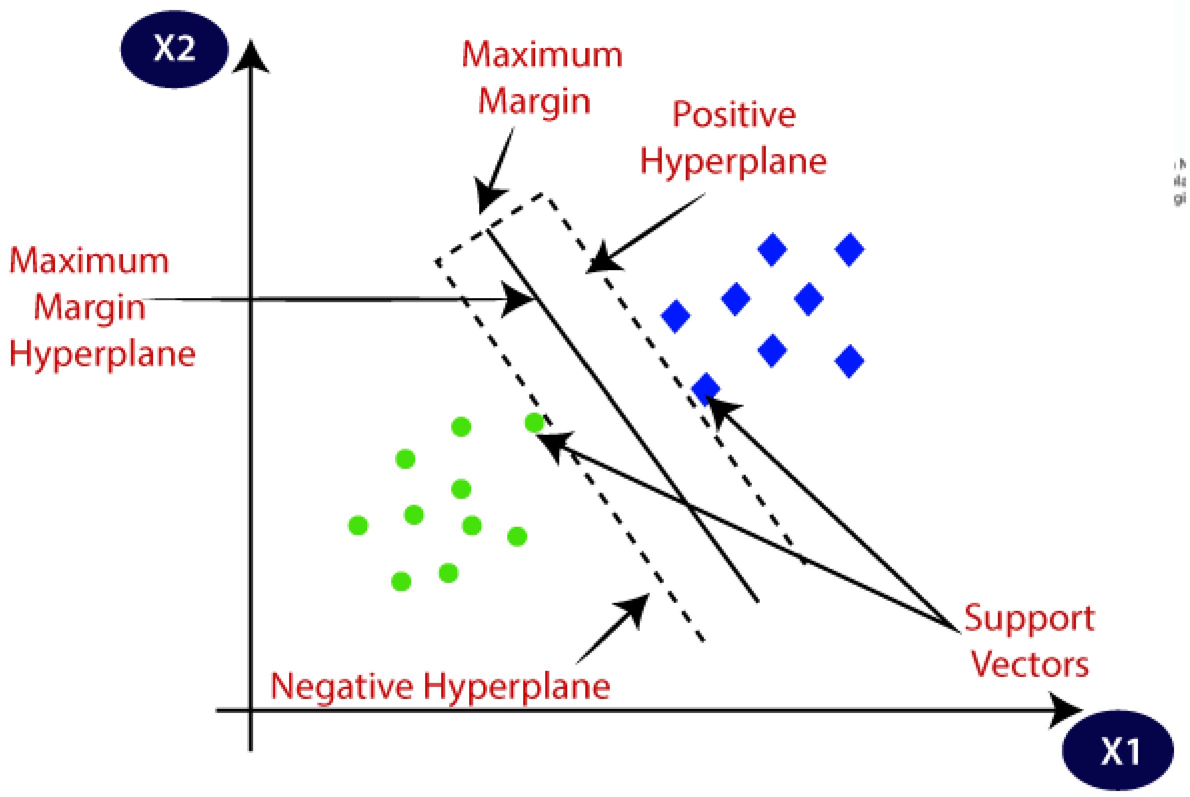
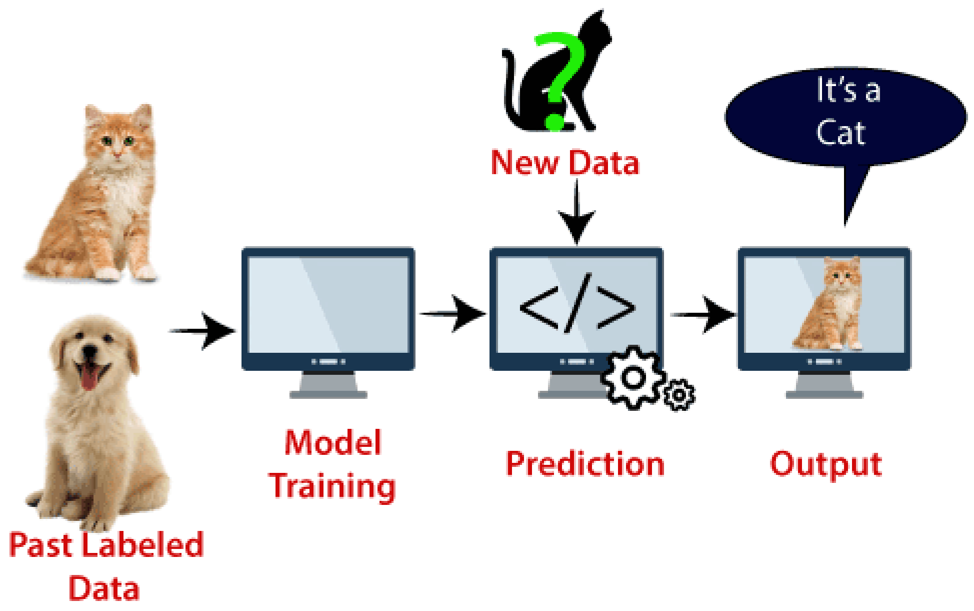

X2
Maximum Margin
Positive Hyperplane
Maximum Margin Hyperplane
Support Vectors
Negative Hyperplane
X1

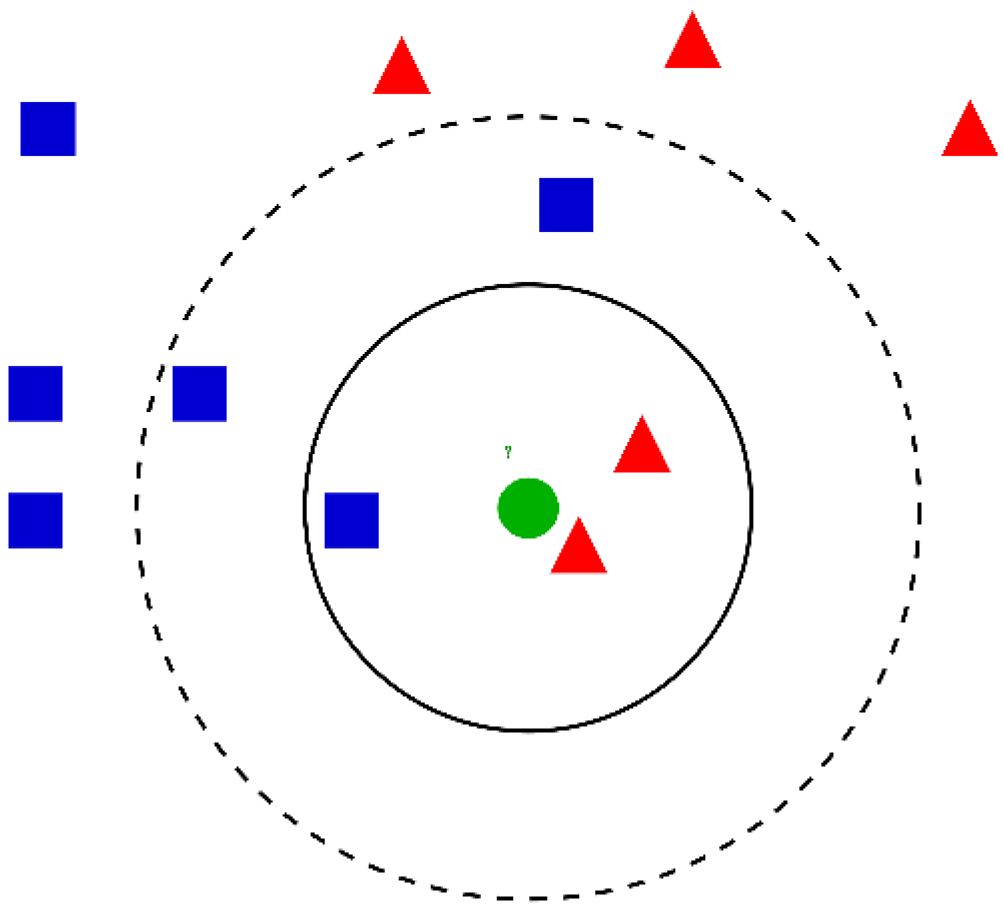

New Data
It's a Cat
Model Training
Prediction
Output
Past Labeled Data

A Complete Guide to K-Nearest Neighbors (Updated 2023)

Introduction

In the four years of my data science career, I have built more than 80% of classification models and just 15-20% of regression models. These ratios can be more or less generalized throughout the industry. The reason behind this bias towards classification models is that most analytical problems involve making decisions.

In this article, we will talk about one such widely used machine learning **classification technique** called k nearest neighbor (KNN) algorithm. Our focus will primarily be on how the algorithm works on new data and how the input parameter affects the output/prediction.

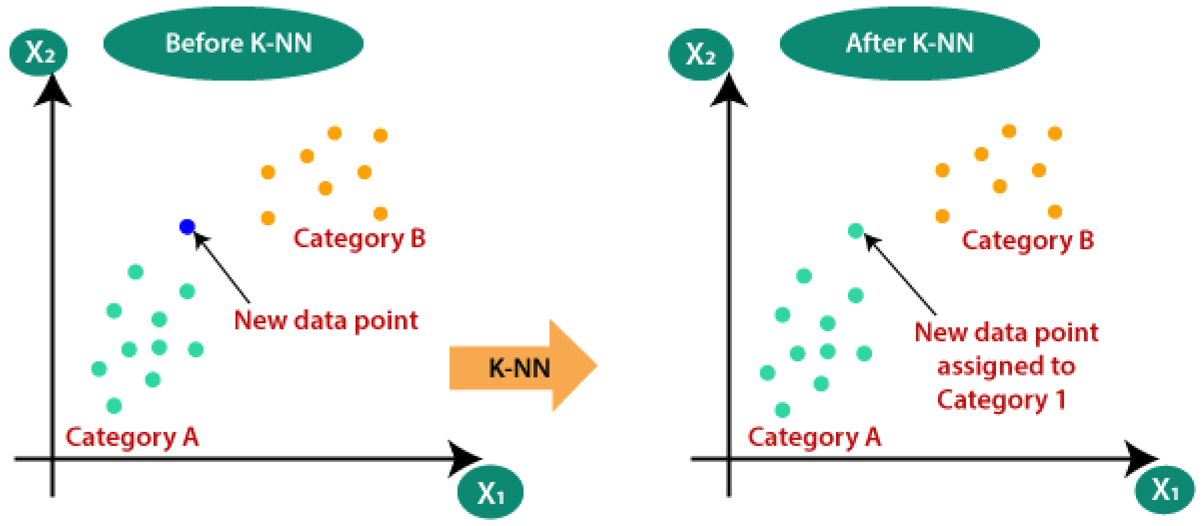


What is KNN (K-Nearest Neighbor) Algorithm?

The K-Nearest Neighbor (KNN) algorithm is a popular machine learning technique used for classification and regression tasks

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.



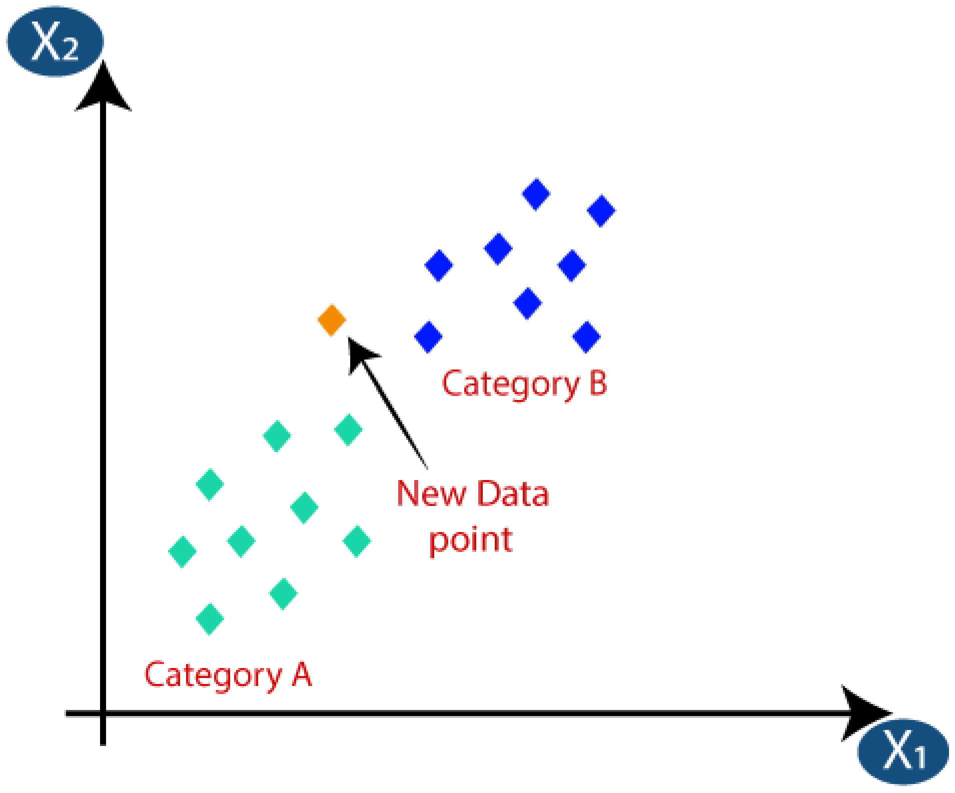
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

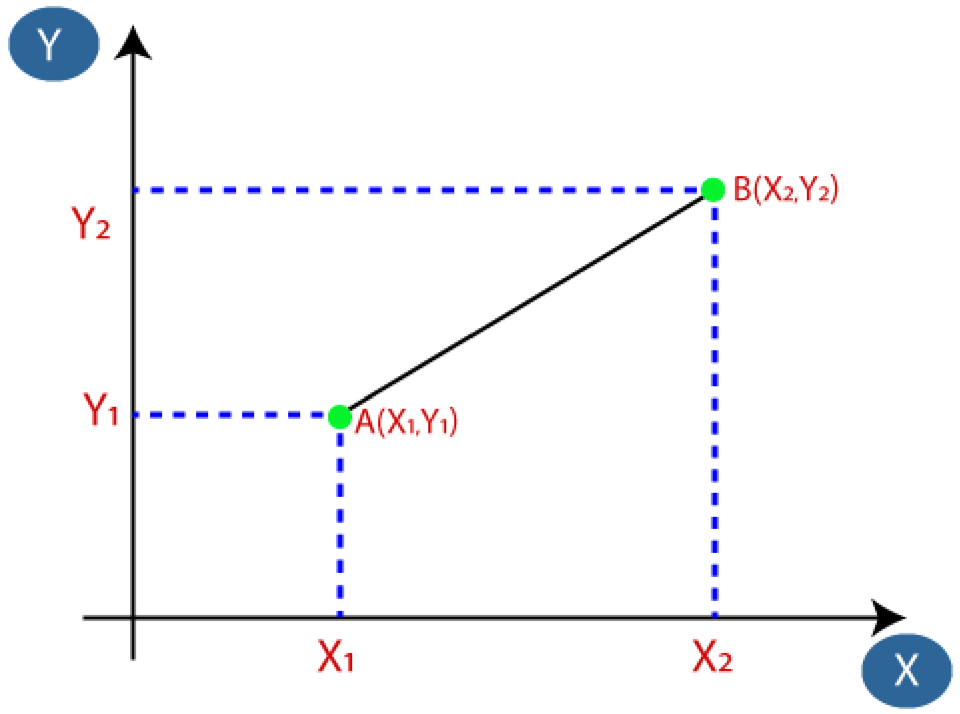
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

- **Step-6:** Our model is ready.

we have a new data point and we need to put it in the required category. Consider the below image:

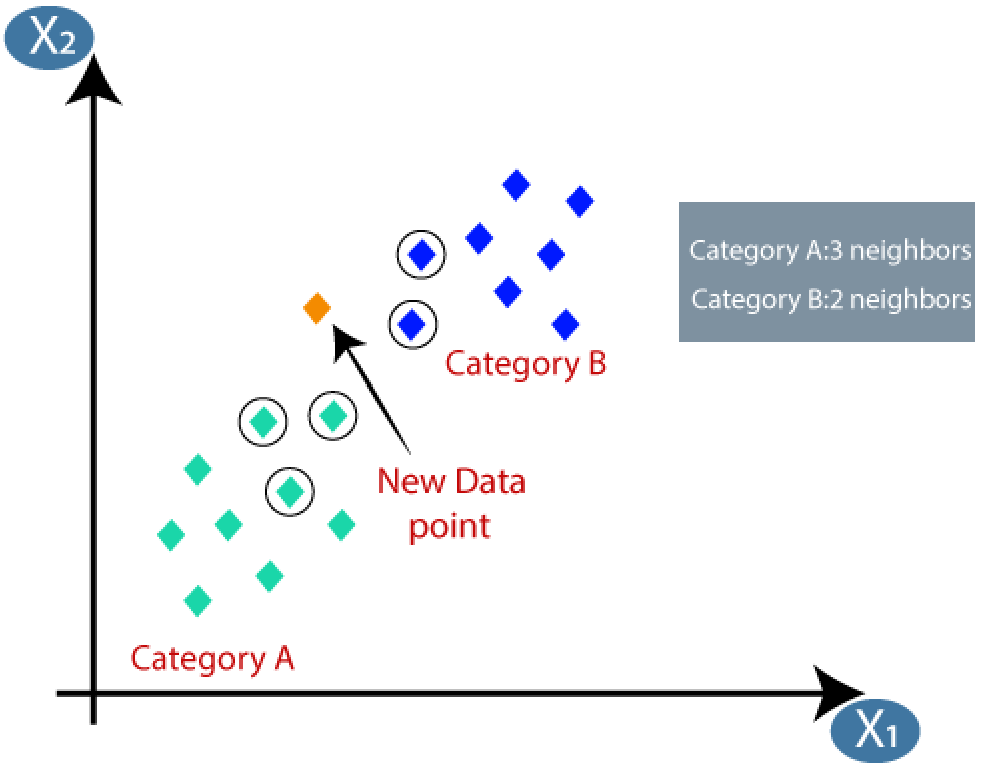


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance, we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

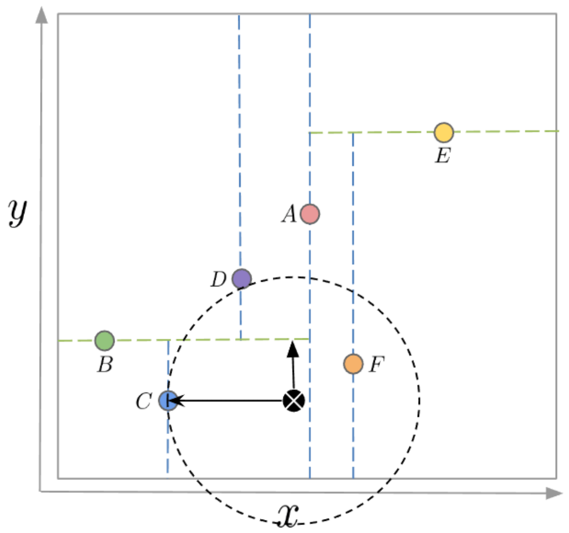
How to select the value of K in the K-NN Algorithm?

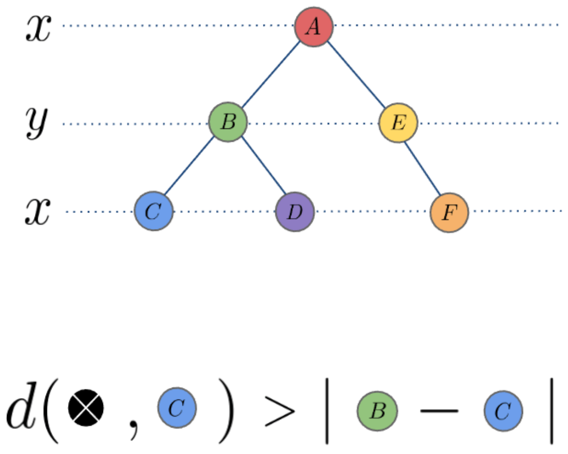
- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

KD Tree and Ball Tree – KNN Algorithm

1. **Brute Force Approach**
   This is the basic approach in which the distance of the data point is calculated with all the other data points and then the k nearest neighbours are selected. Let's assume N is the number of training data in the dataset and D is the dimension or the number of features. Then the complexity of this algorithm is $O[ND^2]$.

2. **KD Tree**
   KD Tree stands for K-Dimensional Tree. It is a tree-based data structure that helps in reducing search complexity. The complexity of this algorithm is $O[ND\log(N)]$ which is much less than Brute Force Approach.
3. **Ball Tree**
   KD Tree becomes inefficient for datasets having higher dimensions. A ball Tree is another tree-based data structure that is much faster for storing multidimensional data.





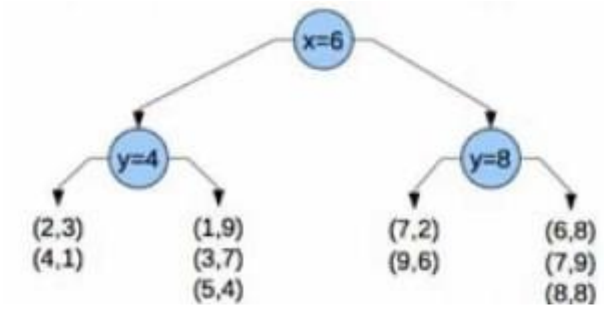
How is KD Tree Built?

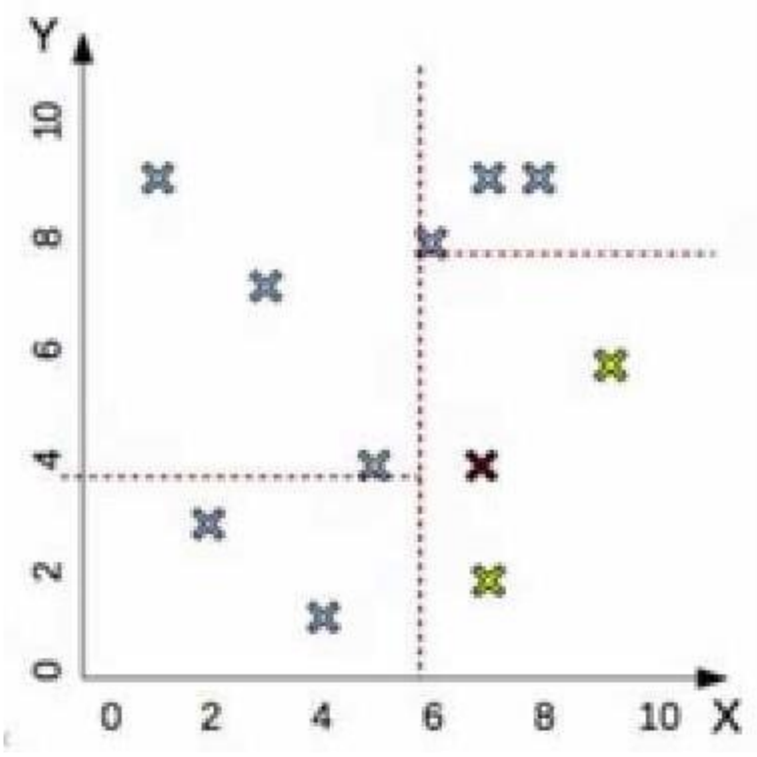
KD tree is built by splitting training data into random dimensions. Let's build a KD tree for 2 dimensional which has an x and y-axis.

Consider training data points with 2 dimensions **(x,y) – (1,9), (2,3), (4,1), (3,7), (5,4), (6,8),(7,2), (8,8), (7,9), (9,6).** Below are the steps to divide the space into multiple parts using KD Tree.

1. **Pick a random dimension.**
   Let us select x dimension, x data points are 1,2,4,3,5,6,7,8,7,9.
2. **Find the median**
   Sort the above data 1,2,3,4,5,6,7,7,8,9. Find the middle value which is 6. The median of these points is 6.
3. **Split the data into approximately equal halves.**
   Split the data on the x-axis.

4. **Repeat the above three steps**
In the next iteration, the other dimension y is selected, its median is found and data is again split. The second and third step is repeated to divide the space into multiple parts. In the below graph, you can see that the space is first divided on x=6 and then on y=4, and y=8.





Ball Tree – KNN Algorithm
Similar to KD Tree, in the Ball Tree the total space of training data is divided into multiple balls (circular blocks), and the distance of testing data is calculated only with the training points in that block instead of calculating with all the training data points.

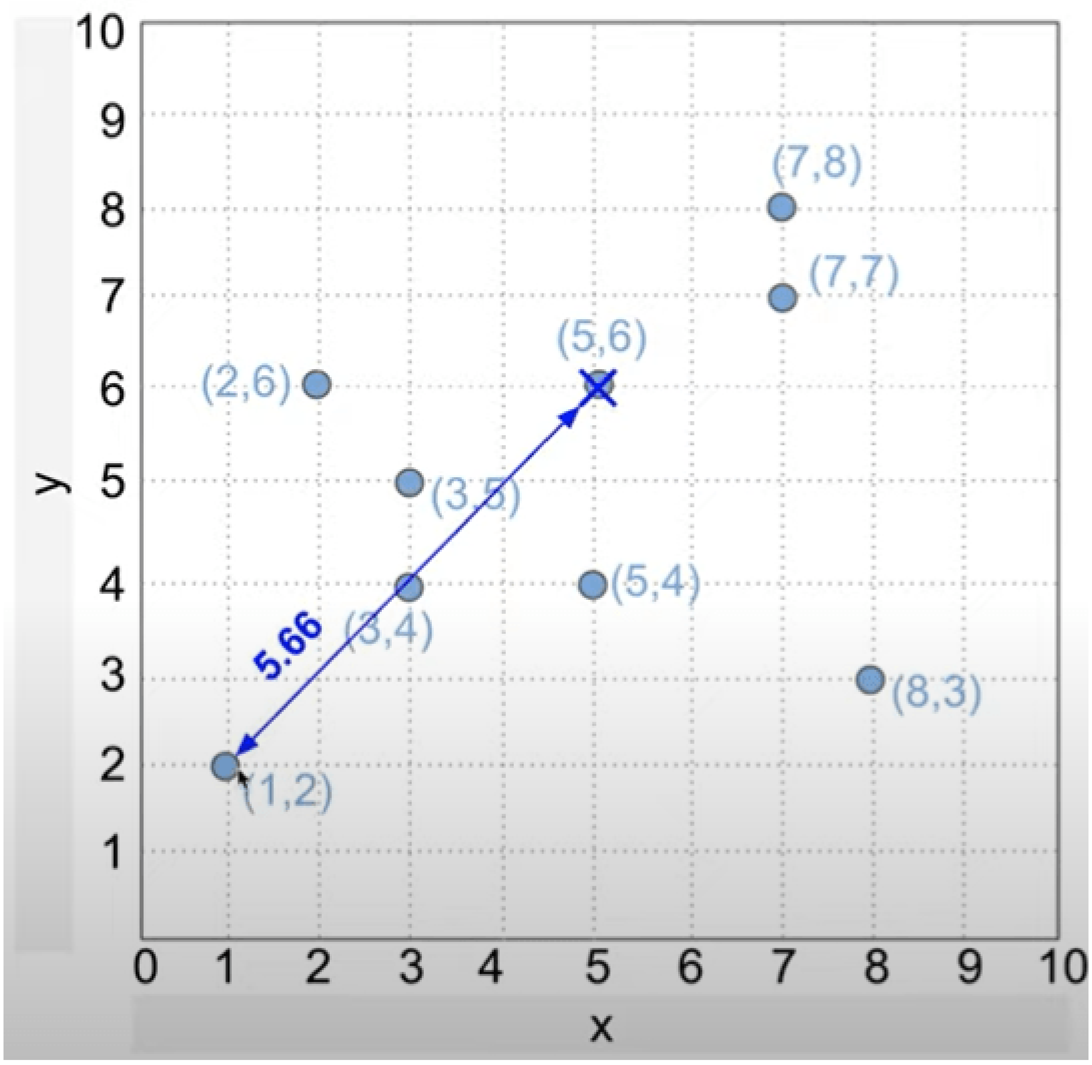
How is Ball Tree Built?

Ball Tree is built by splitting the complete space into multiple smaller circular blocks.

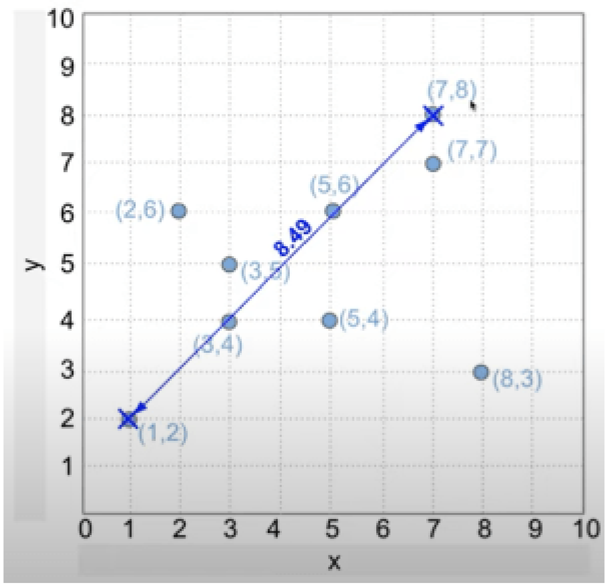
Consider training data with two-dimensional data **(x,y) – (1,2), (2,6), (3,4), (5,6), (7,8), (8,3).** Plot it on a graph. Below are the steps to divide the space into multiple parts using the Ball Tree.

1. **Select a random data point (5,6).**

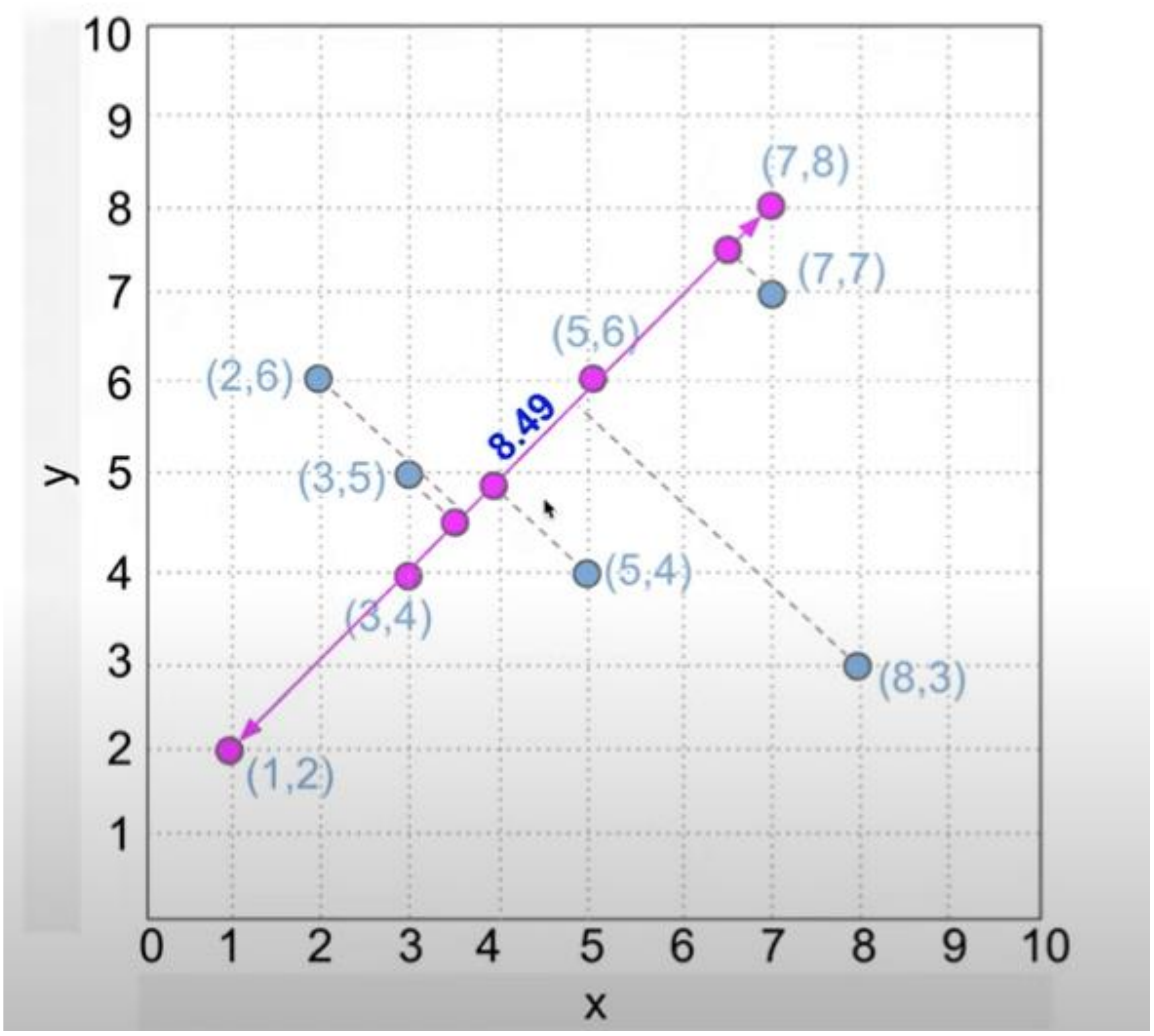
2. **Find a point farthest to that point (1,2).**



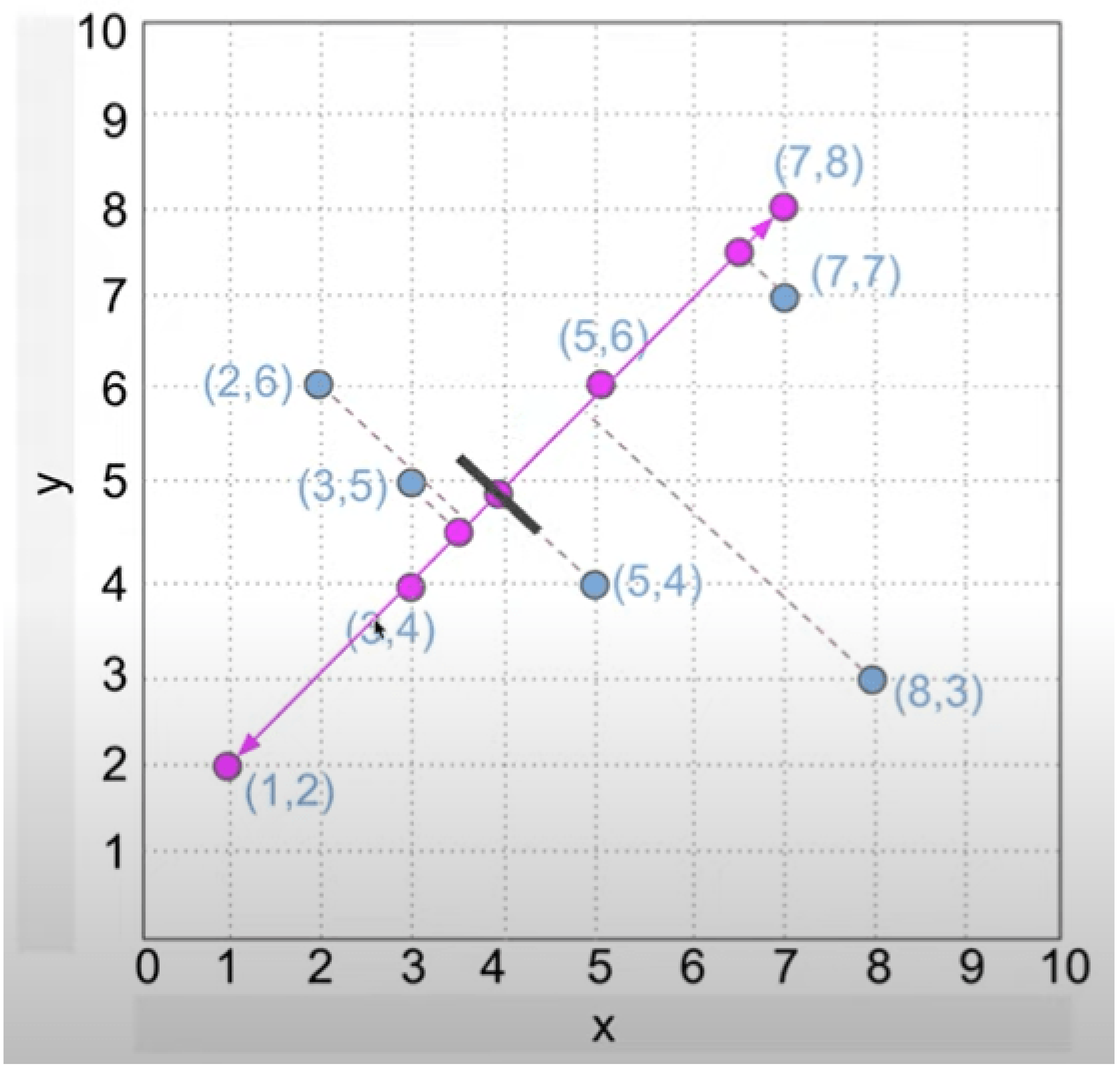
3. **Again find the farthest point to the current point (7,8).**



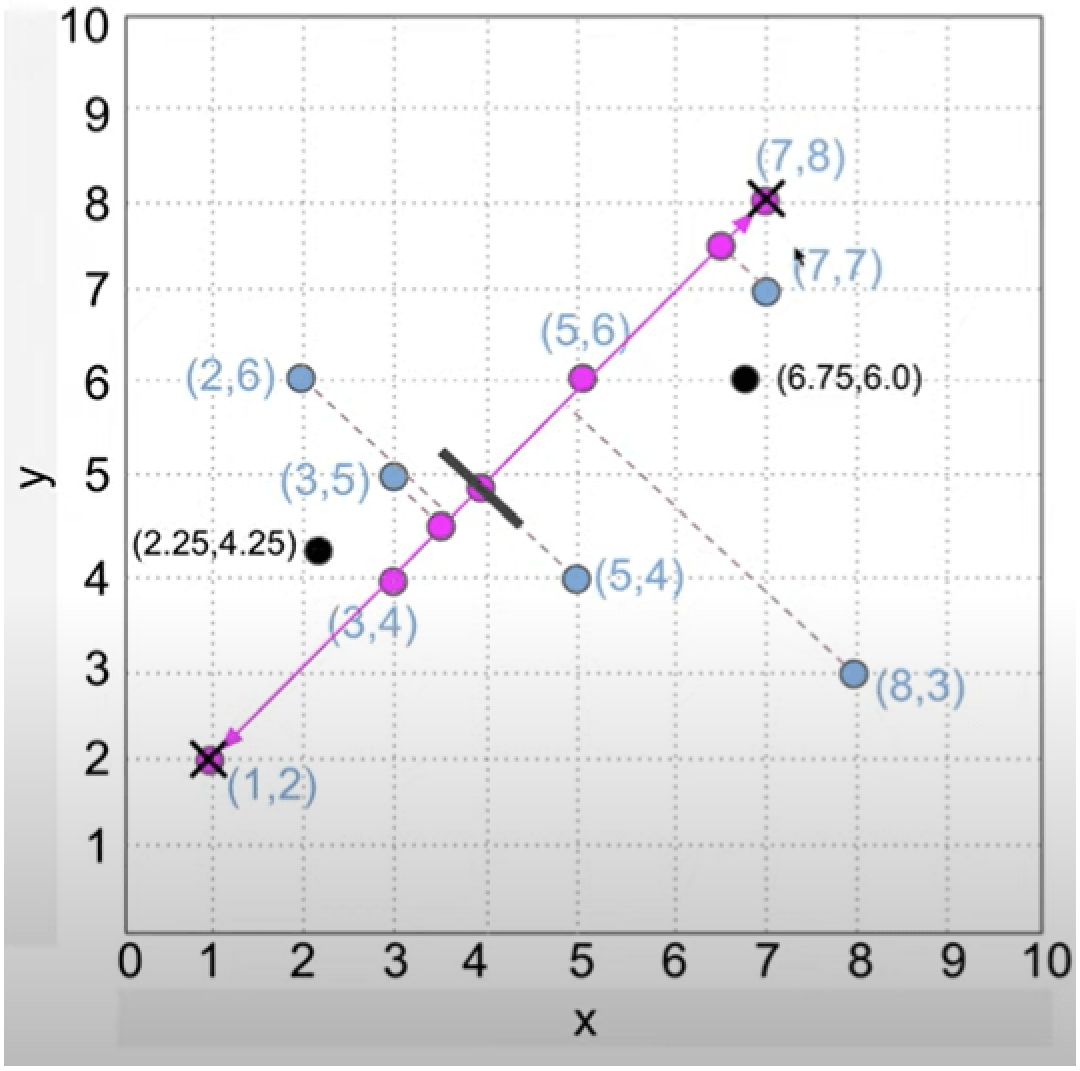
4. **Project all the points on the line joining the farthest points.**



5. **Find the median to divide the space into two halves.**

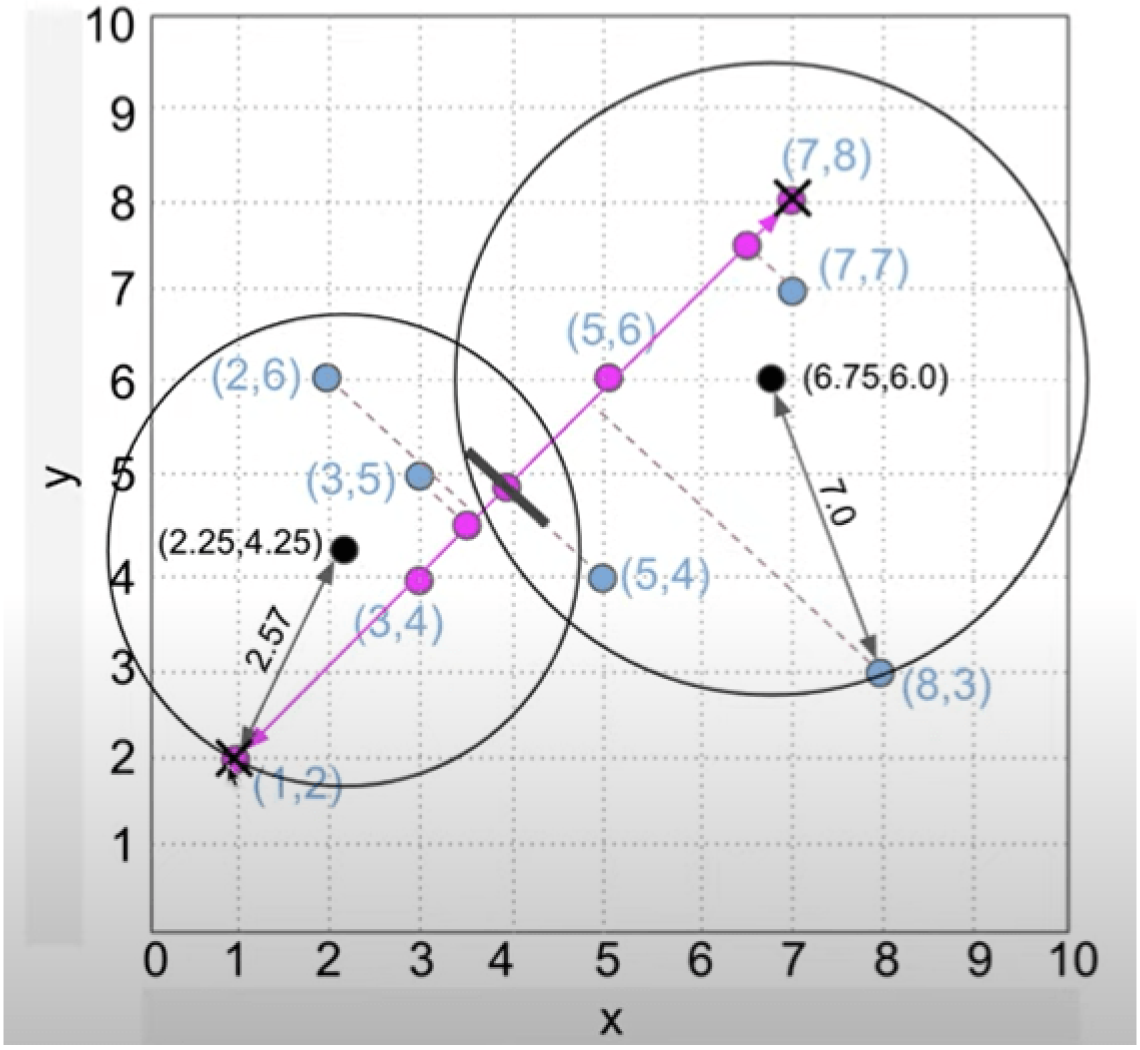


6. **Find the centroid in each halve. The centroids are denoted by black dots.**

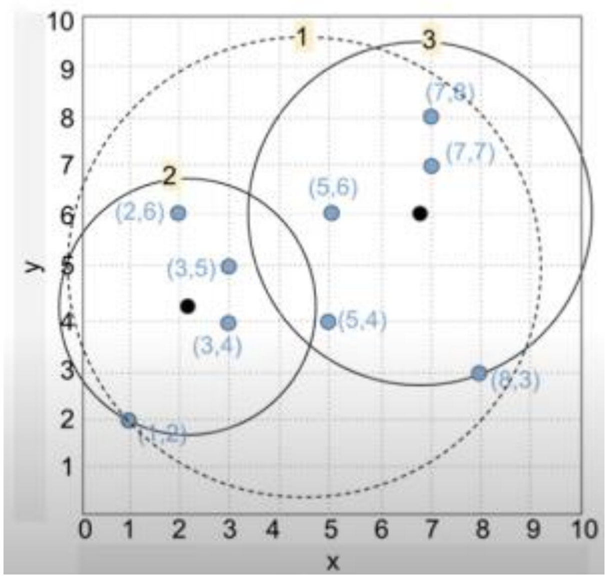


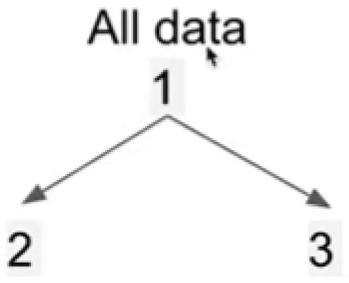
7. **In each halve, find the point farthest from the centroid and draw the circle.** (1,2) is selected as the farthest point in the first ball and (8,3) is the farthest point in another ball. **The radius of each circle is the distance from the centroid to the**

farthest point.

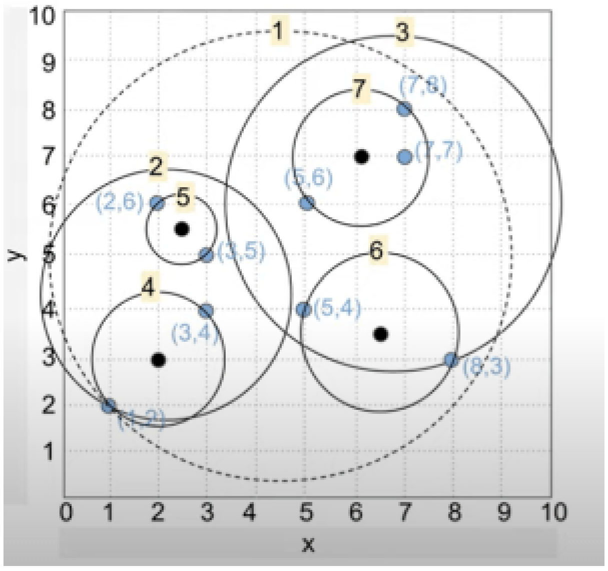


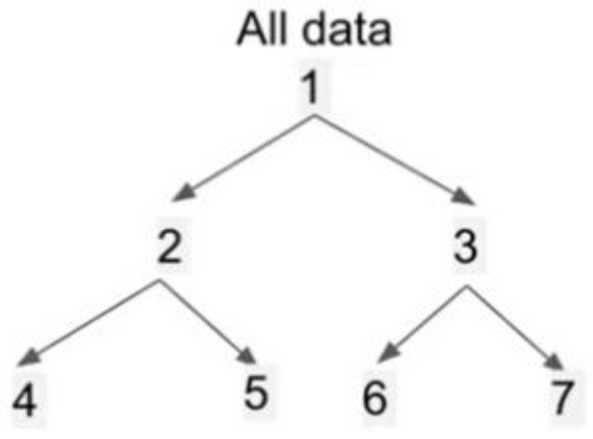
In the below image, ball 1 contains all the training data points which are split into two balls ball 2 and ball 3.





All the above 7 steps are repeated again in each individual ball and the data is further divided into smaller balls.

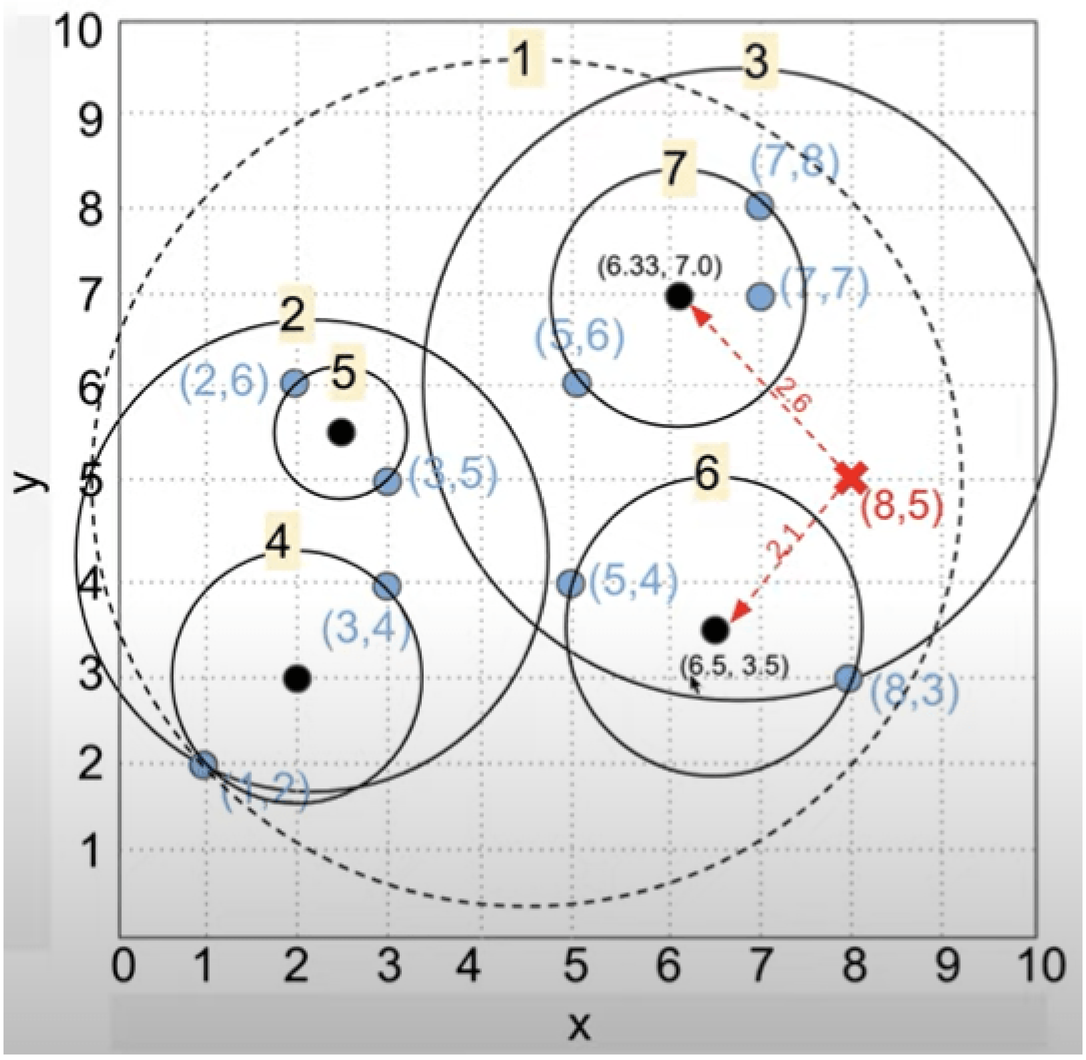




Prediction using Ball Tree

Let (8,5) be a testing data point, it belongs to ball 3 but ball 3 is further divided into ball 6 and ball 7.

Calculate the distance of (8,5) with the centroid of ball 6 and ball 7. The centroid of ball 6 is found to be nearest to the testing data point (8,5). Hence the training data points in ball 6 are used for predicting the output of (8,5).



Principal Component Analysis

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**.

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Basic Terminologies of PCA

Before getting into PCA, we need to understand some basic terminologies,

- **Variance** – for calculating the variation of data distributed across dimensionality of graph
- **Covariance** – calculating dependencies and relationship between features
- **Standardizing data** – Scaling our dataset within a specific range for unbiased output

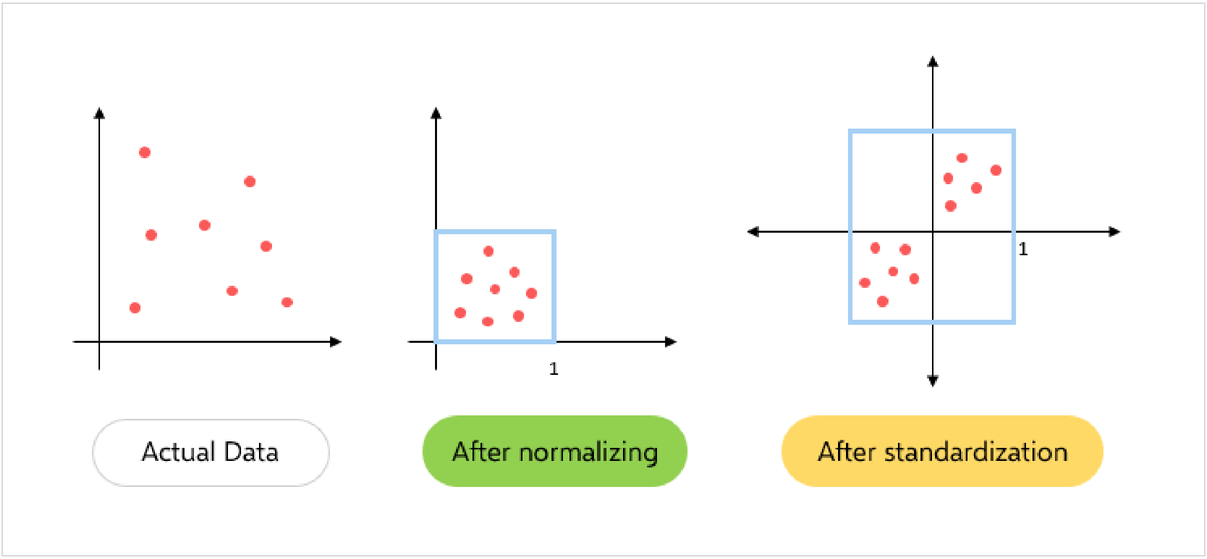
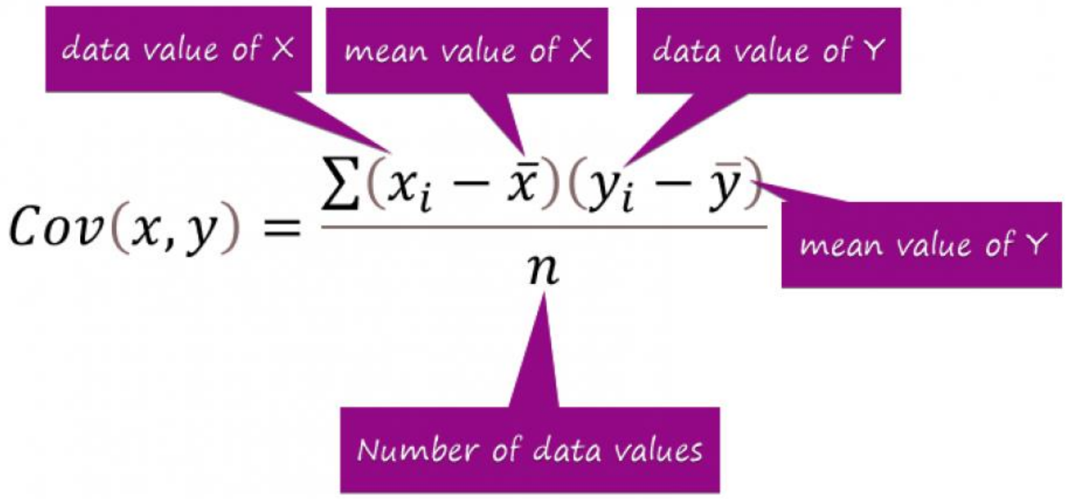


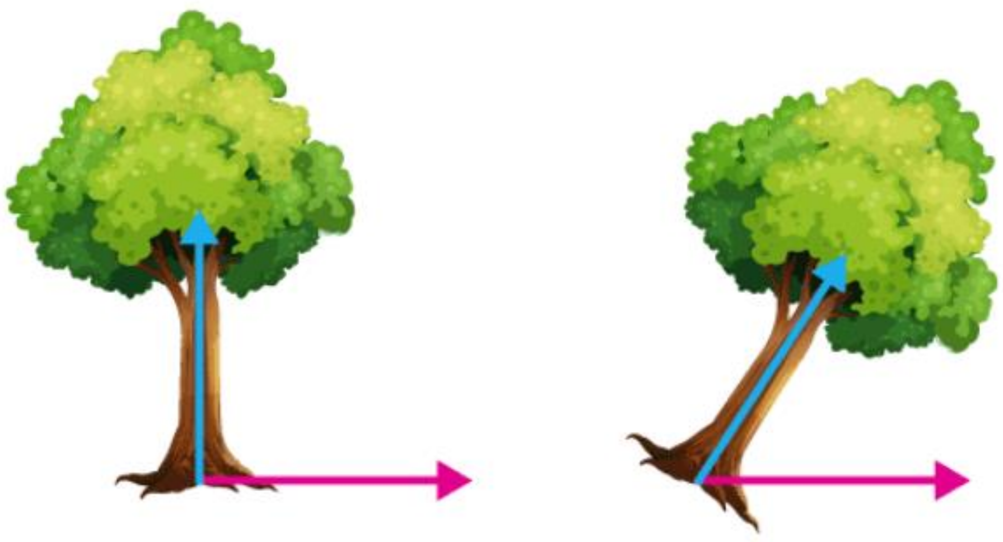
Image Source: PCA Terminologies

- **Covariance matrix** – Used for calculating interdependencies between the features or variables and also helps in reduce it to improve the performance



Source: https://www.exceldemy.com/calculate-covariance-matrix-in-excel/

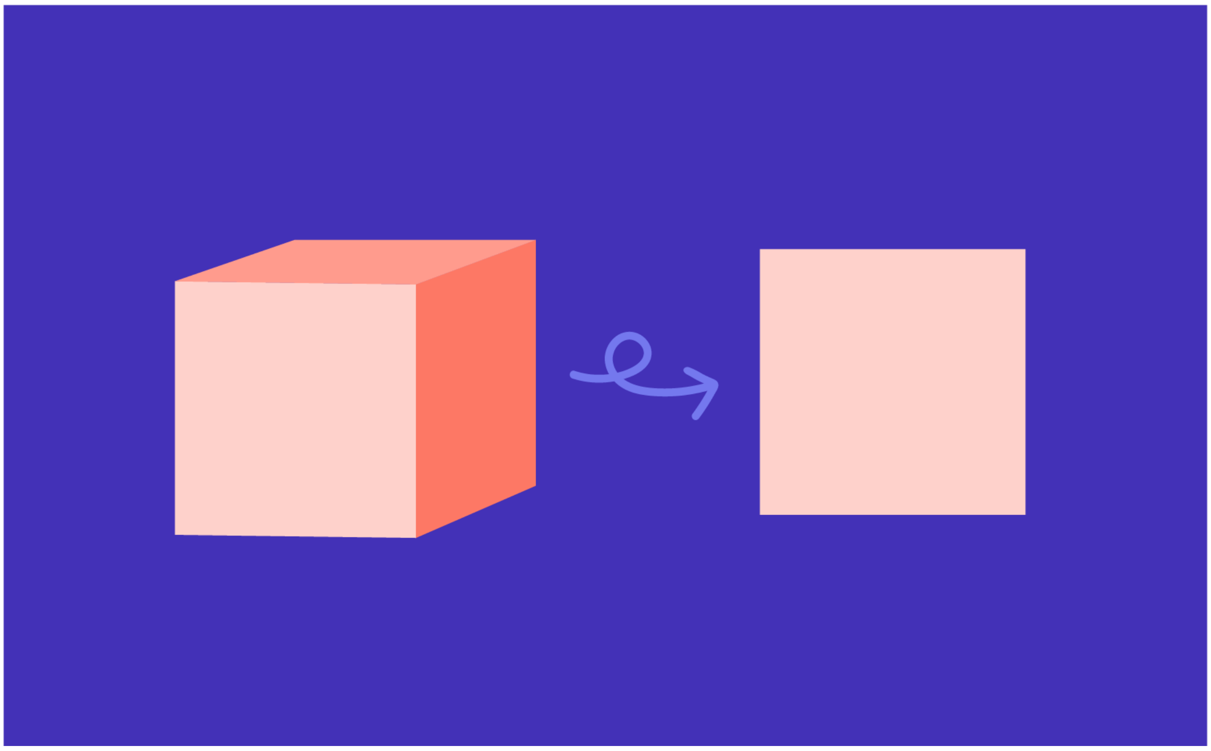
- **EigenValues and EigenVectors** – Eigenvectors' purpose is to find out the largest variance that exists in the dataset to calculate Principal Component. Eigenvalue means the magnitude of the Eigenvector. Eigenvalue indicates variance in a particular direction and whereas eigenvector is expanding or contracting X-Y (2D) graph without altering the direction.



Source: https://byjus.com/maths/eigen-values/

In this shear mapping, the blue arrow changes direction whereas the pink arrow does not. The pink arrow in this instance is an eigenvector because of its constant orientation. The length of this arrow is also unaltered, and its eigenvalue is 1. Technically, PC is a straight line that captures the maximum variance (information) of the data. PC shows direction and magnitude. PC are perpendicular to each other.

- **Dimensionality Reduction –** Transpose of original data and multiply it by transposing of the derived feature vector. Reducing the features without losing information.



Source: https://www.displayr.com/category/data-science/dimension-reduction/

How does PCA work?

The steps involved for PCA are as follows-

1. Original Data
2. Normalize the original data (mean =0, variance =1)
3. Calculating covariance matrix
4. Calculating Eigen values, Eigen vectors, and normalized Eigenvectors
5. Calculating Principal Component (PC)
6. Plot the graph for orthogonality between PCs

Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.