⇒ K Nearest Neighbour (KNN)

Used for both

1. Classification
2. Regression

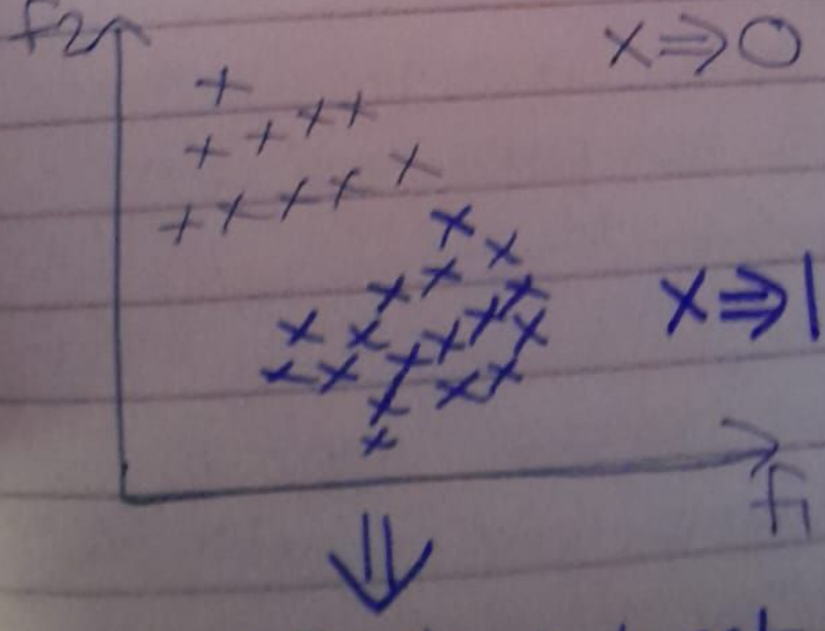
① Classification:

Let we have dataset

Independent → f_1 f_2 ; Dependent → y (Binary Category)

f_1	f_2	y
—	—	1
—	—	0
—	—	0
—	—	1

Let's plot the points of 0 and 1



⇓

Training Dataset

Step 01:

We have to specify K-value

$K > 0 \ldots \infty$

i.e $K = 1, 2, 3, 4, 5$ or any

⇓ **Hyperparameter**

(we will try different values of K in cross-validation and pick one which give highest accuracy)

Step 02:

Find the K-nearest neighbours of new test data i.e we have a new point in the graph drawn on previous page and $K = 5$, we will find 5 nearest neighbours from that new point according to distance.

Step 03:

From nearest neighbours. we found the count that how many of them belongs to 0 class and how many belong to 1 class and we will

- Distance formulas for finding K-nearest neighbours

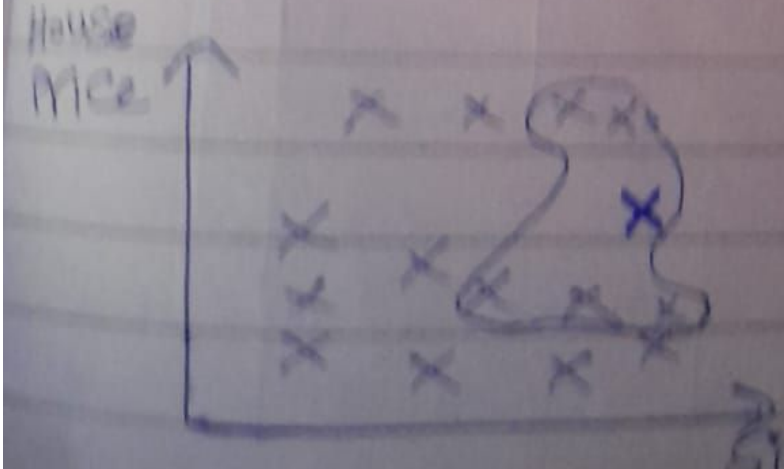
① Euclidean Distance:

- Most of time euclidean distance is used.

② Manhattan Distance:

In it we will not use direct hypot like in euclidean

③ Regression

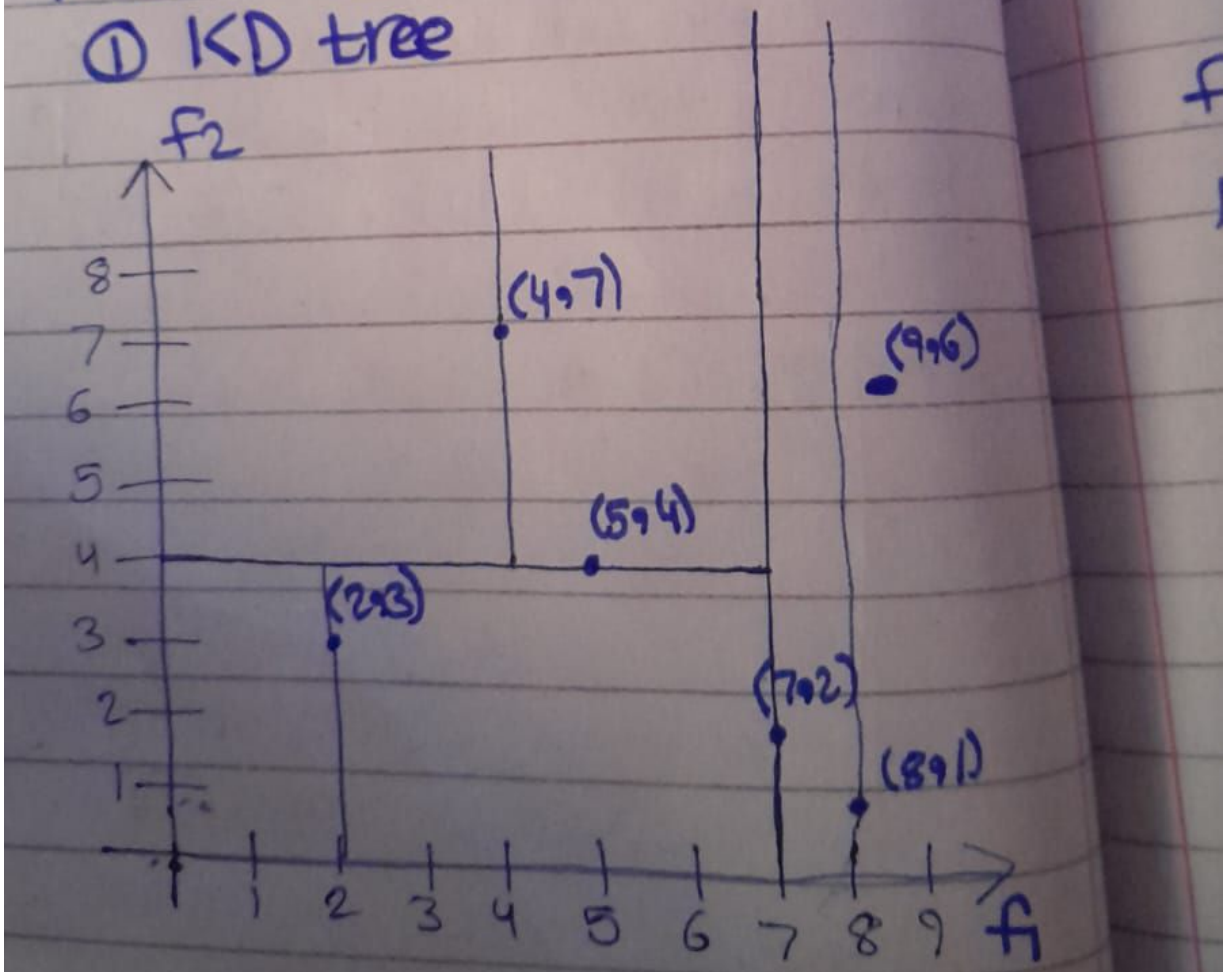


It is similar we will find k-nearest neighbours and we will find average (mean/median) of them.

• **Issue:**
In K-th Nearest Neighbour for prediction we calculate the distance of every point from new point which takes time complexity O(n).
To optimize KNN we use:
① KD tree
② Ball tree

⇒ **Variants of KNN:**

① KD tree



We have dataset

f_1	f_2
7	2
5	4
9	6
2	3
4	7
8	1

Now we will construct KD tree for decreasing time complexity of find K nearest neieigbours

- We will take median of f1 and f2

$f_1 =$ 2, 4, 5, 7, 8, 9 (sorted form)

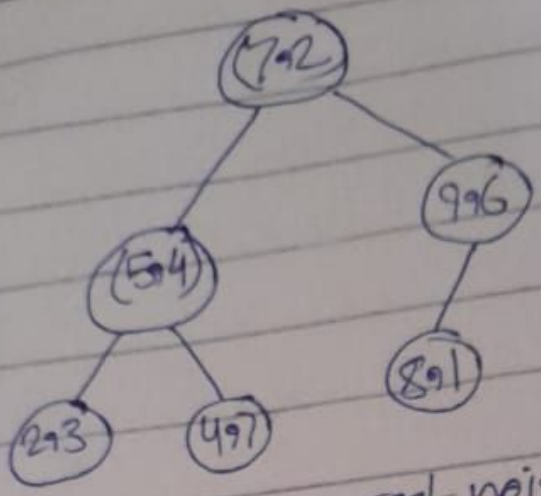
Median $= \frac{5+7}{2} = 6.5$ we will pick 7 from x-axis and draw line as there is no 6.5

$f_2 =$ 1, 2, 3, 4, 6, 7 (sorted form)

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

We will pick 4 from y-axis and draw line as there is no 3.5

So we will keep making groups by taking median of remaining and construct tree)



We find nearest neighbours using this tree. Backtracking is also involved in it

② Ball tree:

It is better than KD tree and we don't need any backtracking

- In it we try to combine adjacent elements