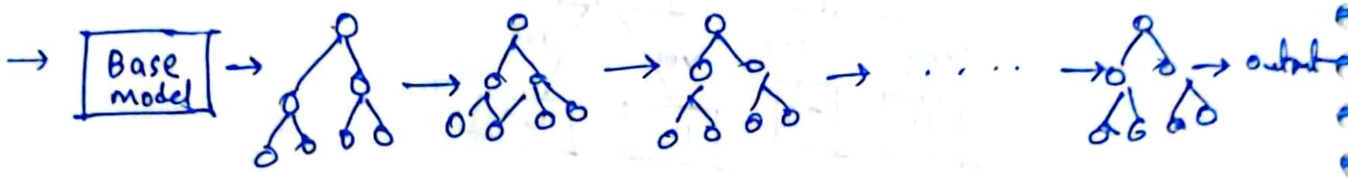
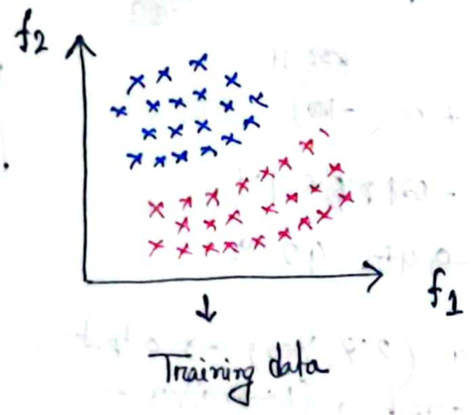
for, $\{x_1, x_2, R_1\}$ we will get another prediction $\hat{y}$ using which we will make R_3, then we will make ~~DT~~ DT split based on ~~R3~~ $\{x_1, x_2, R_3\}$
It will continue till the n ~~nu~~ number of decision tree we have chosen



K Nearest Neighbour (KNN):

Can solve both - classification
- Regression

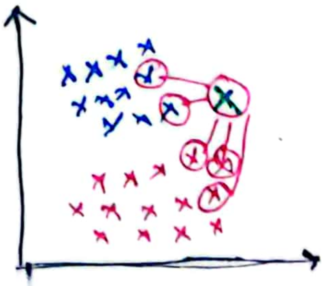
f_1	f_2	y
–	–	0
–	–	1
–	–	1
–	–	0



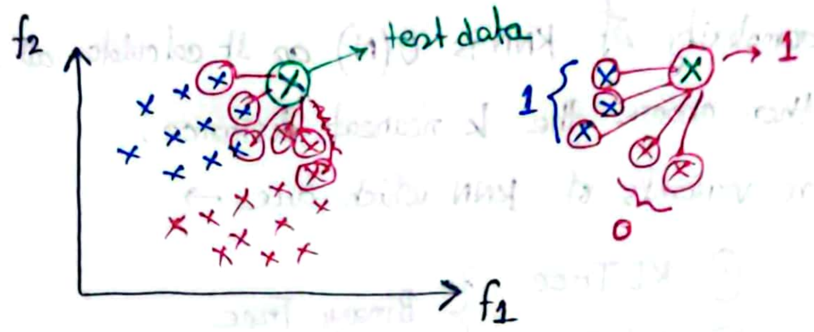
Steps:

① We have to initialize the K value (K: the number of nearest neighbours)
$K > 0$, $K = 1, 2, 3, 4, 5, \ldots\ldots$
K is a hyperparameter.

② Find the K nearest neighbour from test data
Suppose $K = 5$ is selected



③ From the K=5 how many neighbours belongs to 0 category or 1 category. In our examples 2 neighbours belong to 0 and 3 neighbours belong to 1. Maximum number of neighbours are from 1 category. So, the new test data will ~~[illegible]~~ provide prediction 1.



Distance Matrics: (How distance are calculated?

① Eucledian Distance

② Manhattan Distance

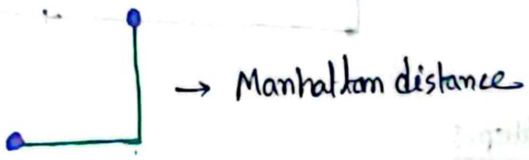
→ Eucledian Distance

Eucledian distance Formulas →

For 2D, $\sqrt{(x_1-x_2)^2+(y_2-y_1)^2}$

For, 3D, $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$

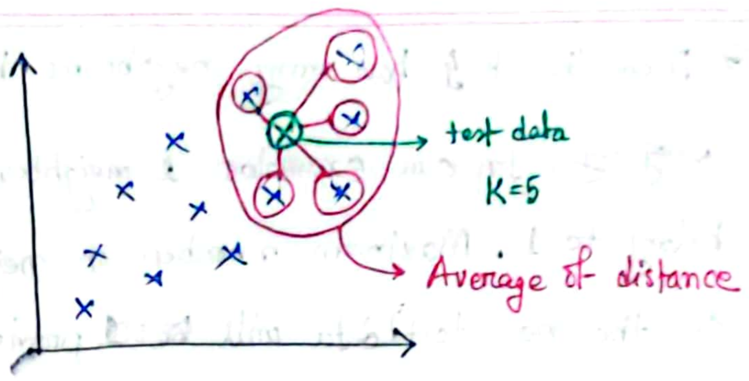
Manhattan distance Formula →



For 2D, $|x_1-x_2|+|y_1-y_2|$

For 3D, $|x_1-x_2|+|y_1-y_2|+|z_1-z_2|$

For Regression:



But time complexity of KNN is O(N) as it calculates all the neighbour distance then choose the k nearest distance.

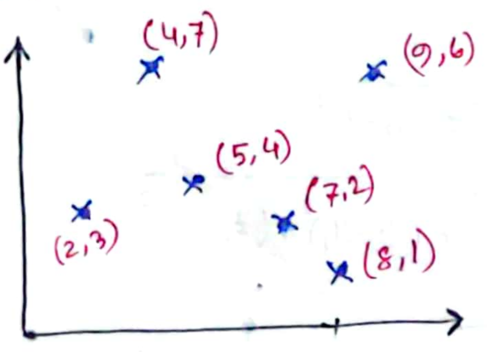
So, we have variants of KNN which are →

1. KD Tree
2. Ball Tree

} Binary Tree

To reduce the time complexity.

KD Tree: (K Dimensions Tree)



Steps:

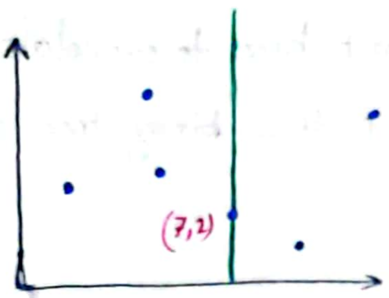
1. Find the median of the two coordinates.

$2, 4, \boxed{5, 7}, 8, 9$

$\downarrow$

$\frac{5+7}{2} = 6$

6 coordinate is not there, so we will take either 5, or 7.



We projected a line in (7,2) in the x axis. That divided the graph into two parts.

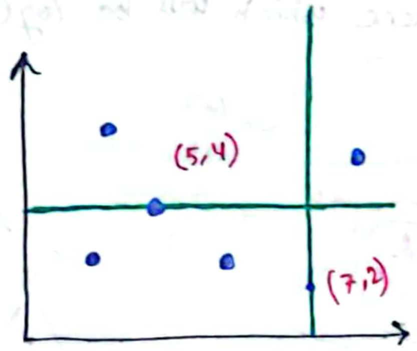
② Find the median of y axis:

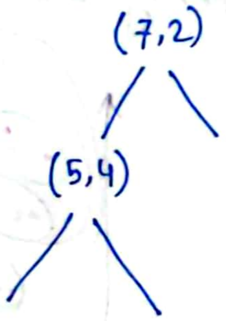
1, 2, $\boxed{3, 4}$, 6, 7

↓

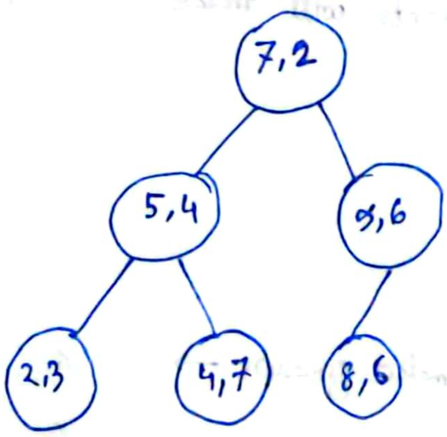
$\frac{3+4}{2} = 3.5$

3.5 is not there. So either we will take 3 or 4. and project a line in y axis.

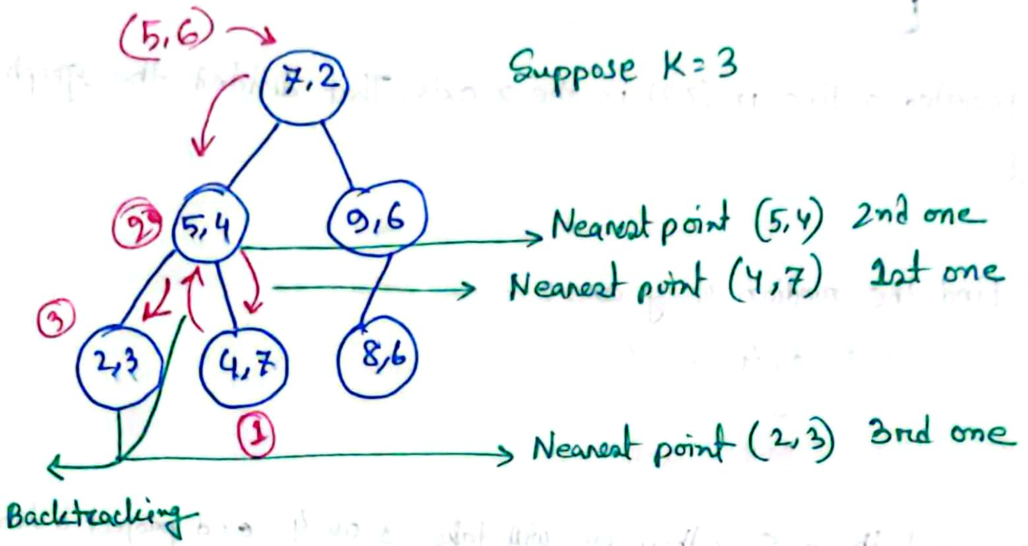




Then dividing like this other branches will form.

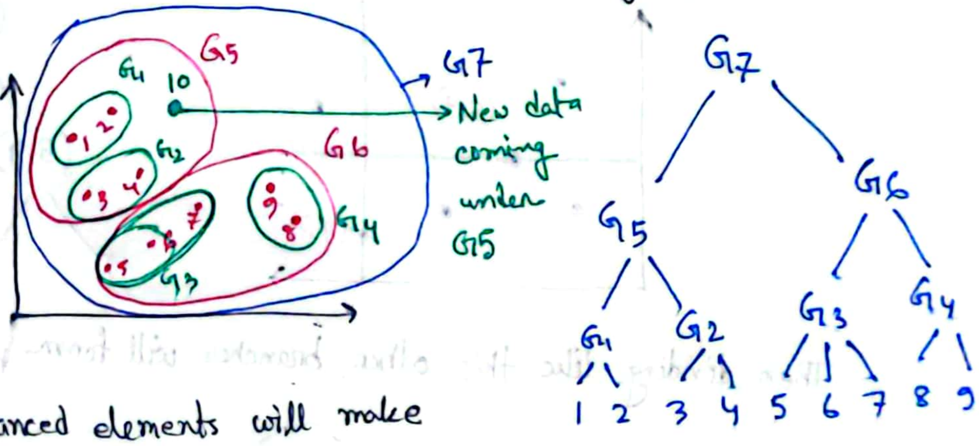


For a new data (5,6) coordinate it doesn't have to calculate all the distance now, rather it can traverse the binary tree and find the shortest distance


(5,6)
7,2
5,4
9,6
2,3
4,7
8,6
Suppose K=3
Nearest point (5,4) 2nd one
Nearest point (4,7) 1st one
Nearest point (2,3) 3rd one
1
2
3
Backtracking

So, time complexity will reduce here which will be $\log(n)$.

Ball tree:



Here, nearest distanced elements will make groups,

$1,2 \rightarrow G_1$

$3,4 \rightarrow G_2$

$5,6,7 \rightarrow G_3$

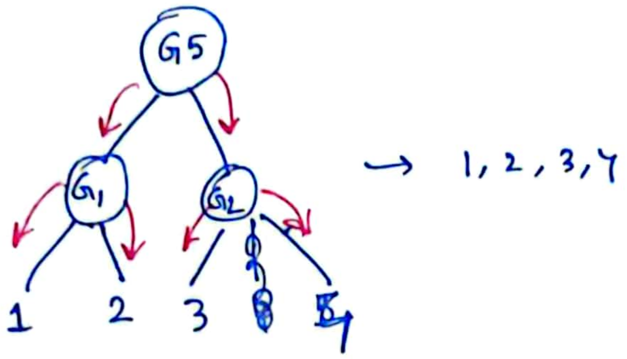
$8,9 \rightarrow G_4$

Then nearest Groups will make groups $\rightarrow$ $G_1, G_2 \rightarrow G_5$

$G_3, G_4 \rightarrow G_6$

$G_5, G_6 \rightarrow G_7$

Suppose for a new data point 10, we can see that it comes under G5. So, it's nearest neighbours will be



Which will reduce the time complexity to find K nearest neighbours.

In sklearn we have "auto" parameter to check KD Tree or Ball Tree which gives better results for our dataset.