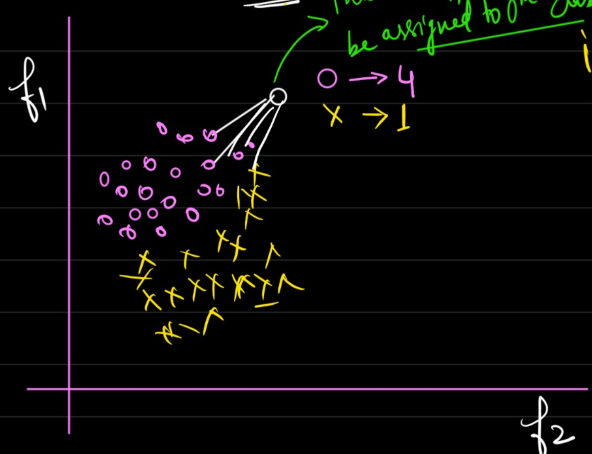
K-Nearest Neighbour (KNN)

① KNN Classifier
② KNN Regressor

idea:- You will be like people surrounding you

① KNN Classifier

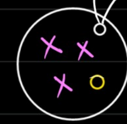
f_1	f_2	y
—	—	0
—	—	1
—	—	0
—	—	0
—	—	1



This new dp will be assigned to 0th class

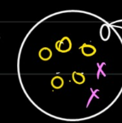
$0 \rightarrow 4$
$X \rightarrow 1$

Sc-1



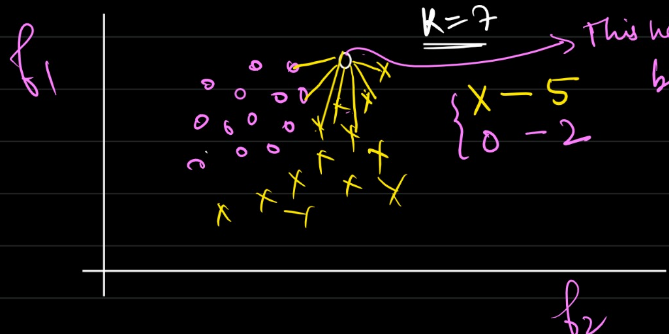
new ball → This ball is of red colour.

Sce-2



new ball → This ball is of yellow colour.

in both of the cases you have considered the majority colour ball in the bag as deciding factor.



This new dp will be assigned X class.

$\begin{cases} X - 5 \\ 0 - 2 \end{cases}$

→ As K changes the class of new dp might also change.

⇓

K is a hyperparameter

① plot the datapoints in n-d space

② Initialise the K value (No of Neighbours you want to consider)

$K \in 1 \rightarrow \infty$

$K > 0$

(Generally taken as $K > 3$)

→ Calculate the distance of new datapoint wrt to all dp's.

→ Sort the distance.

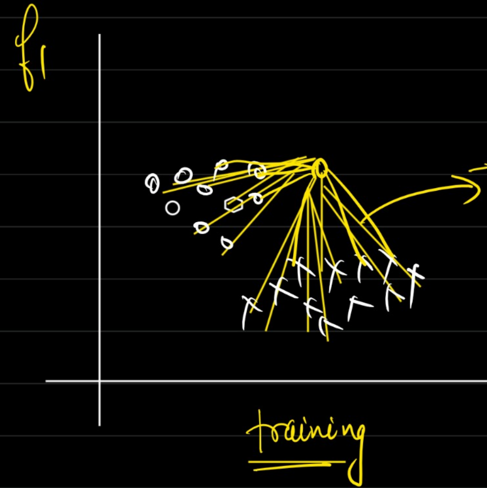
→ based on K find the class of that K nearest datapoint.

③ Find the mode of the class

Corner Case

$K \rightarrow$ odd $\rightarrow 1, 3, 5, 7$

④ Assign the class.



→ distance of new datapoint from all the dp's
→ Sort
→ Select K nearest distance dp
→ Take mode & assign the class.

K=3

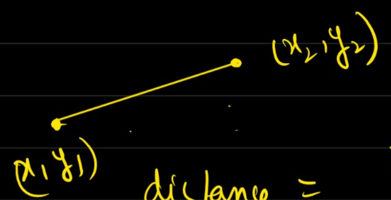
f1	f2		dist
		0	58
		1	60
		0	40
		1	
		0	

dist	class
40	0
58	1
60	1

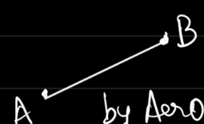
majority
Predicted class = 1

To calculate distance

① Euclidean distance

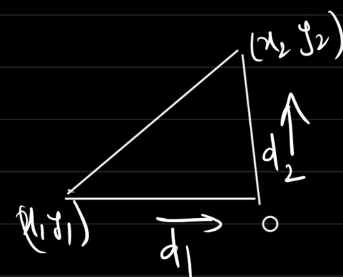


$$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

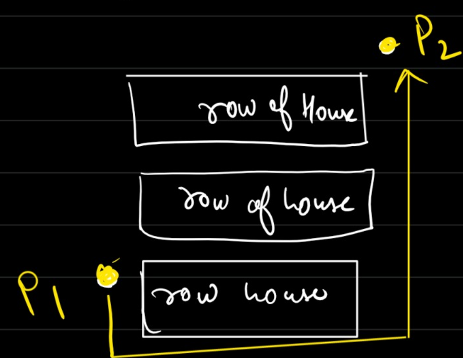


by Aeroplane → Euclidean distance.

② Manhattan distance



$$\text{distance} = d_1 + d_2$$

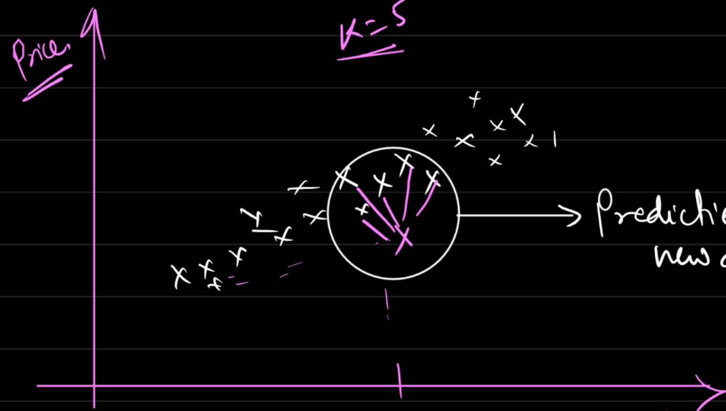


* K – hyperparameter

* for different K, you keep track of train/test accuracy.

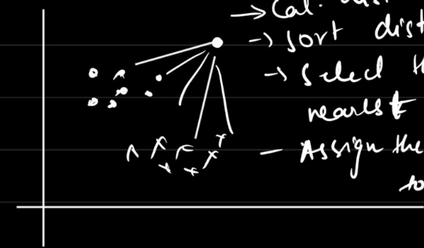
K-Nearest Neighbors (KNN) is a simple machine learning method. For each new data point, it looks at the 'K' closest points from the training data. If most of the closest points are of a certain class, the new point is classified the same way. For predictions, it takes the average value of the closest points. It's easy to understand and use but can be slow with lots of data and can struggle with too many features or noise.

② Regression



→ Prediction for new dp will be avg y(Price) of nearest 5 dp's

Conclusion of Classification



→ In case of outlier, instead of Avg you can take median.

* Advantage

→ Easy to Understand | very intuitive.
→ Performance of model in terms of evaluation metric is good

* disadvantage.

→ Lazy learner
  → At the real time the distance of test data from each of dp is calculated ⇒ Brute force.

→ n dp ⇒ n-1 distance you will be calculating

→ Computationally expensive

Time complexity O(N)

ML models

- eager learner: (All the model parameters are calculated while training and used for prediction)
- Lazy learner ⇓ you are calculating Parameters of model in real time

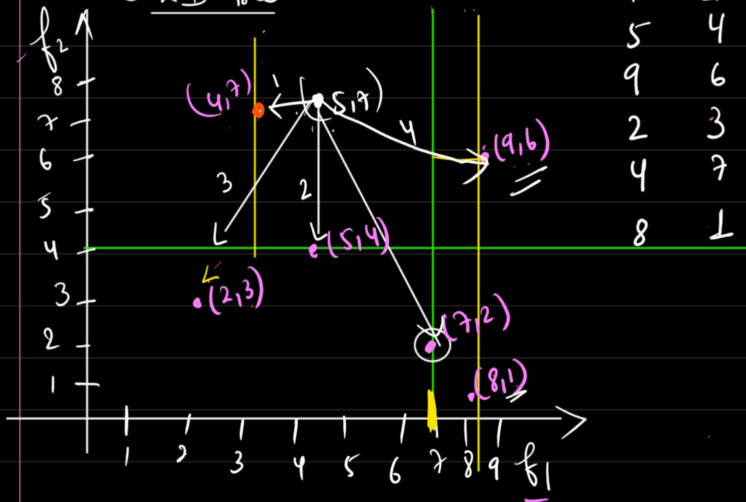
* Variants of KN

① KD Tree
② Ball tree } optimizers

Variants of KNN

① KD Tree (K-dimensional tree)
② Ball Tree

① KD tree



f1	f2
7	2
5	4
9	6
2	3
4	7
8	1

→ Distance calculation from each of dp of the test data is to be Calculated ⟹ computationally expensive.

* KD tree

↳ partition the data in a binary tree.

Step 1 Sort the feature f1 & f2
Step 2 Calculate the median of f1 & f2
Step 3 – Partition the data based on median
Step 4 – Partition all the data recursively in each of sub partition.

① f1 – 2, 4, 5, 7, 8, 9 | f2 = 1, 2, 3, 4, 6, 7

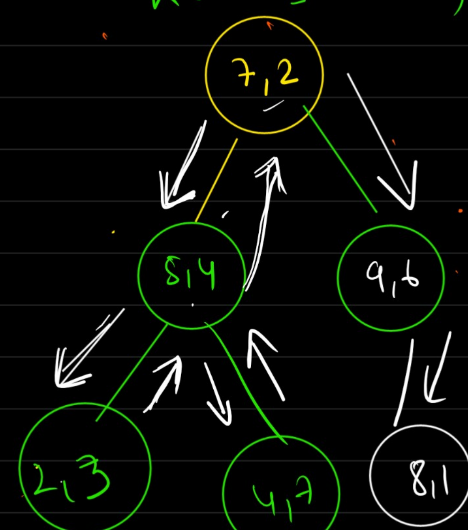
② median → ~~$\frac{5+7}{2}$ 6.5~~ take 7 as median. | take 4 as median.

③ Partition of data based on median.

④ recursively → (9, 6), (8, 1)

$f_1 = 8, 9 \longrightarrow 9$

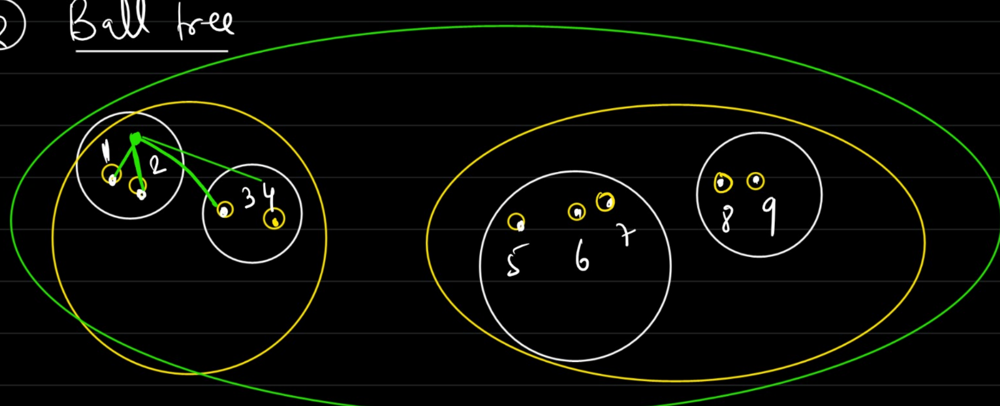
KD-tree (Binary tree)



Advantage

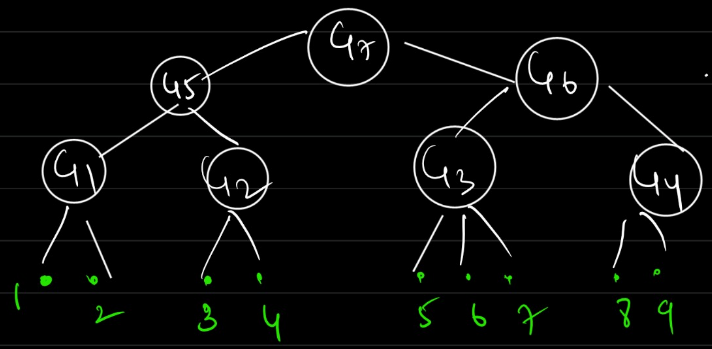
→ You don't need to calculate all the distance.

② Ball tree



Advantage

→ only need calculate distance of the nearest group elements



A ball tree is a data structure used to organize points in a multi-dimensional space. Each node in a ball tree represents a region of space (a "ball") defined by a center point and a radius. The tree is built by recursively dividing the data into smaller balls, each containing points closer to its center. Ball trees are useful for efficient nearest neighbor searches and range queries, especially in high-dimensional spaces. They are commonly used in machine learning and data science for tasks like clustering and finding similar points.

A KD tree is a way to organize points in a space with many dimensions, like coordinates in 2D or 3D. It's a type of binary tree where each node is a point. To build it, you split the points based on one dimension at a time, like dividing by x-coordinate, then y-coordinate, and so on. This makes it easier and faster to find nearby points or search within a range. KD trees are used in tasks like finding nearest neighbors or grouping similar points in fields like computer graphics and machine learning.

A ball tree is a data structure used to organize points in a multi-dimensional space. Each node in a ball tree represents a region of space (a "ball") defined by a center point and a radius. The tree is built by recursively dividing the data into smaller balls, each containing points closer to its center. Ball trees are useful for efficient nearest neighbor searches and range queries, especially in high-dimensional spaces. They are commonly used in machine learning and data science for tasks like clustering and finding similar points.

Key Concepts:
Instance-Based Learning: KNN is an instance-based or lazy learning algorithm, meaning it doesn't learn an explicit model but instead memorizes the training instances.

Similarity Measure: It relies on a distance metric (e.g., Euclidean distance) to measure the similarity between data points.

K-Value: The 'K' in KNN refers to the number of nearest neighbors to consider when making a prediction.

How it Works:
Training: KNN doesn't involve any training phase in the traditional sense. The training data is stored and used directly for making predictions.

Prediction:

Classification: For a new data point, the algorithm finds the 'K' closest points from the training set. The class of the new point is determined by the majority class among these neighbors.
Regression: The value of the new data point is calculated as the average (or sometimes weighted average) of the values of its 'K' nearest neighbors.
Steps to Implement KNN:
Choose the number of neighbors (K).
Calculate the distance between the new data point and all the points in the training set.
Select the K-nearest neighbors to the new data point.
Vote for the most common class (for classification) or average the values (for regression).
Advantages:
Simple to understand and implement.
No assumptions about the data distribution.
Effective with a small number of dimensions and training data.
Disadvantages:
Computationally intensive, especially with large datasets.
Performance can degrade with high-dimensional data (curse of dimensionality).
Sensitive to noisy data and irrelevant features.