Machine Learning

(IT312)

Prepared by

Mr. Umesh B. Sangule

Assistant Professor

Unit-IV

CLUSTERING

Course Objectives : *To acquire the knowledge of Clustering techniques*

Course Outcome(CO4) : *Apply Clustering technique,*

Clustering

Clustering:

- Clustering is an important part of data cleaning, used in the field of artificial intelligence, deep learning, and data science.
- Distance metrics are the backbone of clustering.
- Distance metrics basically deal with finding the proximity or distance between data points and determining if they can be clustered together.

Distance Metrics:

- Distance Metrics are used in both supervised and unsupervised learning,
- Distance Metrics are used to calculate the similarity between data points,

Distance Metrics:

- We can calculate the distance between points and then define the similarity between them,

Properties of Distance Metrics:

1. Symmetry

If **x** and **y** are two points in a metric space, then the distance between **x** and **y** should be equal to the distance between **y** and **x**.

$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$$

Properties of Distance Metrics:

2. Non-negativity

Distances should always be non negative. Meaning it should be greater than or equal to zero.

$$d(\mathbf{x}, \mathbf{y}) \geq 0$$

The above inequality holds with equality (d(x,y) = 0) if and only if x and y denote the same point, i.e., x = y.

3. Triangle Inequality

Given three points x, y, and z, the distance metric should satisfy the triangle inequality:

$$d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$$

Distance Metrics:

- We can calculate the distance between points and then define the similarity between them,
- Types of Distance Metrics in Machine Learning:

1) Euclidean Distance
2) Manhattan Distance
3) Minkowski Distance
4) Hamming Distance

Distance Metrics

Euclidean Distance

- Euclidean Distance represents the shortest distance between two vectors.It is the square root of the sum of squares of differences between corresponding elements.
- We use use Euclidean distance when calculating the distance between two rows of data that have numerical values, such a floating point or integer values.
- If columns have values with differing scales, it is common to normalize or standardize the numerical values across all columns prior to calculating the Euclidean distance.

Distance Metrics

Euclidean Distance

- Euclidean distance is the shortest distance between any two points in a metric space. Consider two points x and y in a two-dimensional plane with coordinates (x1, x2) and (y1, y2), respectively.
- This distance is given by the square root of the sum of the squared differences between the corresponding coordinates of the two points. Mathematically, the Euclidean distance between the points x and y in two-dimensional plane is given by:

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

Distance Metrics

Computing Euclidean Distance in Python

- The Euclidean distance and the other distance metrics can be computed using convenience functions from the spatial module in SciPy.
- As a first step, let's import distance from Scipy's spatial module:

```
from scipy.spatial import distance
x = [3,6,9]
y = [1,0,1]
print(distance.euclidean(x,y))
```

Output >> 10.198039027185569

Distance Metrics

Computing Euclidean Distance:

Example 1: Find the Euclidean distance between data points P(3, 2) and Q(4, 1).

Given:P(3, 2) = (x1,y1) Q(4, 1) =(x2,y2)

Using Euclidean distance formula,

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

PQ = √[(4 – 3)2 + (1 – 2)2]

PQ = √[(1)2 + (-1)2]

PQ = √2 units.

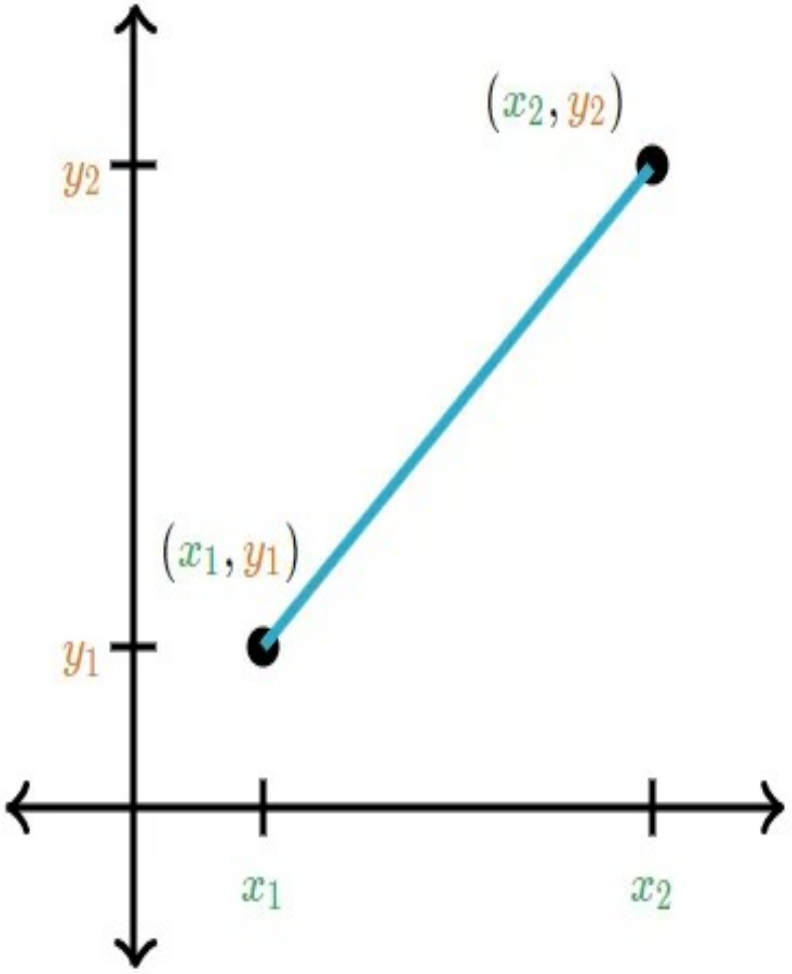
Distance Metrics

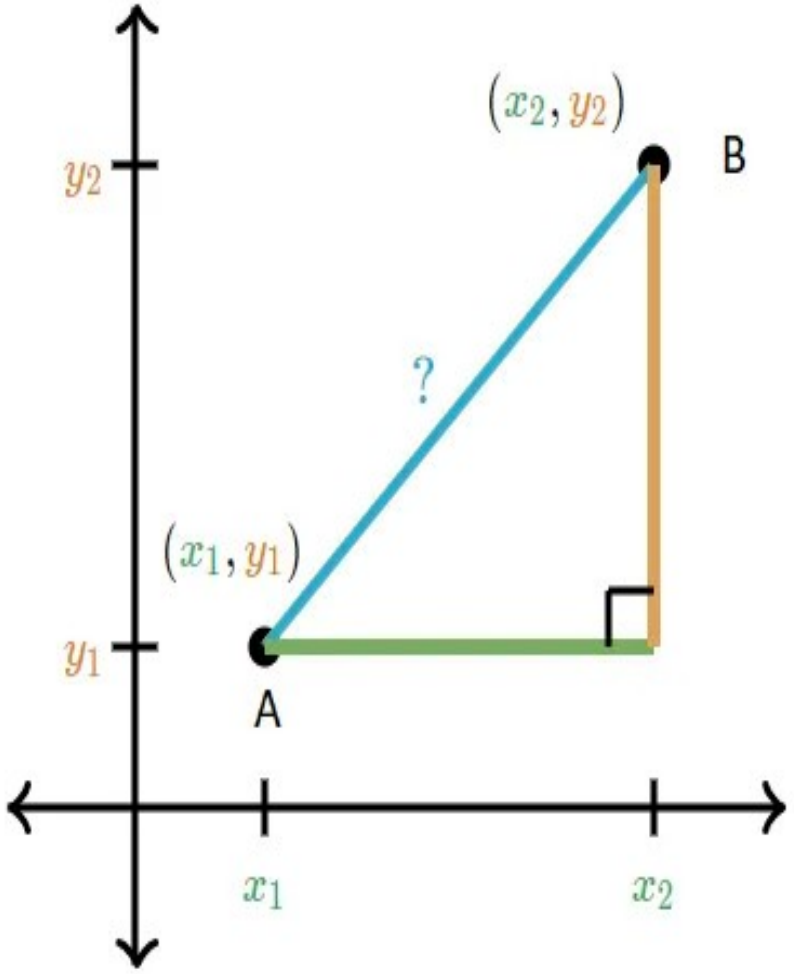
Manhatten Distance:

- The Manhattan distance, also called taxicab distance or cityblock distance, is another popular distance metric.
- Manhattan Distance is the sum of absolute differences between points across all the dimensions.
- Manhattan 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.
- It might make sense to calculate Manhattan distance instead of Euclidean distance for two vectors in an integer feature space.

Distance Metrics

Manhatten Distance:





Distance Metrics

Manhatten Distance:

- The Manhattan distance between the points x and y is given by:

Manhatten Distance= |(X2 – X1)| + |(Y2-Y1)|

- In n-dimensional space, where each point has n coordinates, the Manhattan distance is given by:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

Distance Metrics

Computing Manhattan Distance in Python

```
from scipy.spatial import distance

x = [3,6,9]

y = [1,0,1]

print(distance.cityblock(x,y))
```

Output >> 16

Distance Metrics

Computing Manhatten Distance:

Example 1: Find the distance between data points P(6, 4) and Q(8, 2).

Given:P(6, 4) = (x1,y1) Q(8, 2) =(x2,y2)

Using Manhatten distance formula,

$d = |(x2 - x1)| + |(y2 - y1)|$

$d = |(8 - 6)| + |(2 - 4)|$

$d = |2| + |-2|$

$d = 4$

Distance Metrics

Minkowski Distance

- 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.

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \quad for\ p \geq 1$$

- It is pretty straightforward to see that for **p = 1**, the Minkowski distance equation takes the same form as that of **Manhattan distance**.

- for **p = 2**, the Minkowski distance is equivalent to the **Euclidean distance,**

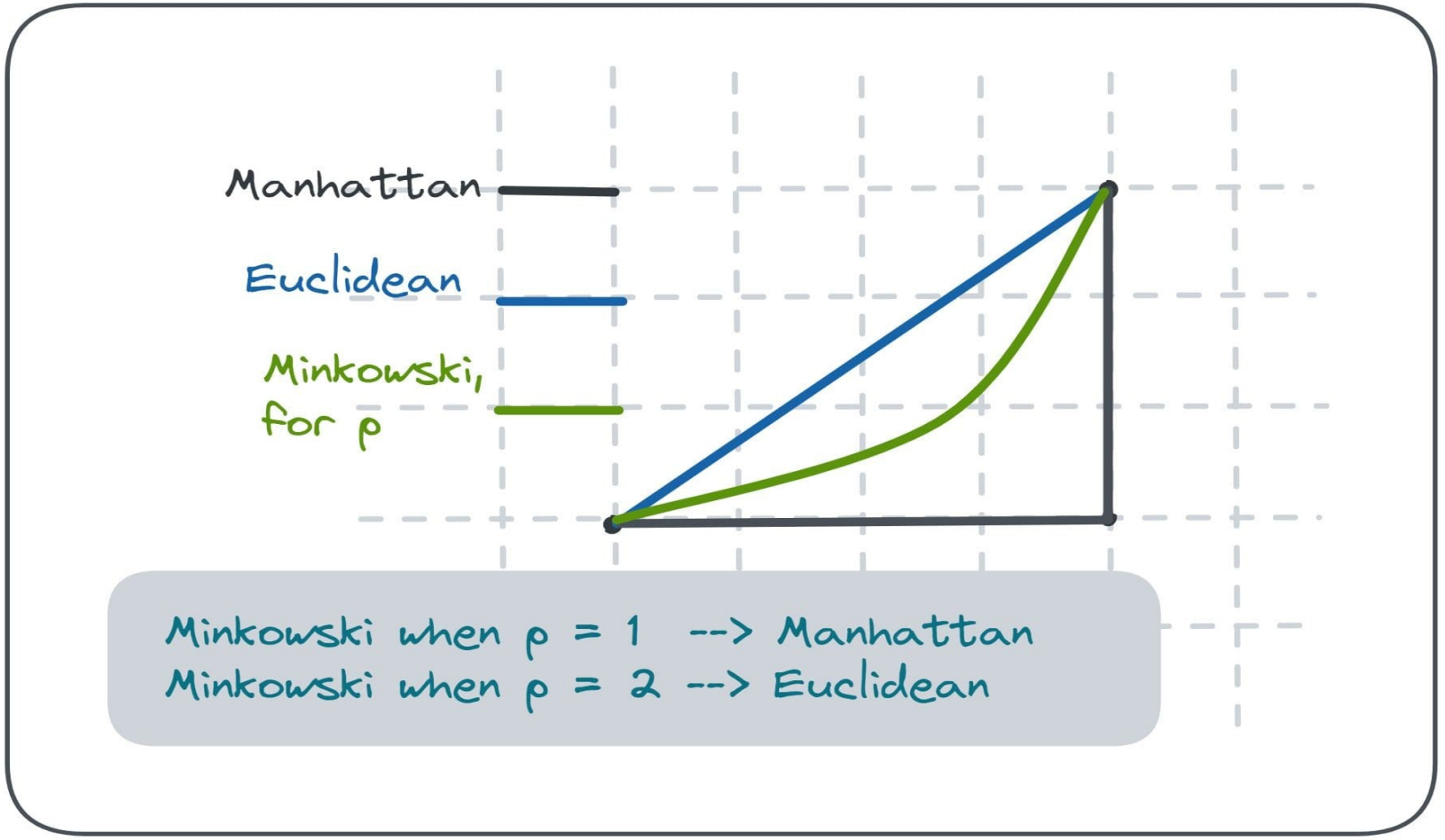
Distance Metrics

Computing Minkowski Distance in Python

```
from scipy.spatial import distance

x = [3,6,9]

y = [1,0,1]

print(distance.minkowski(x,y,p=1))

Output >> 16.0

print(distance.minkowski(x,y,p=2))

Output >> 10.198039027185569
```

Distance Metrics

Minkowski Distance:



Distance Metrics

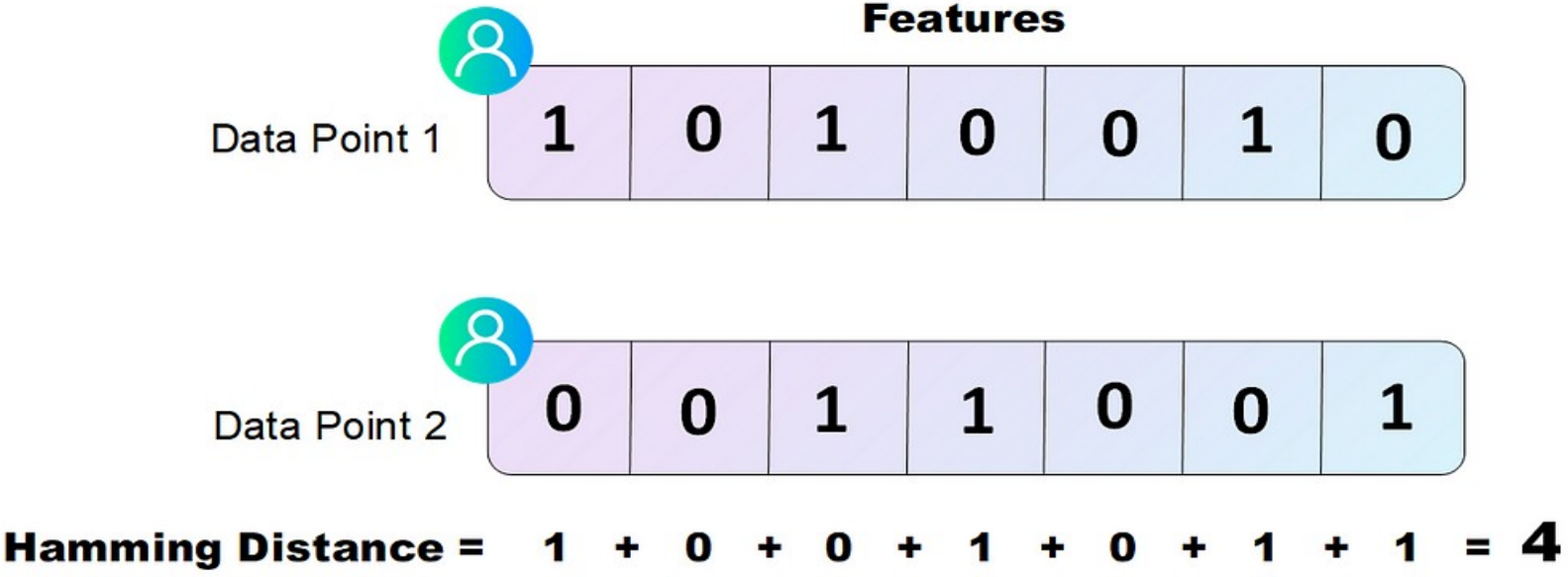
Hamming Distance:

- Hamming Distance measures the similarity between ***two strings of the same length.***
- The Hamming Distance between two strings of the same length is ***the number of positions at which the corresponding characters are different.***
- This specific metric is useful when calculating the distance in between observations for which we have ***only binary features***.
- This means that the Hamming distance is best calculated when all of the features of our data take either 0 or 1 as a value. It is basically a metric used to compare binary strings (understanding for strings here a row of binary features).

Distance Metrics

Hamming Distance:

- When we want to calculate the similarity in between two different users to see if they go in the same cluster, we can easily compute the Hamming Distance by counting the number of features which have different values, like shown in the following figure.



Distance Metrics

Hamming Distance:

Let’s understand the concept using an example. Let’s say we have two strings:

“euclidean” and **“manhattan”**

Since the length of these strings is equal, we can calculate the Hamming Distance. We will go character by character and match the strings. The first character of both the strings (e and m, respectively) is different. Similarly, the second character of both the strings (u and a) is different. and so on.

Look carefully – seven characters are different, whereas two characters (the last two characters) are similar:

euclidean and manhattan

Hence, the Hamming Distance here will be 7. Note that the larger the Hamming Distance between two strings, the more dissimilar those strings will be (and vice versa).

Hamming Distance in Python:

```
# defining two strings
string_1 = 'euclidean'
string_2 = 'manhattan'
# computing the hamming distance
hamming_distance = distance.hamming(list(string_1), list(string_2))*len(string_1)
print('Hamming Distance b/w', string_1, 'and', string_2, 'is: ', hamming_distance)
```

Hamming Distance b/w euclidean and manhattan is: 7.0

Distance Metrics

Hamming Distance in Python:

```
# calculating hamming distance between bit strings
from scipy.spatial.distance import hamming
# define data
row1 = [0, 0, 0, 0, 0, 1]
row2 = [0, 0, 0, 0, 1, 0]
# calculate distance
dist = hamming(row1, row2)
print(dist)
```

Clustering methods

Types of Clustering in Machine Learning:

Clustering broadly divides into two subgroups:

Hard Clustering: Each input data point either fully belongs to a cluster or not.

Soft Clustering: Rather than assigning each input data point to a distinct cluster, it assigns a probability or likelihood of the data point being in those clusters.

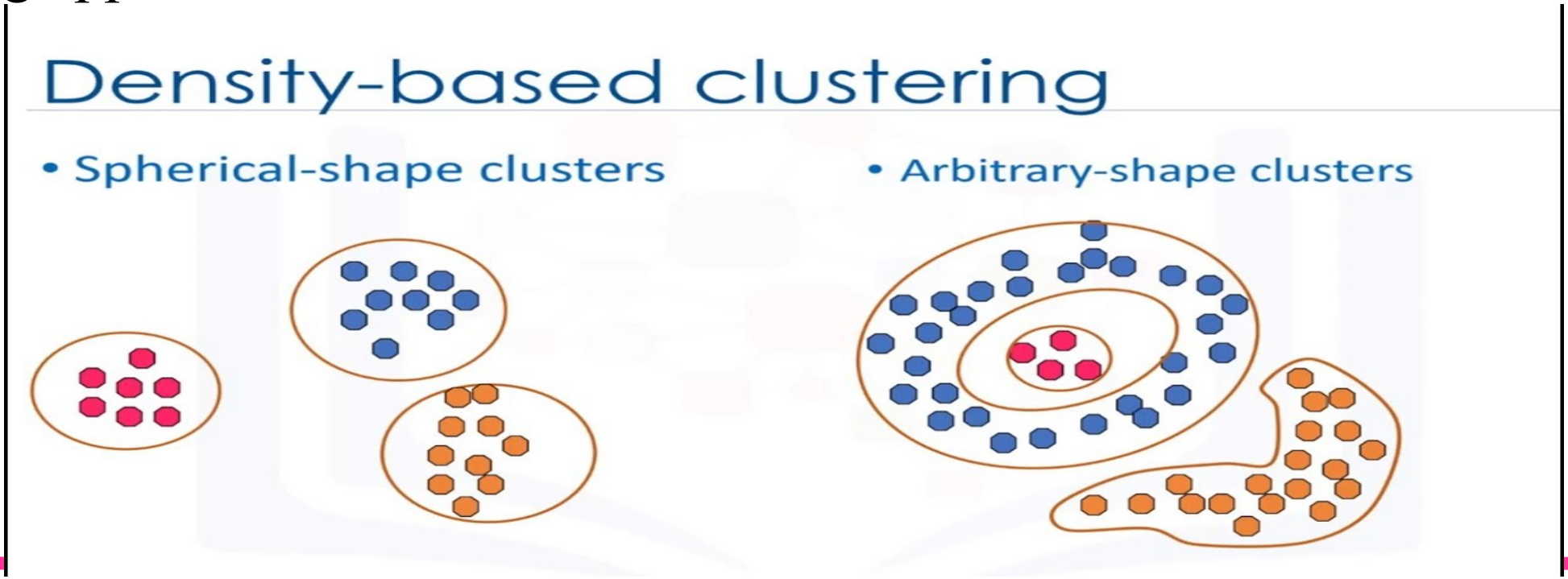
Different Types of Clustering Methods:

Density Clustering :Density-based clustering method considers density ahead of distance. Data is clustered by regions of high concentrations of data objects bounded by areas of low concentrations of data objects. The clusters formed are grouped as a maximal set of connected data points.

Clustering methods

Density Clustering :

- The clusters formed vary in arbitrary shapes and sizes and contain a maximum degree of homogeneity due to similar density.
- This clustering approach includes the noise and outliers in the datasets effectively.
-



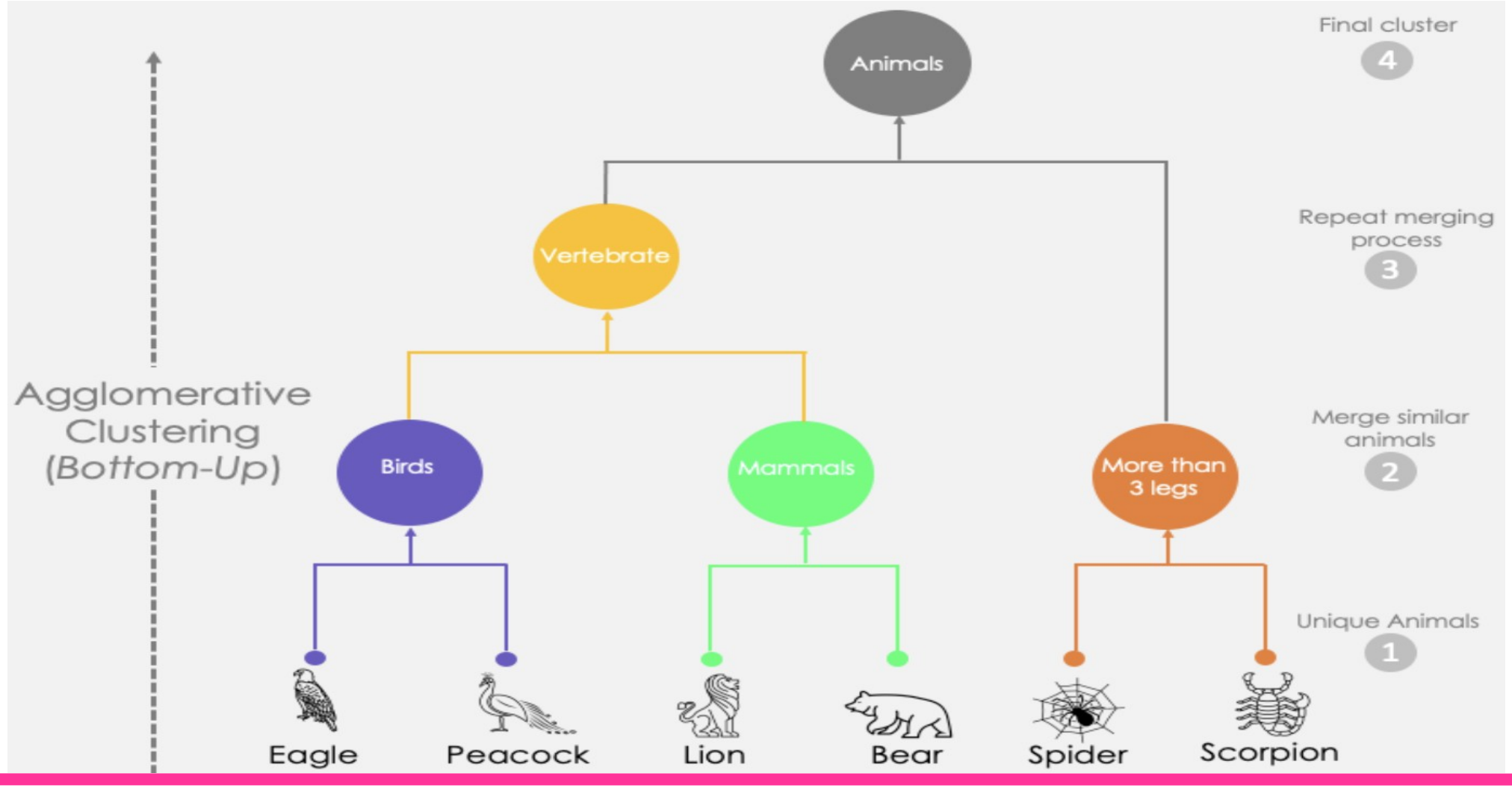
Clustering methods

Hierarchical Clustering:

- Hierarchical clustering, as the name suggests, is an algorithm that builds a **hierarchy of clusters**.
- This algorithm starts with all the data points assigned to a cluster of their own.
- Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left.
- The results of hierarchical clustering can be shown using a **dendrogram**.
- This algorithm can be implemented using a **bottom-up** approach or a **top-down** approach starting with all data points assigned in the same cluster and recursively performing splits till each data point is assigned a separate cluster.

Clustering methods

Hierarchical Clustering:



K-means clustering Algorithm

K-means clustering Algorithm:

- **Clustering** is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data.
- In clustering, we do not have a target to predict. We look at the data, try to club similar observations, and form different groups. Hence it is an unsupervised learning problem.
- K-means clustering is a widely used method for cluster analysis where the aim is to partition a set of objects into **K clusters** in such a way that the sum of the squared distances between the objects and their assigned cluster mean is **minimized.**
- k-means clustering divides data into a **predefined number of clusters**,

K-means clustering Algorithm:

Example

Let's try understanding this with a simple example. A bank wants to give credit card offers to its customers. Currently, they look at the details of each customer and, based on this information, decide which offer should be given to which customer.

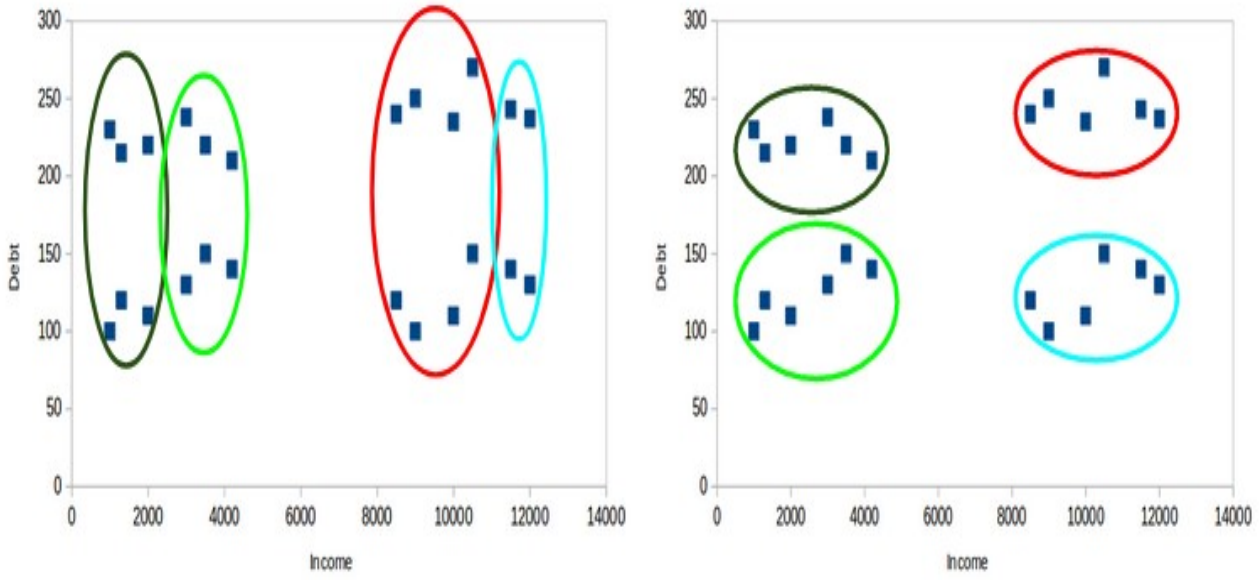
Now, the bank can potentially have millions of customers. Does it make sense to look at the details of each customer separately and then make a decision? Certainly not! It is a manual process and will take a huge amount of time.

So what can the bank do? One option is to segment its customers into different groups. For instance, the bank can group the customers based on their income:

K-means clustering Algorithm

Properties of Clusters:

- All the data points in a cluster should be similar to each other.
- The data points from different clusters should be as different as possible. This will intuitively make sense if you've grasped the above property.



Case - I Case - II

K-means clustering Algorithm

K-means clustering Algorithm:

- K-means clustering is a method for **grouping** n observations into **K clusters**.
- It uses vector quantization and aims to assign each observation to the cluster with the nearest mean or centroid, which serves as a prototype for the cluster.
- Originally developed for signal processing, K-means clustering is now widely used in machine learning to partition data points into K clusters based on their **similarity.**
- The goal is **to minimize the sum of squared distances between the data points** and their corresponding cluster centroids, resulting in clusters that are internally homogeneous and distinct from each other.

K-means clustering Algorithm

K-means clustering Algorithm:

- K-means is a centroid-based algorithm or a distance-based algorithm, where we calculate the distances to assign a point to a cluster.
- In K-Means, each cluster is associated with a **centroid.**
- ***The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.***
- Optimization plays a crucial role in the k-means clustering algorithm. The goal of the optimization process is to find the best set of centroids that minimizes the sum of squared distances between each data point and its closest centroid. ***This process is repeated multiple times until convergence, resulting in the optimal clustering solution.***

K-means clustering Algorithm

How to Apply K-Means Clustering Algorithm:

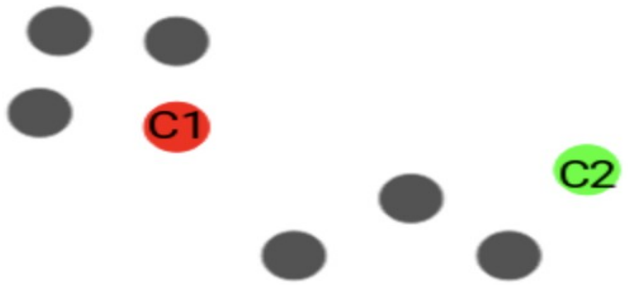
We have these 8 points, and we want to apply k-means to create clusters for these points. Here's how we can do it.

1. **Choose the number of clusters *k***

   The first step in k-means is to pick the number of clusters, k.

2. **Select k random points from the data as centroids**

   Next, we randomly select the centroid for each cluster. Let's say we want to have 2 clusters, so k is equal to 2 here. We then randomly select the centroid:



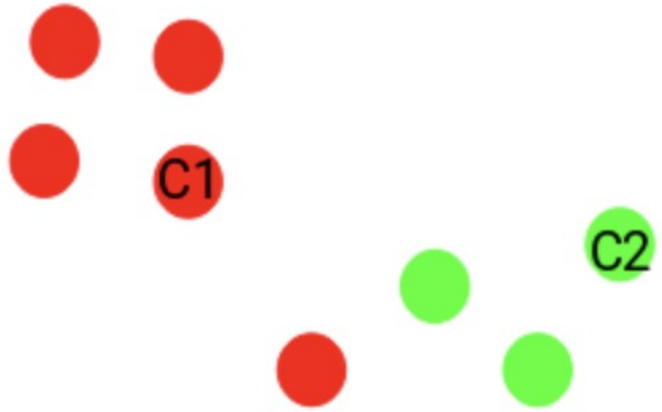
Here, the red and green circles represent the centroid for these clusters.

K-means clustering Algorithm

How to Apply K-Means Clustering Algorithm:

3. **Assign all the points to the closest cluster centroid**

Once we have initialized the centroids, we assign each point to the closest cluster centroid:



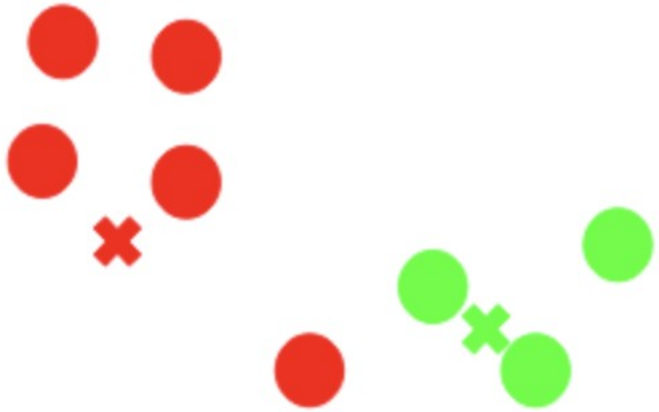
Here you can see that the points closer to the red point are assigned to the red cluster, whereas the points closer to the green point are assigned to the green cluster.

K-means clustering Algorithm

How to Apply K-Means Clustering Algorithm:

4. **Recompute the centroids of newly formed clusters**

Now, once we have assigned all of the points to either cluster, the next step is to compute the centroids of newly formed clusters:



Here, the red and green crosses are the new centroids.

K-means clustering Algorithm

How to Apply K-Means Clustering Algorithm:

5. **Repeat steps 3 and 4**

We then repeat steps 3 and 4:

The step of computing the centroid and assigning all the points to the cluster based on their distance from the centroid is a single iteration. But wait – when should we stop this process? It can't run till eternity, right?

K-means clustering Algorithm

How to Apply K-Means Clustering Algorithm:

Stopping Criteria for K-Means Clustering

There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

1. Centroids of newly formed clusters do not change
2. Points remain in the same cluster
3. Maximum number of iterations is reached

We can stop the algorithm if the centroids of newly formed clusters are not changing. Even after multiple iterations, if we are getting the same centroids for all the clusters, we can say that the algorithm is not learning any new pattern, and it is a sign to stop the training.

Another clear sign that we should stop the training process is if the points remain in the same cluster even after training the algorithm for multiple iterations.

K-means clustering Algorithm

How to Apply K-Means Clustering Algorithm:

Stopping Criteria for K-Means Clustering

There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

1. Centroids of newly formed clusters do not change
2. Points remain in the same cluster
3. Maximum number of iterations is reached

We can stop the algorithm if the centroids of newly formed clusters are not changing. Even after multiple iterations, if we are getting the same centroids for all the clusters, we can say that the algorithm is not learning any new pattern, and it is a sign to stop the training.

Another clear sign that we should stop the training process is if the points remain in the same cluster even after training the algorithm for multiple iterations.

K-means clustering Algorithm

K-Means Clustering Algorithm:

Advantages of k-means

1. Simple and easy to implement: The k-means algorithm is easy to understand and implement, making it a popular choice for clustering tasks.
2. Fast and efficient: K-means is computationally efficient and can handle large datasets with high dimensionality.
3. Scalability: K-means can handle large datasets with a large number of data points and can be easily scaled to handle even larger datasets.
4. Flexibility: K-means can be easily adapted to different applications and can be used with different distance metrics and initialization methods.

K-means clustering Algorithm

K-Means Clustering Algorithm:

Disadvantages of K-Means:

1. Sensitivity to initial centroids: K-means is sensitive to the initial selection of centroids and can converge to a suboptimal solution.
2. Requires specifying the number of clusters: The number of clusters k needs to be specified before running the algorithm, which can be challenging in some applications.
3. Sensitive to outliers: K-means is sensitive to outliers, which can have a significant impact on the resulting clusters.

K-means clustering Algorithm

K-Means Clustering Algorithm:

Applications of K-Means Clustering

K-Means clustering is used in a variety of examples or business cases in real life, like:

- Academic performance
- Diagnostic systems
- Search engines
- Wireless sensor networks

Academic Performance

Based on the scores, students are categorized into grades like A, B, or C.

K-means clustering Algorithm

K-Means Clustering Algorithm Applications:

Diagnostic systems

The medical profession uses k-means in creating smarter medical decision support systems, especially in the treatment of liver ailments.

Search engines

Clustering forms a backbone of search engines. When a search is performed, the search results need to be grouped, and the search engines very often use clustering to do this.

Wireless sensor networks

The clustering algorithm plays the role of finding the cluster heads, which collect all the data in its respective cluster.

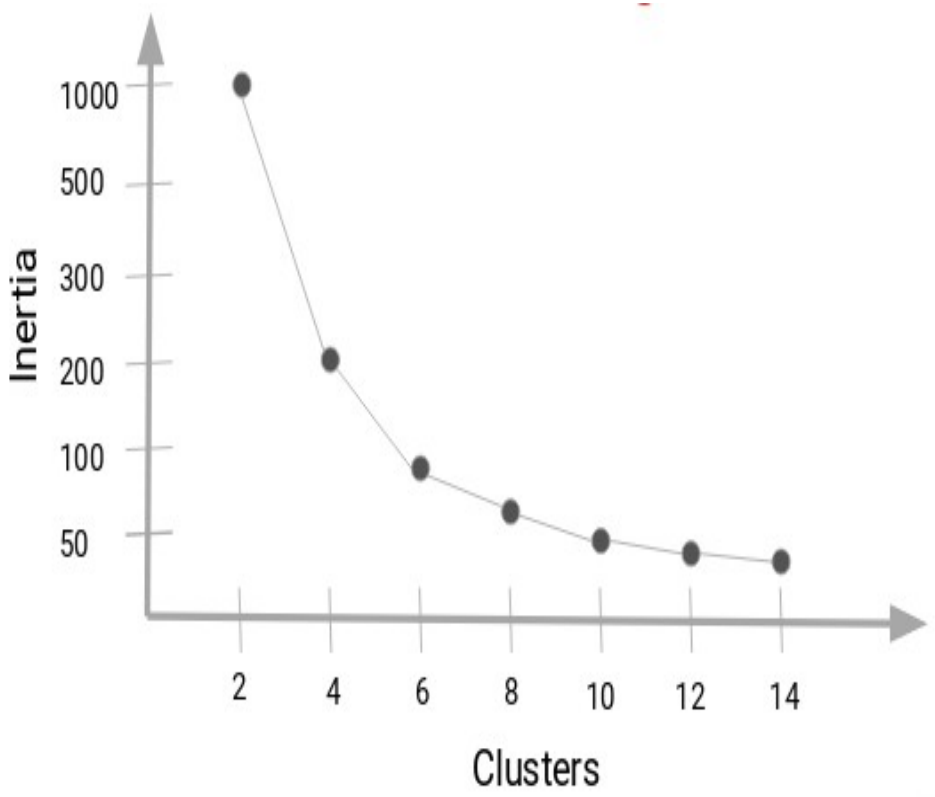
K-means clustering Algorithm

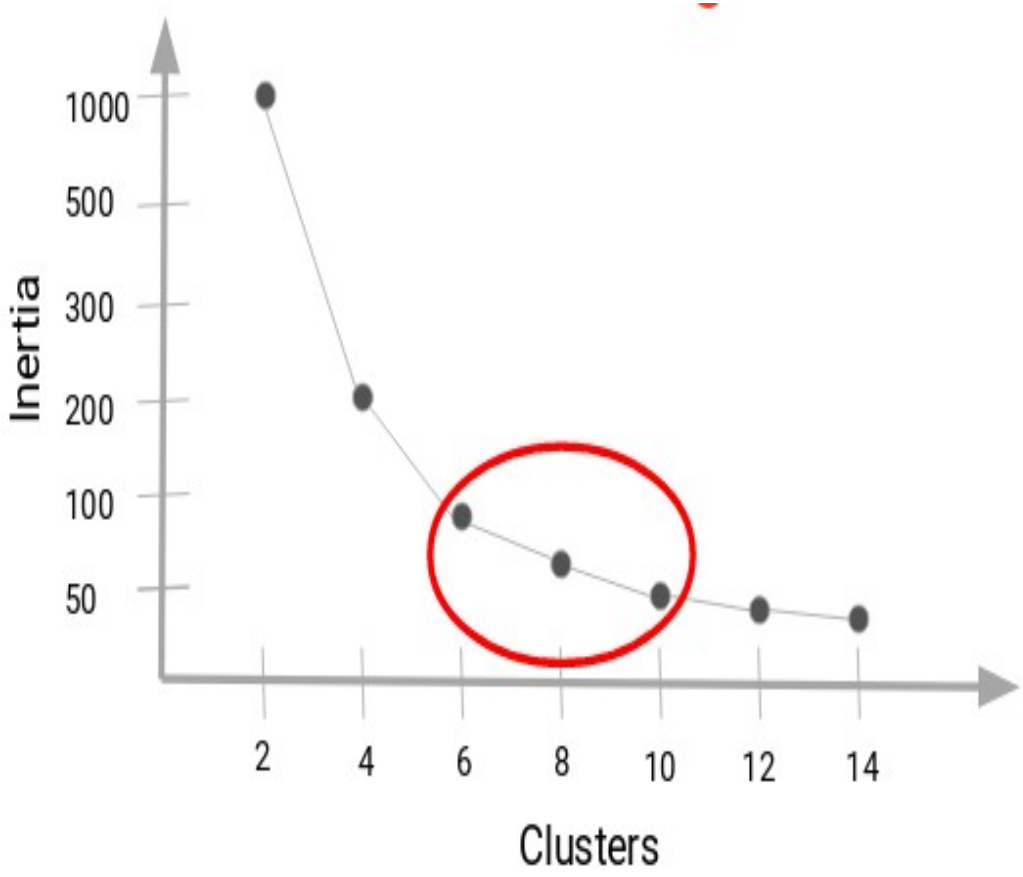
How to Choose the Right Number of Clusters in K-Means Clustering?

- The maximum possible number of clusters will be equal to the number of observations in the dataset.
- But then, how can we decide the optimum number of clusters? One thing we can do is plot a graph, also known as an **elbow curve**, where the x-axis will represent the **number of clusters** and the y-axis will be an **evaluation metric**.
- we will increase the number of clusters, train the model again, and plot the inertia value.
- The cluster value where this decrease in inertia value becomes constant can be chosen as the right cluster value for our data.

K-means clustering Algorithm

How to Choose the Right Number of Clusters in K-Means Clustering?





k-medoid Clustering algorithm

- K-Medoids is an unsupervised clustering algorithm using the **partition method of clustering.**
- It is an improvised version of the K-Means clustering algorithm especially used **to handle outlier data**. It requires unlabeled data to work with.
- In the K-Medoids algorithm, **each data point** is called a **medoid**. The medoids serve as **cluster centers**.
- The medoid is a point such that its sum of the distance from all other points in the same cluster is minimum. For distance, any suitable metric like Euclidian distance or Manhattan distance can be used.

k-medoid Clustering algorithm

➢ *K-Medoids Clustering Algorithm:*

1) First, we select K random data points from the dataset and use them as **medoids.**

2) Now, we will calculate the **distance** of each data point from the medoids. You can use any of the Euclidean, Manhattan distance,

3) Once we find the distance of each data point from the medoids, we will assign the data points to the clusters associated with each medoid. The data points are assigned to the medoids at the closest distance.

4) After determining the clusters, we will calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. Let the cost be Ci.

k-medoid Clustering algorithm

➢ ***K-Medoids Clustering Algorithm:***

5) Now, we will select a random data point Dj from the dataset and swap it with a medoid Mi. Here, Dj becomes a temporary medoid. After swapping, we will calculate the distance of all the non-medoid data points to the current medoid of each cluster. Let this cost be Cj.

6) If Ci>Cj, the current medoids with Dj as one of the medoids are made permanent medoids. Otherwise, we undo the swap, and Mi is reinstated as the medoid.

7) Repeat 4 to 6 until no change occurs in the clusters.

k-medoid Clustering algorithm

➢ ***K-Medoids Clustering Algorithm:***

➢ The K-Medoid is applied in such a way that:

- A single point can belong to only one cluster
- Each cluster has a minimum one point,

k-medoid Clustering algorithm

Example

Let us consider the following set of data. We will take k=2 and the distance formula to be used

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

Sl. no	x	y
1	9	6
2	10	4
3	4	4
4	5	8
5	3	8
6	2	5
7	8	5
8	4	6
9	8	4
10	9	3

k-medoid Clustering algorithm

M1 = (8, 4)

M2 = (4, 6) **and**

M1 = (8,5)

M2 = (4,6)

For k = 2 let us take two random points P1(8,4) and P2(4,6) and calculate thie distances from other points.

Sl. no	x	y	Dist from P1 (8,4)	Dist from P2 (4,6)
1	9	6	3	5
2	10	4	2	8
3	4	4	4	2
4	5	8	7	3
5	3	8	9	3
6	2	5	7	3
7	8	5	1	5
8	4	6	-	-
9	8	4	-	-
10	9	3	2	8

k-medoid Clustering algorithm

M1 = (3, 4)

M2 = (7, 3)

and

M1 = (3, 4)

M2 = (7, 4)

Point	Coordinates
A1	(2, 6)
A2	(3, 8)
A3	(4, 7)
A4	(6, 2)
A5	(6, 4)
A6	(7, 3)
A7	(7,4)
A8	(8, 5)
A9	(7, 6)
A10	(3, 4)

k-medoid Clustering algorithm

➢ *Applications of K-Medoids Clustering in Machine Learning*

- E-commerce websites, supermarkets, and malls use transaction data for customer segmentation based on buying behavior. It helps them to increase sales by recommending products according to user behavior using recommendation systems.
- K-Medoids clustering can also be used in cyber profiling. It involves grouping people based on their connection to each other. By collecting usage data from individual users, you can cluster them into groups for profiling.
- By grouping similar pixels into clusters, you can also perform image segmentation using k-medoids clustering.
- You can also perform fraud detection on banking and insurance data using k-medoids clustering. By using historical data on frauds, you can find probable fraudulent transactions from a given dataset.
- You can also use k-medoids clustering in various other tasks such as social media profiling, ride-share data analysis, identification of localities with a high rate of crime, etc.

k-medoid Clustering algorithm

➢ *Advantages of K-Medoids Clustering*

- K-Medoids clustering is guaranteed to converge. Hence, we are guaranteed to get results when we perform k-medoids clustering on any dataset.
- K-Medoids clustering doesn't apply to a specific domain. Owing to the generalization, we can use k-medoids clustering in different machine learning applications ranging from text data to Geo-spatial data and financial data to e-commerce data.
- The medoids in k-medoids clustering are selected randomly. We can choose the initial medoids in a way such that the number of iterations can be minimized. For improving the performance of the algorithm, you can warm start the choice of medoids by selecting specific data points as medoids after data analysis.
- Compared to other partitioning clustering algorithms such as K-median and k-modes clustering, the k-medoids clustering algorithm is faster in execution.
- K-Medoids clustering algorithm is very robust and it effectively deals with outliers and noise in the dataset. Compared to k-means clustering, the k-medoids clustering algorithm is a better choice for analyzing data with significant noise and outliers.

KNN (K-Nearest Neighbor) Algorithm

KNN (K-Nearest Neighbour) Algorithm:

- The K-Nearest Neighbors (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values.

- During the training phase, the KNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training examples, using a chosen distance metric such as **Euclidean distance**.

- K represents the number of nearest neighbors that needs to be considered while making prediction.

KNN (K-Nearest Neighbour) Algorithm

KNN (K-Nearest Neighbour) Algorithm:

- In the case of classification, the algorithm assigns the most common class label among the K neighbours as the predicted label for the input data point.
- For regression, it calculates the average or weighted average of the target values of the K neighbours to predict the value for the input data point.
- KNN Algorithm can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease of interpreting output 2. Calculation time 3. Predictive Power

KNN (K-Nearest Neighbor) Algorithm

KNN (K-Nearest Neighbour) Algorithm:

	Logistic Regression	CART	Random Forest	KNN
1. Ease to interpret output	2	3	1	3
2. Calculation time	3	2	1	3
3. Predictive Power	2	2	3	2

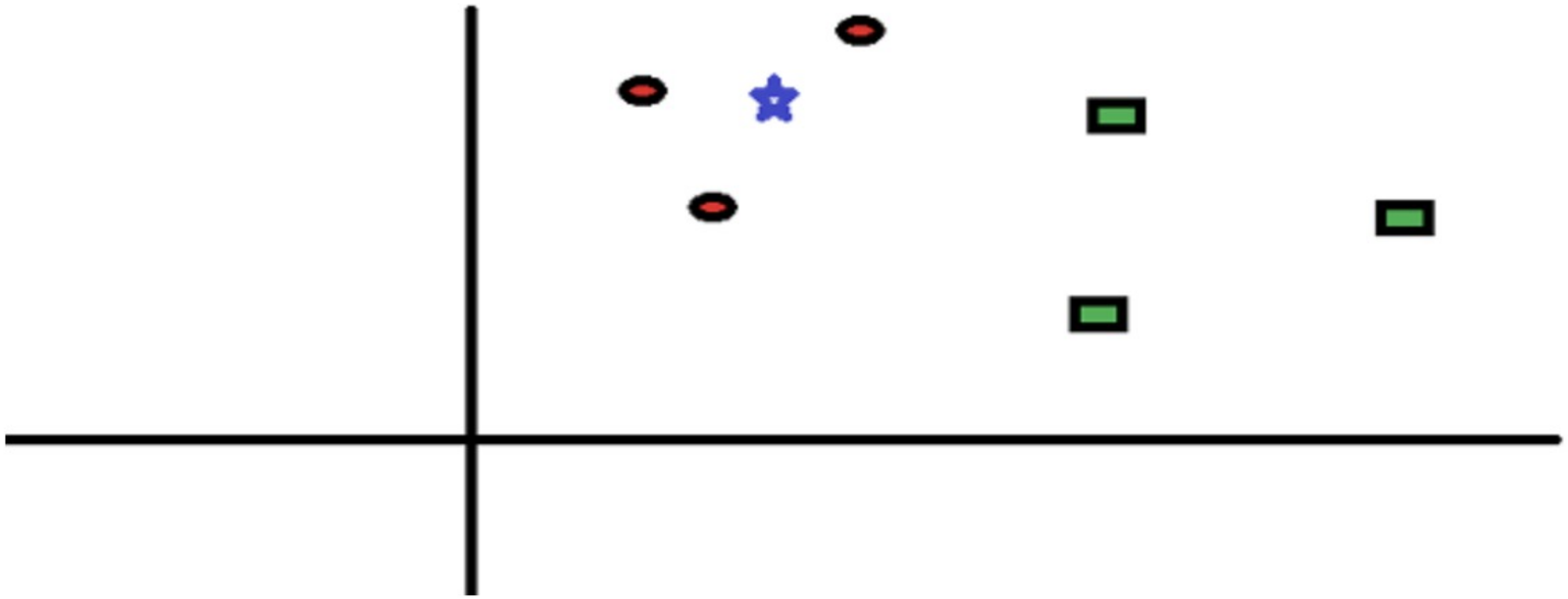
- KNN Algorithm can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease of interpreting output 2. Calculation time 3. Predictive Power

KNN (K-Nearest Neighbor) Algorithm

How Does the KNN Algorithm Work?

Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS):

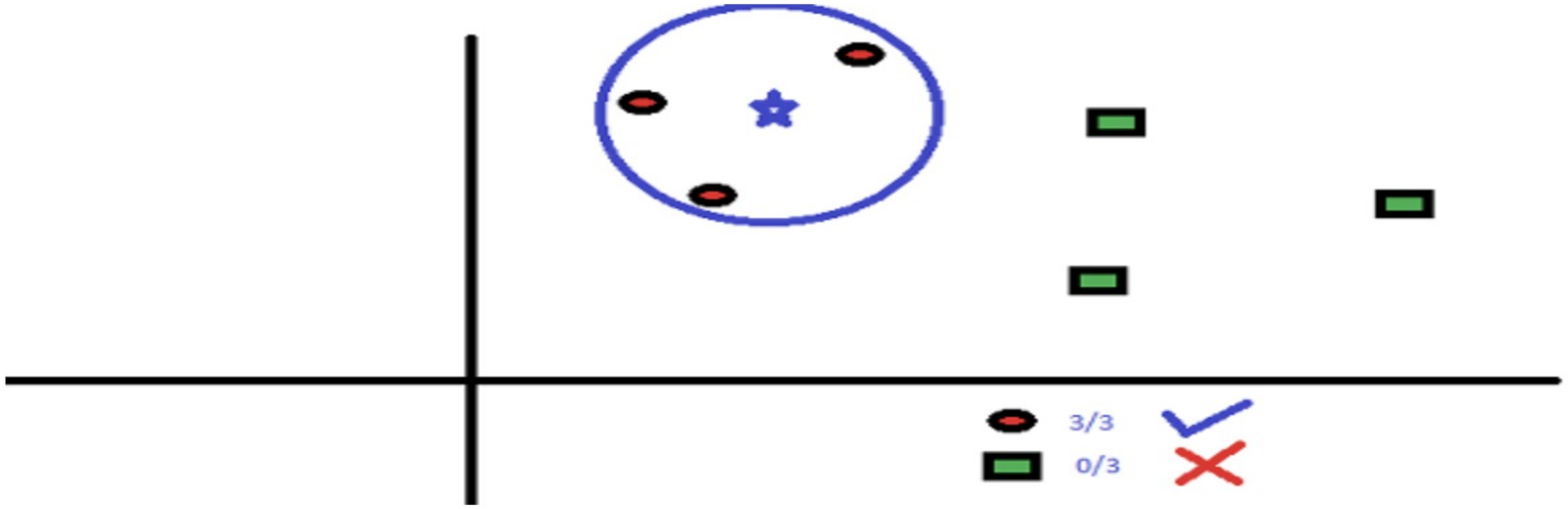


You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else.

KNN (K-Nearest Neighbor) Algorithm

How Does the KNN Algorithm Work?

The “K” in KNN algorithm is the nearest neighbor we wish to take the vote from. Let’s say K = 3. Hence, we will now make a circle with BS as the center just as big as to enclose only three data points on the plane. Refer to the following diagram for more details:



The three closest points to BS are all RC. Hence, with a good confidence level, we can say that the BS should belong to the class RC. Here, the choice became obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm.

KNN (K-Nearest Neighbor) Algorithm

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    - Calculate the distance between test data and each row of training dataset. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other distance function or metrics that can be used are Manhattan distance, Minkowski distance, Chebyshev, cosine, etc. If there are categorical variables, hamming distance can be used.
    - Sort the calculated distances in ascending order based on distance values
    - Get top k rows from the sorted array
    - Get the most frequent class of these rows
    - Return the predicted class

KNN (K-Nearest Neighbor) Algorithm

Advantages of the KNN Algorithm

- **Easy to implement** as the complexity of the algorithm is not that high.
- **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

KNN (K-Nearest Neighbor) Algorithm

Disadvantages

- It fails when variables have different scales.
- It is difficult to choose K-value.
- It leads to ambiguous interpretations.
- It is sensitive to outliers and missing values.
- Does not work well with large datasets.
- It does not work well with high dimensions.

KNN (K-Nearest Neighbor) Algorithm

KNN Overview:

- KNN classifier operates by finding the k nearest neighbors to a given data point, and it takes the majority vote to classify the data point.
- The value of k is crucial, and one needs to choose it wisely to prevent overfitting or underfitting the model.
- One can use cross-validation to select the optimal value of k for the k-NN algorithm, which helps improve its performance and prevent overfitting or underfitting. Cross-validation is also used to identify the outliers before applying the KNN algorithm.

KNN (K-Nearest Neighbor) Algorithm

Performance Measure:

Rand Index

Another commonly used metric is the Rand Index. 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 clusterings.

The formula of the Rand Index is:

$$RI = \frac{\text{Number of Agreeing Pairs}}{\text{Number of Pairs}}$$

The RI can range from zero to 1, a perfect match.

KNN (K-Nearest Neighbor) Algorithm

Performance Measure:

Adjusted Rand Index

Rand index adjusted for chance.

The Rand Index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings.

The raw RI score is then “adjusted for chance” into the ARI score using the following scheme:

$$ARI = \frac{\text{RI - Expected RI}}{\text{Max(RI) - Expected RI}}$$

The Adjusted Rand Index, similarly to RI, ranges from zero to one, with zero equating to random labelling and one when the clusters are identical.


THANK YOU