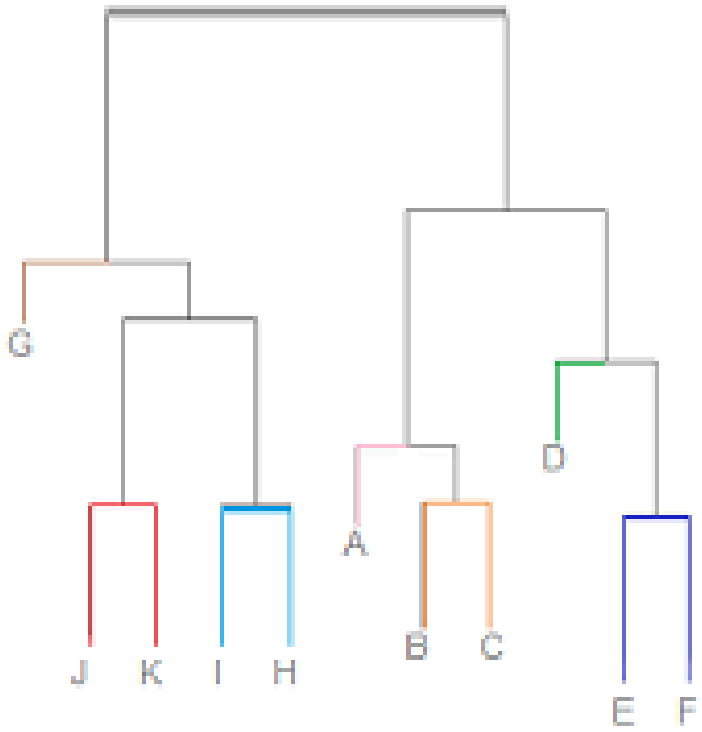
The clade is the branch. Usually labeled with Greek letters from left to right (e.g. α β, δ…).

Each clade has one or more leaves. The leaves in the above image are:
Single (simplicifolius): F
Double (bifolius): D E
Triple (trifolious): A B C

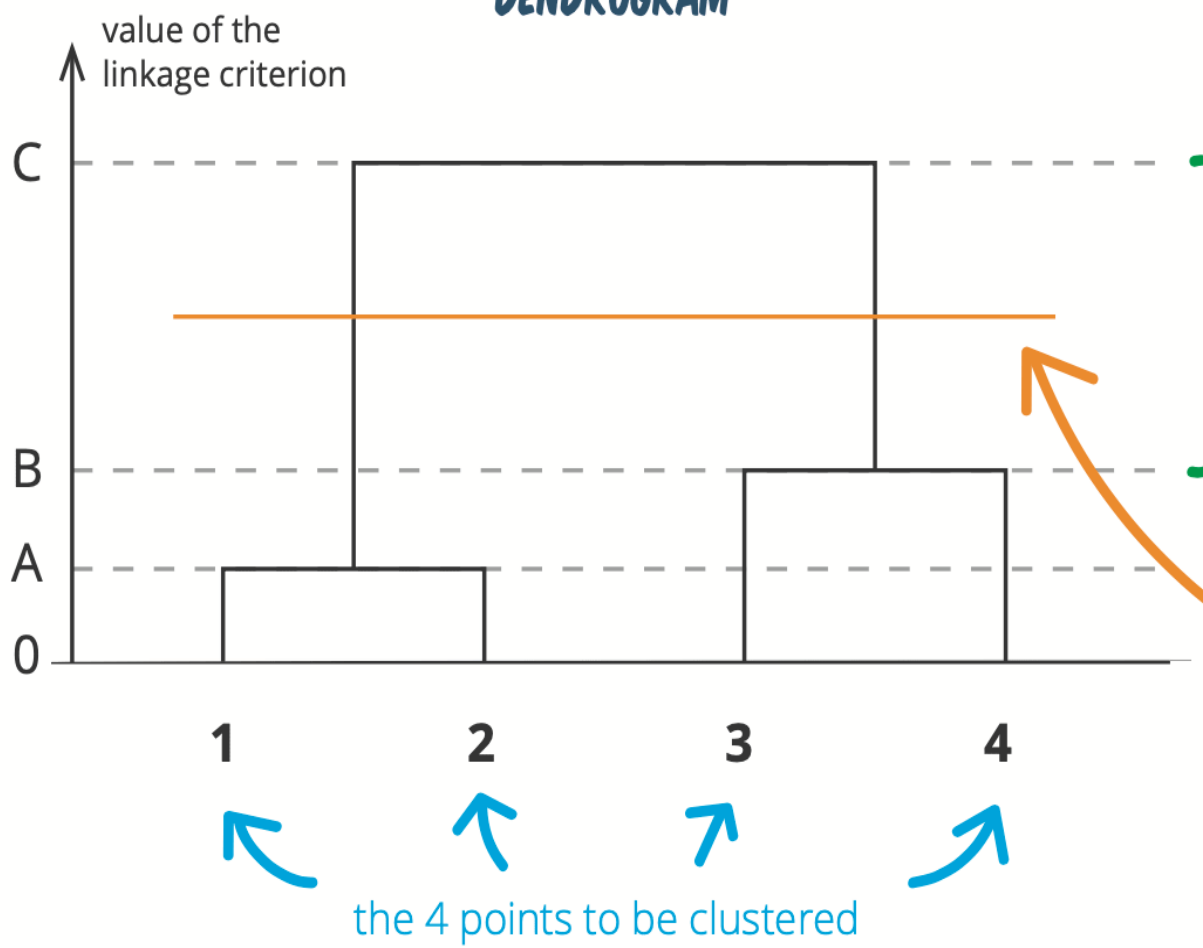


Samatrix.io





DENDROGRAM



the value of the linkage criterion between {1,2} and {3,4} is C, they are merged third

the value of the linkage criterion between 3 and 4 is B, they are merged second

the value of the linkage criterion between 1 and 2 is A, they are merged first

the gap between B and C is the largest (much larger than the gap between 0 and A or the gap between A and B

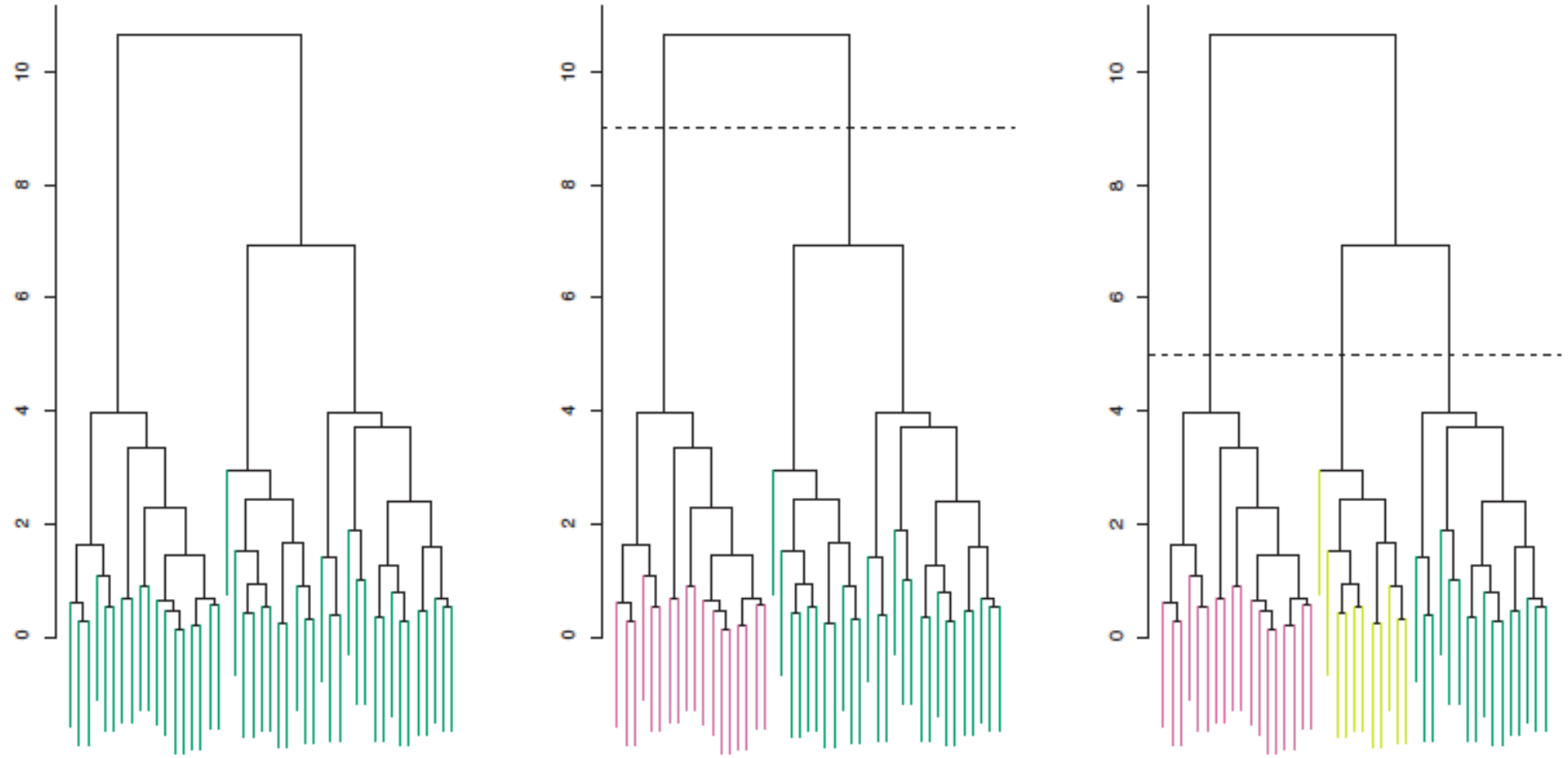
the "cut" associated with the largest gap generates two clusters: {1,2} and {3,4}

the 4 points to be clustered

Interpreting a Dendrogram

- We have plotted the hierarchical clustering using 45 observations in a two-dimensional space.
- The results have been shown in Figure – 6. In the left-hand panel of Figure – 6, each leaf of the dendrogram represents one of the 45 observations.
- Following a bottom-up approach, as we move up the tree, some leaves that are similar to each other start to fuse into branches.
- As we go higher up the tree, branches fuse with either other branches or leaves.
- During the fusion process, a similar group of observations fuses with each other at the early stage (lower in the tree).

Interpreting a Dendrogram



Samatrix.io

Interpreting a Dendrogram

- The observations that fuse at a later stage (near the top of the tree), can be different from each other.
- Hence, we need to look for the point in the tree where the branches containing those two observations are fused first.
- We can measure the height of the fusion on the vertical axis.
- The observations that fuse with each other at the bottom of the tree are very similar to each other.
- On the other hand, the observations that fuse towards the top of the tree will be quite different from each other.

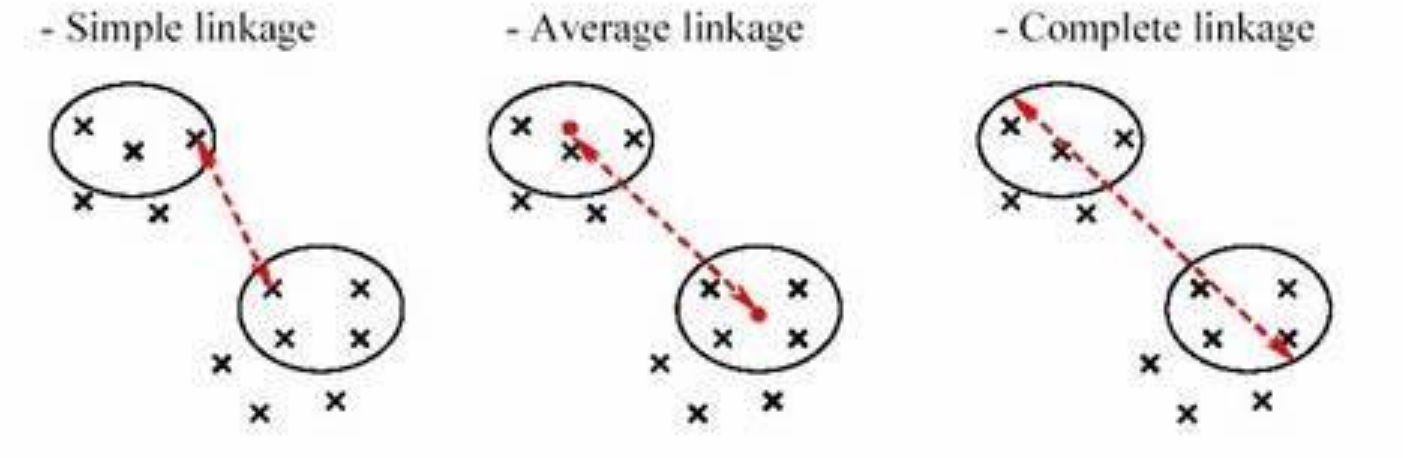
Interpreting a Dendrogram

- We can identify the clusters using the dendrogram. As shown in the middle and right-hand panels, we cut across the dendrogram.
- The distinct set of observations below the cut represents clusters. In the middle panel, we cut the dendrogram at the height of 9,
- Therefore, we get two clusters. In the right-hand side panel, we get three clusters by cutting the dendrogram at the height of 5.
- The range of vertical heights is between 0 cuts and n cuts (so that each observation is in its own cluster).
- So, the height of the cut in the dendrogram plays a similar role as played by the K in K-means clustering.
- Any number of clusters can be obtained by using one single dendrogram.

Linkage

- One of the important concepts is the dissimilarity between pairs of observation and pairs of groups of observations.
- The term linkage defines the dissimilarity between two groups of observations.
- The three most common types of linkage are as follows
  - Complete – Maximal inter-cluster dissimilarity
  - Single – Minimal inter-cluster dissimilarity
  - Average – Mean inter-cluster dissimilarity

Linkage

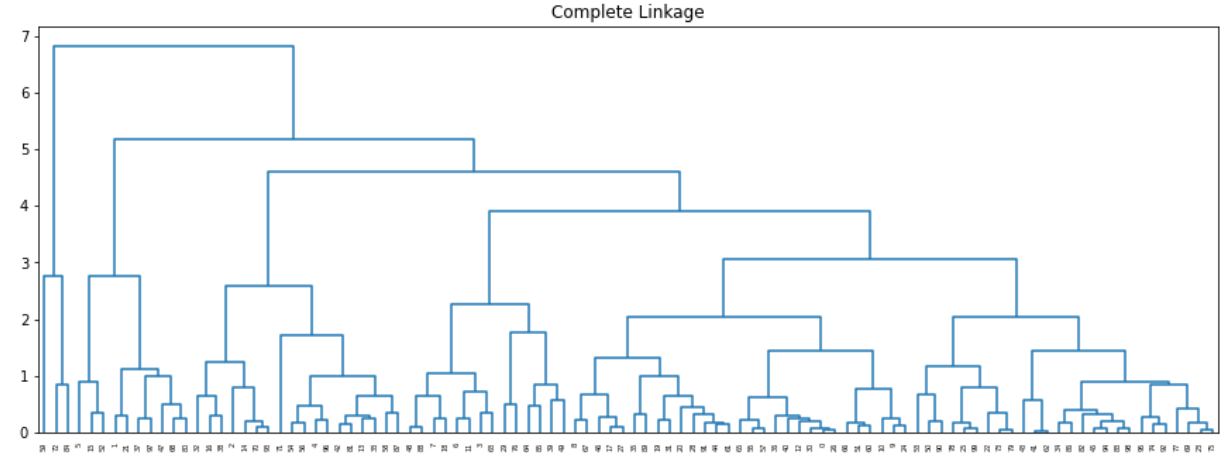


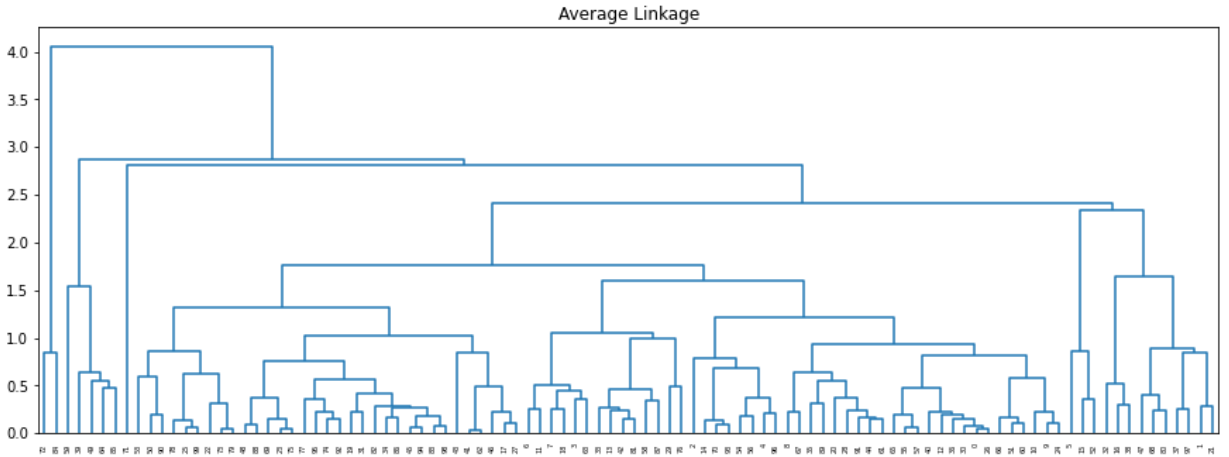
- **Complete linkage**: similarity of the farthest pair. One drawback is that outliers can cause merging of close groups later than is optimal.
- **Single-linkage**: similarity of the closest pair. This can cause premature merging of groups with close pairs, even if those groups are quite dissimilar overall.
- **Simple Average-linkage**: distance between clusters as the average distance between each of the members, weighted so that the two clusters have an equal influence on the final output.

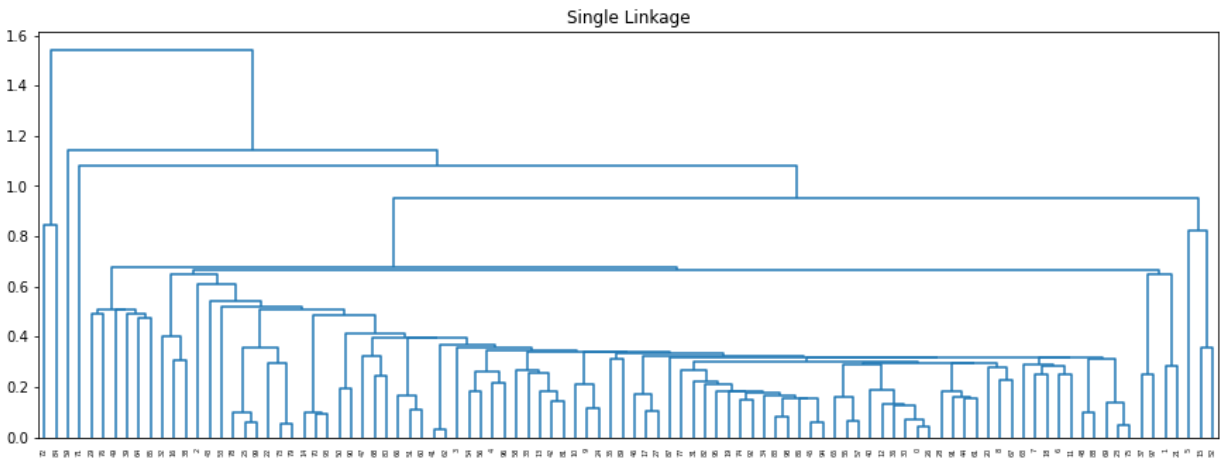
Linkage

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster import hierarchy


np.random.seed(2)
X = np.random.standard_normal((100,2))
 fig, (ax1,ax2,ax3) = plt.subplots(3,1, figsize=(15,18))
 for linkage, cluster, ax in zip([hierarchy.complete(X), hierarchy.average(X), hierarchy.single(X)], ['c1','c2','c3'], [ax1,ax2,ax3]):
            cluster = hierarchy.dendrogram(linkage, ax=ax, color_threshold=0)


ax1.set_title('Complete Linkage')
ax2.set_title('Average Linkage')
ax3.set_title('Single Linkage');
```

Linkage







Samatrix.io

Hierarchical methods	*Non-hierarchical methods (K-means)*
• No decision about the number of clusters • Problems when data contain a high level of error • Can be very slow, preferable with small data-sets • Initial decisions are more influential (one-step only) • At each steps they require computation of the full proximity matrix	• Faster, more reliable, works with large data-sets • Need to specify the number of clusters • Need to set the initial seeds • Only cluster distances to seeds need to be computed in each iteration

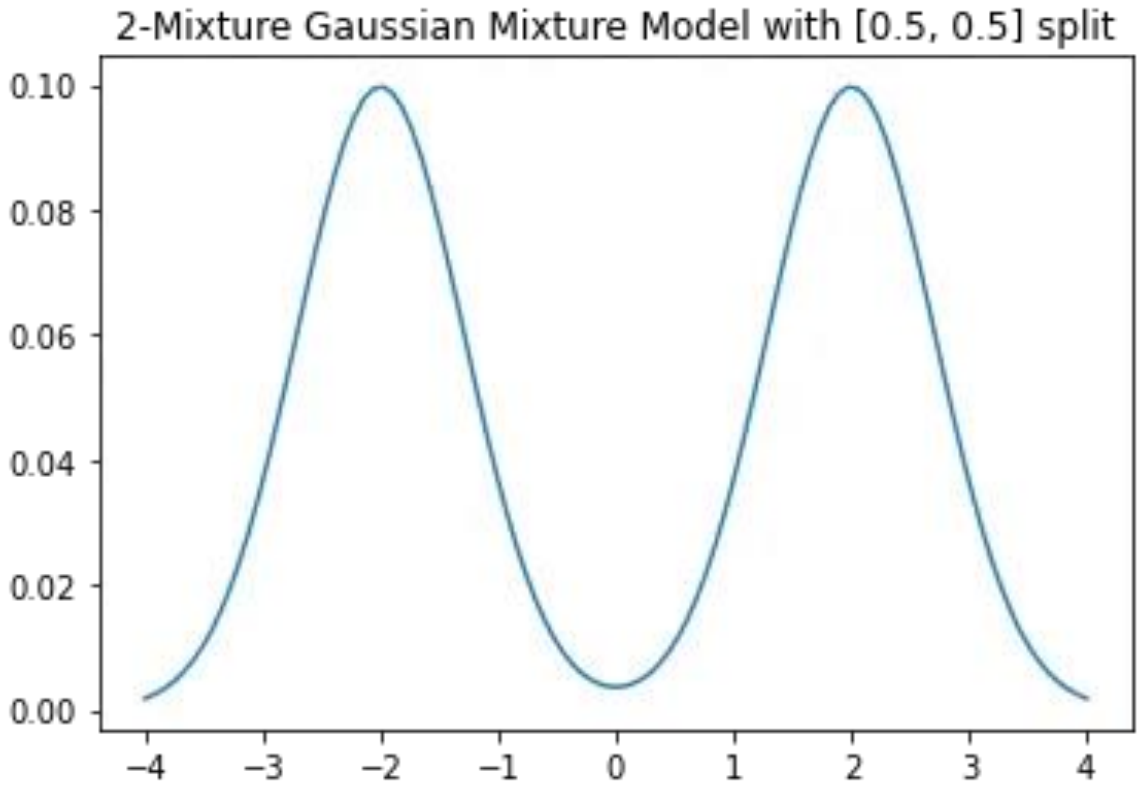
Gaussian Mixture Models

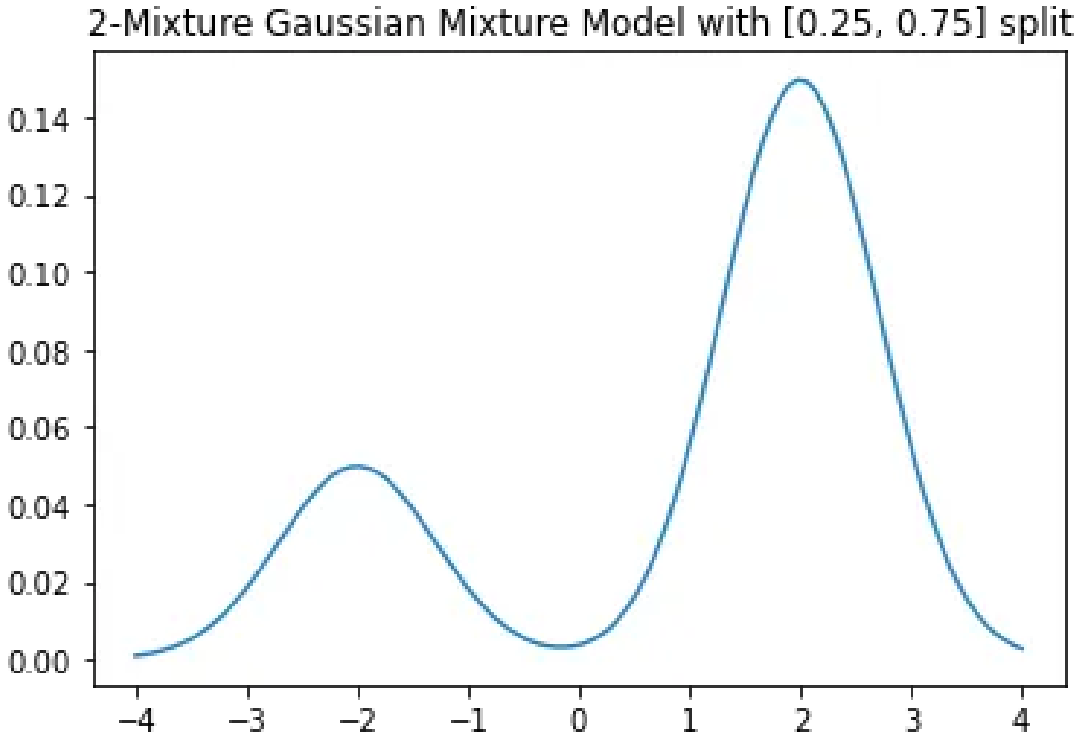
Normal or Gaussian Distribution

- Many datasets can be modeled by Gaussian Distribution.
- So it is quite natural and intuitive to assume that the clusters come from different Gaussian Distributions.
- Or in other words, it tried to model the dataset as a mixture of several Gaussian Distributions.
- **This is the core idea of this model.**

Gaussian Mixture Model

- Essentially, a mixture model (or mixture distribution) is a combination of multiple probability distributions into a single one and requires the estimation of the mean and standard deviation parameters for each.
- To combine these distributions together, we assign a weight to each component distribution such that the total probability under the distribution sums to 1.
- A simple example would be a mixture distribution that consists of 2 gaussians. We can have 2 distributions of different means and variance and combine the 2 using different weights





2 GMMs with the same mean but different π values

Expectation-Maximization (EM) Algorithm

- The Expectation-Maximization (EM) algorithm is an iterative way to find maximum-likelihood estimates for model parameters when the data is incomplete or has some missing data points or has some hidden variables.
- EM chooses some random values for the missing data points and estimates a new set of data.
- These new values are then recursively used to estimate a better first date, by filling up missing points, until the values get fixed.

- In the Expectation-Maximization (EM) algorithm, the estimation step (E-step) and maximization step (M-step) are the two most important steps that are iteratively performed to update the model parameters until the model convergence.

  - E-Step. Estimate the expected value for each latent variable.
  - M-Step. Optimize the parameters of the distribution using maximum likelihood.

Density Based Clustering (DBSCAN)

DBSCAN

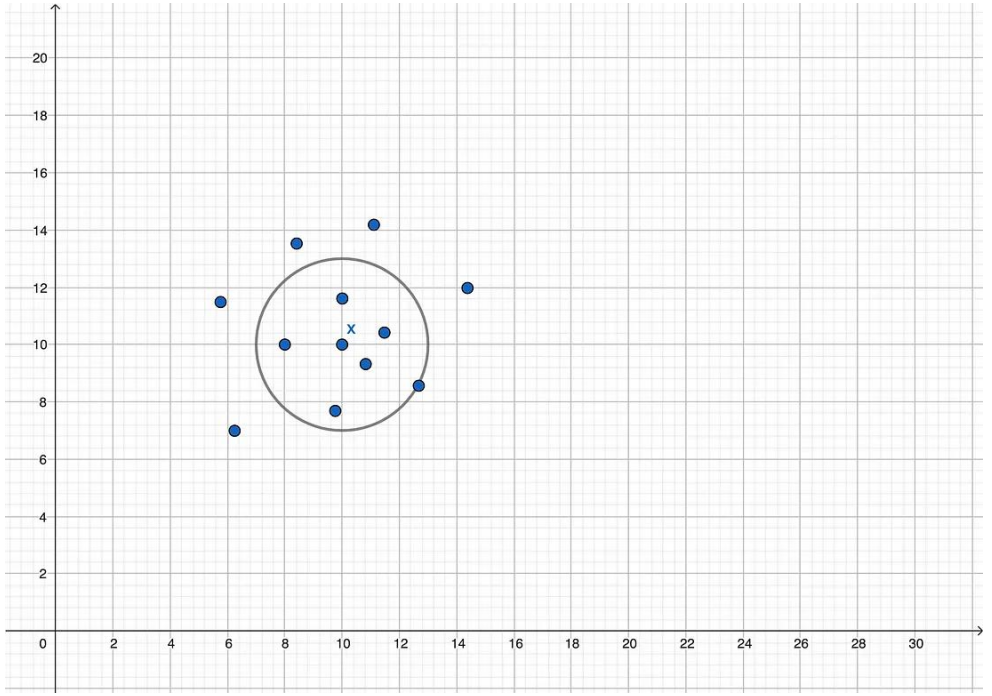
- DBSCAN stands for **Density-Based Spatial Clustering of Applications with Noise**.
- It depends on a density-based notion of cluster.
- It also identifies clusters of arbitrary size in the spatial database with outliers.

DBSCAN Parameters

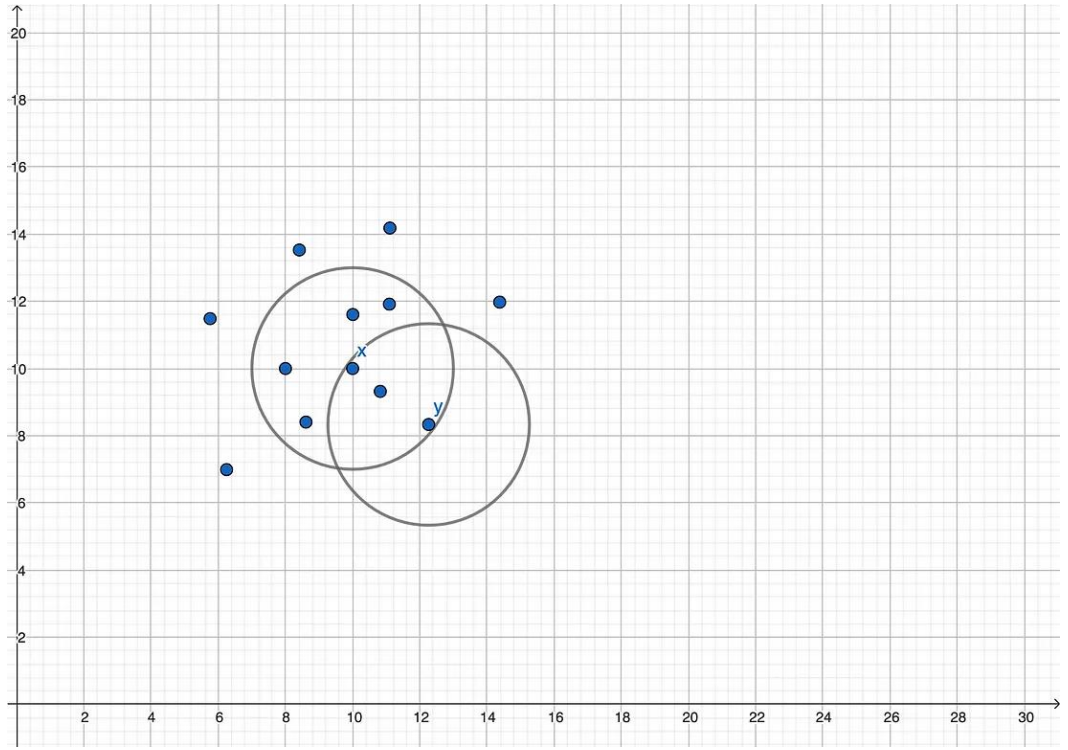
- MinPts = The minimum points that a neighborhood of a data point needs to have in order for the data point to be considered a core point of the cluster
- ε = The radius of the neighborhoods (i.e. how big we want our “circles” around the center data point to be).

Based on those parameters, the DBSCAN algorithm divides data points into three categories:

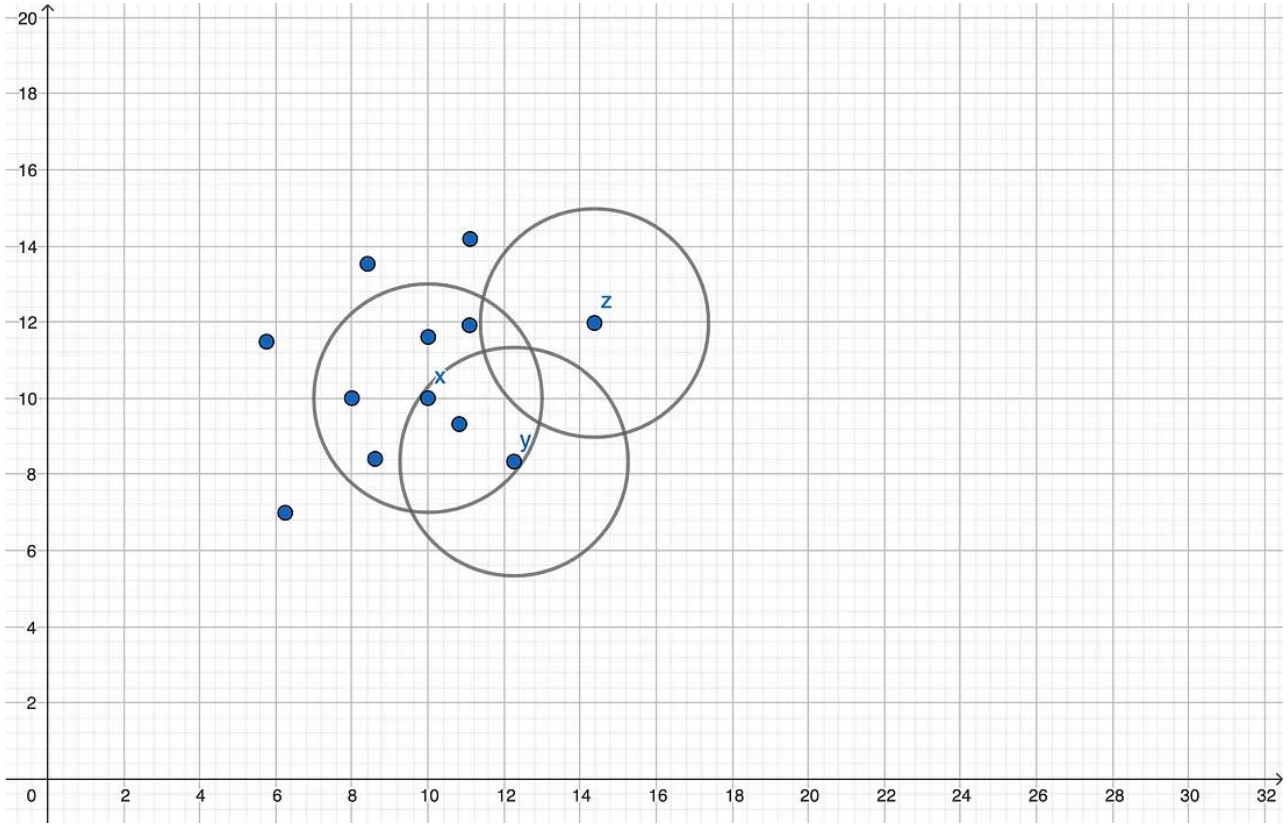
- Core Point : This type of data point has density ≥ MinPts. Meaning that the number of data points in N(x, ε) is greater than or equal to MinPts.
- Example : MinPts = 4, ε = 3 then **x is a core point**



- 2. Border Point : This type of data point has density < MinPts but distance(y, cp) ≤ ε where cp is a core point.
- Meaning that those points do not have the density required to be core points but they are also not that far away from a core point to be considered noise.
- Example : MinPts = 4, ε = 3 then **y is a border point**

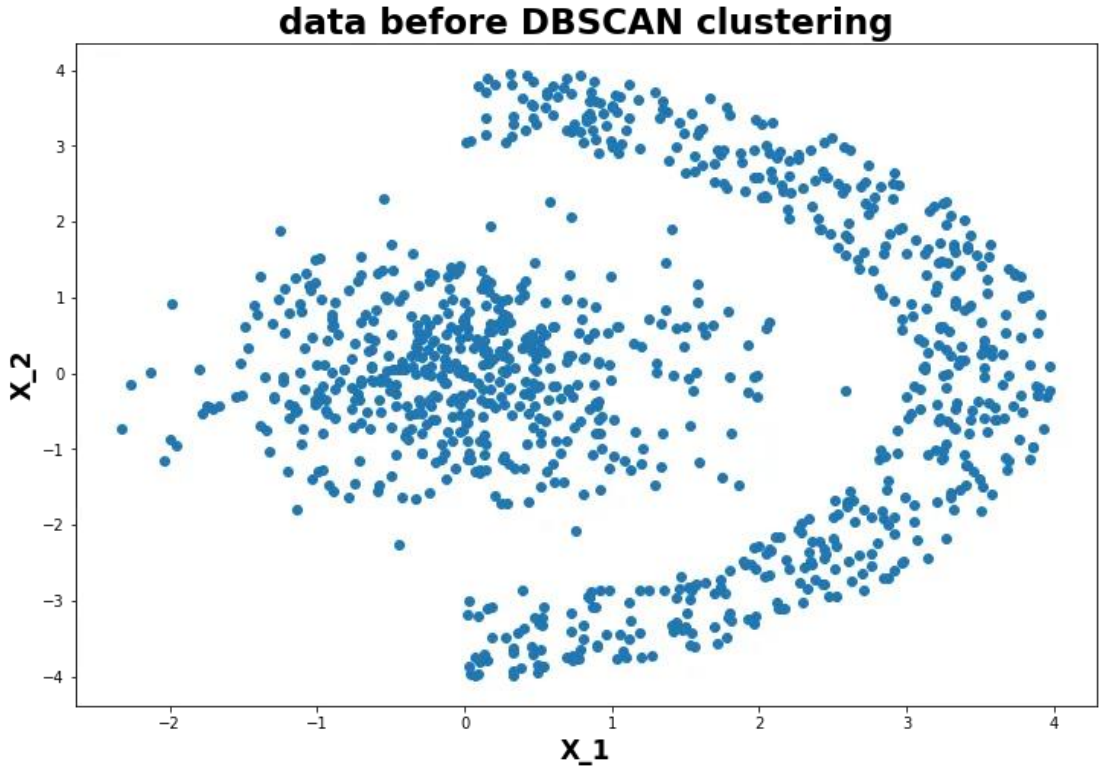


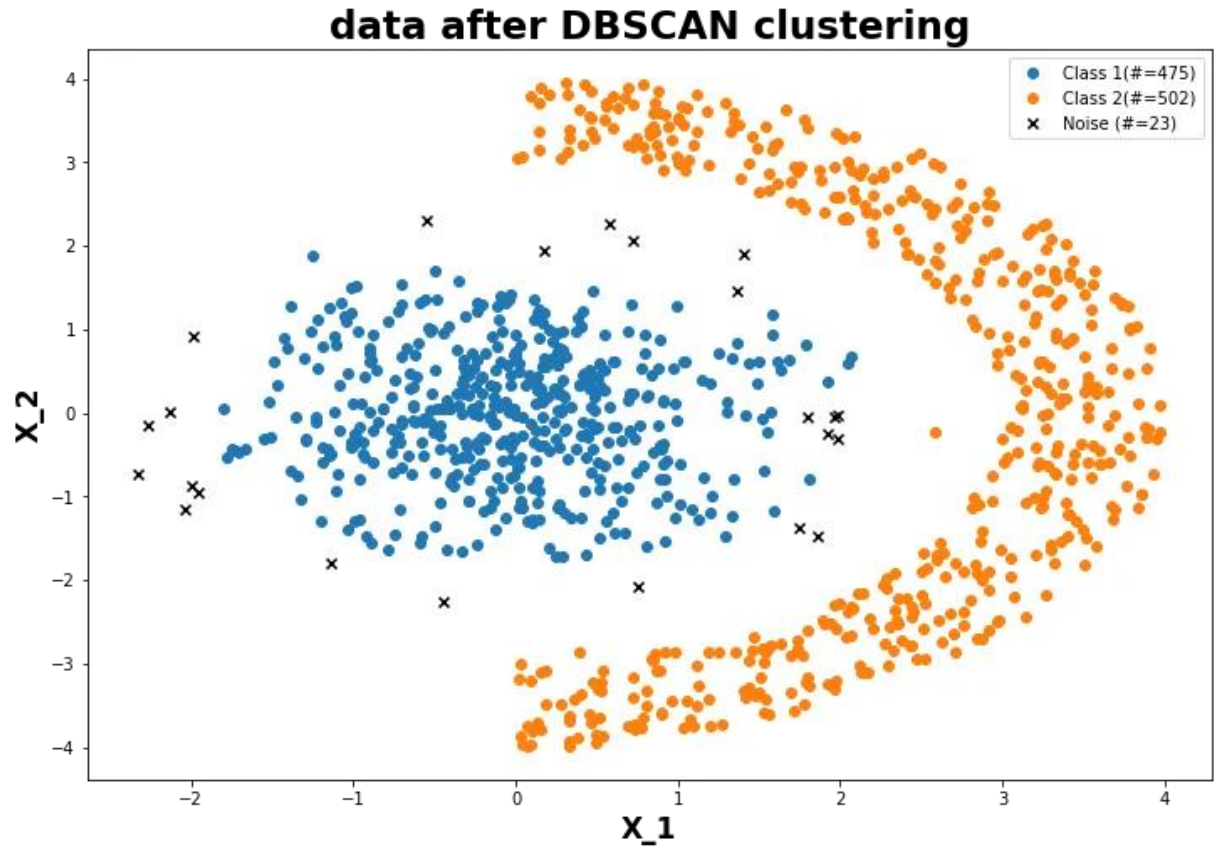
- 3. Noise Point : This type of data point has density < MinPts and, distance(z, cp) ≥ ε.
- Meaning that it doesn’t have the density required to be a core point and it is far away from any core point.
- Example : MinPts = 4, ε = 3 then z is a noise point



DBSCAN Advantages

- Identifies clusters of arbitrary shapes.
- Does not need the number of clusters to be specified.
- It has high efficiency on small and medium size datasets.
- It detects outliers and does not include them in the clusters


data before DBSCAN clustering
X_2
X_1


data after DBSCAN clustering
Class 1(#=475)
Class 2(#=502)
Noise (#=23)
X_2
X_1

DBSCAN Limitations

- The choice of the parameters can significantly affect the clustering results.
- It is an impractical algorithm for large datasets due to its complexity of $O(n^2)$.
- The clusters of our data need to have relatively similar density, otherwise the algorithm will struggle to come up with a satisfying result.
- In high-dimensional spaces points tend to be far away from each other which makes it difficult for the algorithm to find meaningful clusters.

When Should We Use DBSCAN Over K-Means In Clustering Analysis?

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and K-Means are both clustering algorithms that group together data that have the same characteristic.
- However, They work on different principles and are suitable for different types of data.
- We prefer to use DBSCAN when the data is not spherical in shape or the number of classes is not known beforehand.

DBSCAN	K-Means
In DBSCAN we need not specify the number of clusters.	K-Means is very sensitive to the number of clusters so it need to specified
Clusters formed in DBSCAN can be of any arbitrary shape.	Clusters formed in K-Means are spherical or convex in shape
DBSCAN can work well with datasets having noise and outliers	K-Means does not work well with outliers data. Outliers can skew the clusters in K-Means to a very large extent.
In DBSCAN two parameters are required for training the Model	In K-Means only one parameter is required is for training the model

Cluster Evaluation Metrics

Rand Index

Adjusted Rand Index

Silhouette Coefficient

Rand Index

- Rand index measures the outcome of a clustering algorithm by comparing the outcome with a known or expected outcome.
- It computes a similarity measure between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering's.
- The Rand index always takes on a value between 0 and 1 where:

  0: Indicates that two clustering methods do not agree on the clustering of any pair of elements.

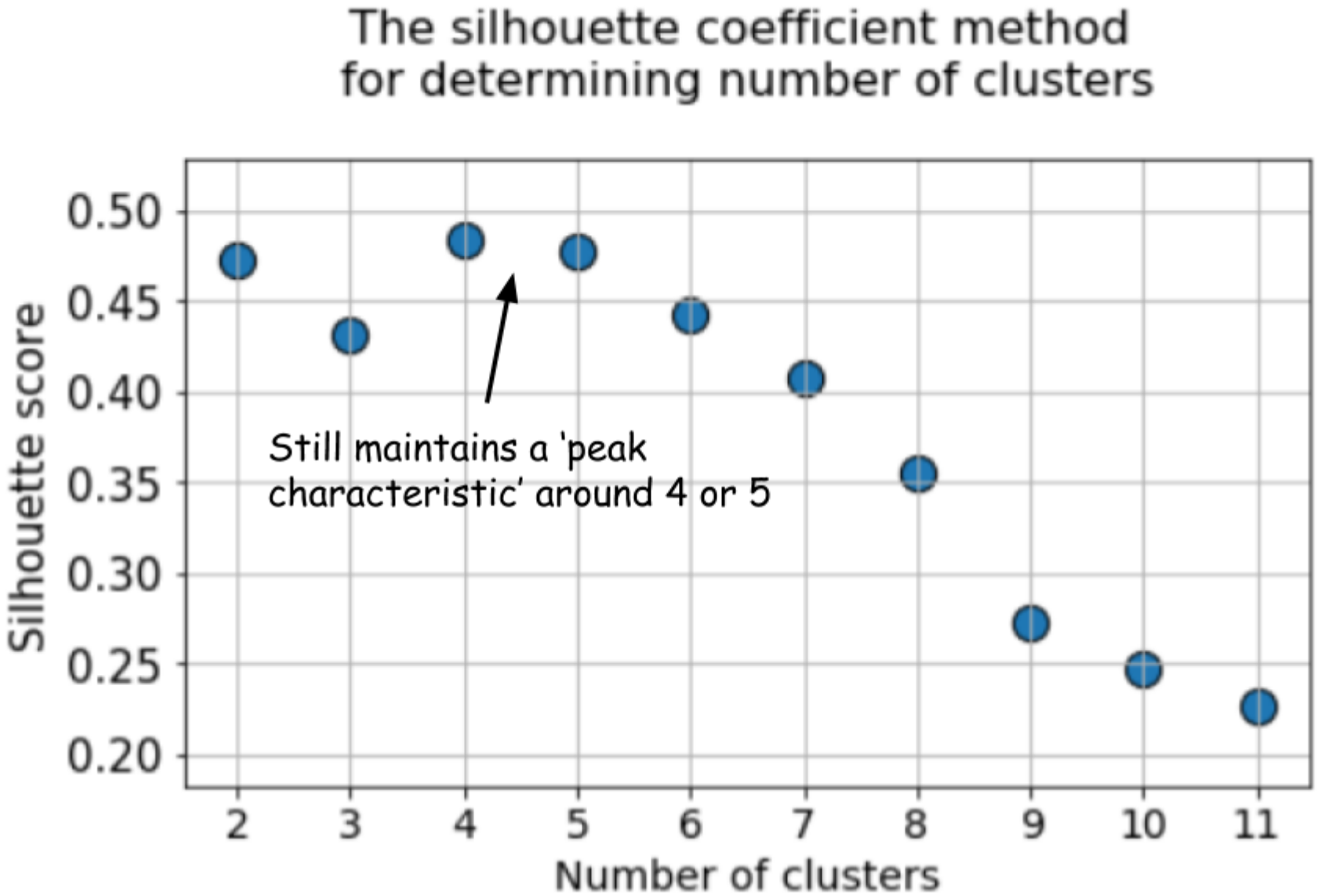
  1: Indicates that two clustering methods perfectly agree on the clustering of every pair of elements.

Adjusted Rand Index

- A major problem with the RI is that the expected value of Rand Index of two random cluster or partition does not take a constant value. To solve the problem , Adjusted Rand Index was introduced.
- The adjusted rand index is an evaluation metric that is used to measure the similarity between two clustering by considering all the pairs of the n_samples and calculating the counting pairs of the assigned in the same or different clusters in the actual and predicted clustering.
- Adjusted Rand Index range between -1 and 1.
- A score above 0.7 is considered to be a good match.

Silhouette Score/ Coefficient

- Silhouette's score is in the range of -1 to 1.
- A score near 1 denotes the best meaning that the data point is very compact within the cluster to which it belongs and far away from the other clusters.
- The worst value is -1.
- Values near 0 denote overlapping clusters.


The silhouette coefficient method
for determining number of clusters
Silhouette score
0.50
0.45
0.40
0.35
0.30
0.25
0.20
Still maintains a 'peak
characteristic' around 4 or 5
2
3
4
5
6
7
8
9
10
11
Number of clusters

When to use the Silhouette Coefficient?

- The Silhouette Coefficient is intuitive and easy to understand because you can use the Silhouette plots to visualize the results. Therefore the Silhouette Coefficient has a high interpretability
- Silhouette Coefficient has a value range of [-1, 1], from incorrect to highly dense clustering, with 0 overlapping clusters. The bounded range makes it easy to compare the scores between different algorithms.
- If you value well-defined clusters: The Silhouette Coefficient aligns with the common notion of good clusters, which are both dense and well-separated.

When not to use the Silhouette Coefficient?

- If you are evaluating various clustering approaches:
- The Silhouette Coefficient may give an advantage to density-based clustering methods, and thus, may not be an equitable comparison metric for other types of clustering algorithms.

Thanks

Samatrix Consulting Pvt Ltd