Complete KNN algorithm

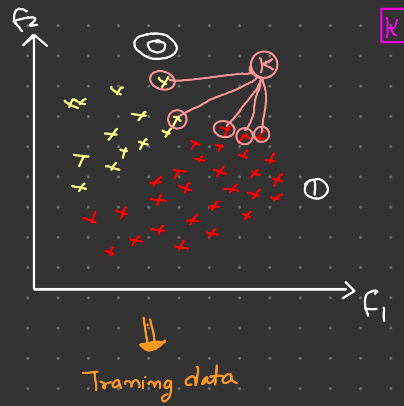
K Nearest Nighbour Algorithm (KNN)

K Nearest Neighbour (KNN) :-

helps in

1. Classification Problems
2. Regression Problems

For the Classification



Here :-

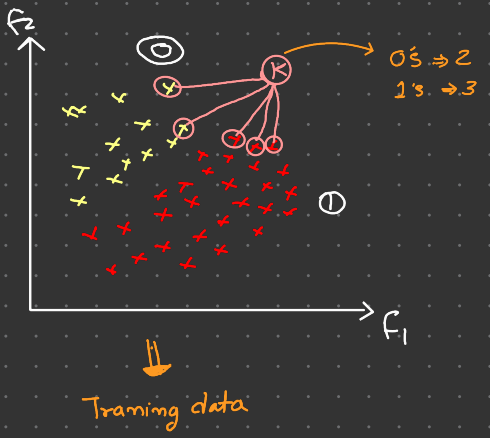
f_1	f_2	y (O/P)
-	-	0
-	-	1
-	-	1
-	-	0
-	-	1

Can be → [Binary categorical, multiclass]

Step that follow in KNN :-

- We have to initialize the K Value :
  (How many K nearest points do we need to see)
  Always → $K > 0$
  $K = 1, 2, 3, 4, 5, \ldots \Rightarrow$ Hyperparameter
- We find the KNN from the test Data :-
  (Here we are finding the distance of K to the nearest points)
  $K = 5$

➤ From those K=5 How many neighbours Belongs to 0's category and 1's category



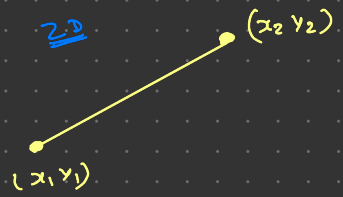
Here maximum no. of neighbours belongs to 1's category.

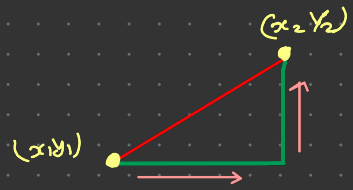
So, o/p ⟶ 1

New Predicted Test Data

* Distance Metrics :-

- Eucledian Distance
- Manhattan Distance





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

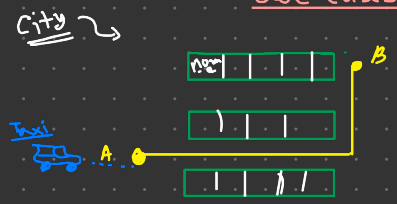
3D ⇒ Distance = $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$

Use Cases :-

* Air Traffic Control

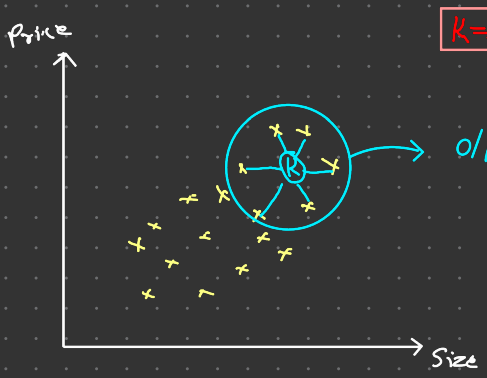
Ex India to Us flight Travel Distance

Use cases :-



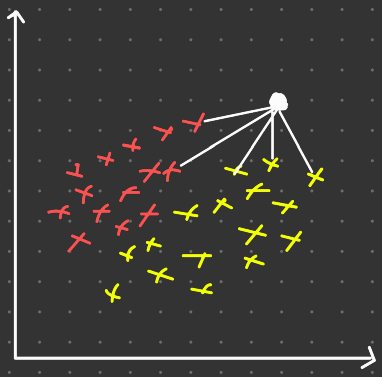
If a person want to go A → B then he book Uber and Uber Taxi choose shortest way to A → B

For Regression



In Regression we find the nearest points and than the o/p is Average of all the nearest points

Variant of KNN



When we have (data points) → Millions

↓

Create Time complexbility ⇒ ↑↑↑

↓

To decrease Time complexbility we use

↓

Variant of KNN

↓

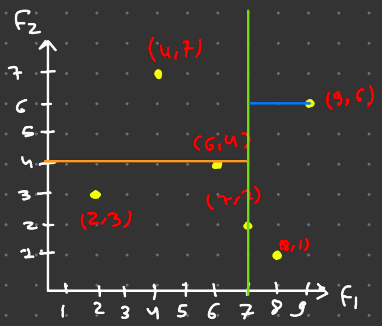
Time complexbility ⇒ ↓↓↓

1. KD-Tree
2. Ball tree

→ Binary tree

Variant of KNN

1. KD-Tree (K-Dimention tree) :-



KD-Tree follow these Steps

- Median of the x-Cordinate :-

2, 4, 6, 7, 8, 9 → (Shorted)

$\frac{6+7}{2} = 6.5$

[But 6.5 is not present the graph so we choose x=7 for the Split]

- Median of the y-cordinate :-

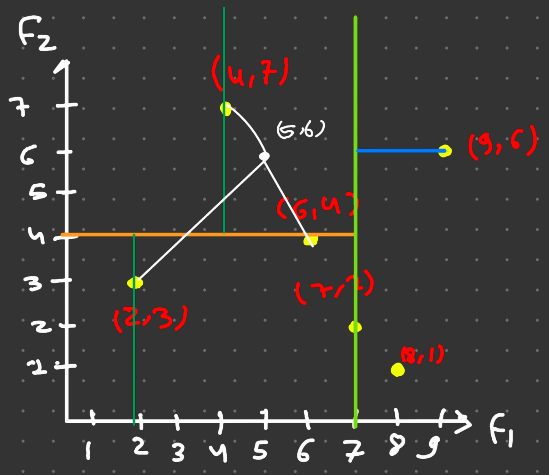
1, 2, 3, 4, 6, 7 → [Shorted]

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

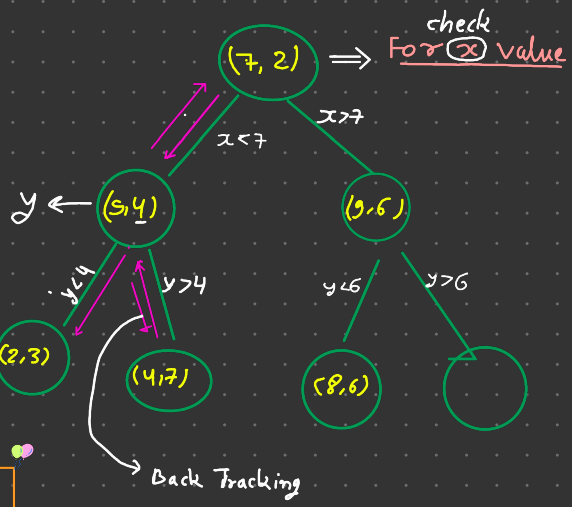
[But 3.5 is not in graph So we choose y=4 foo the Splitting]

What is Back Tracking

Back tracking involes making iterative adjustment and revisiting Decision to optimize the model performance. By this we find our nearest neighbour Points



In 2D.T.



↳ Root node ⇒ check for x cordinate

↳ Child node ⇒ check for y cordinate

Im Short :-

K.D. Tree

↓

Back Tracking

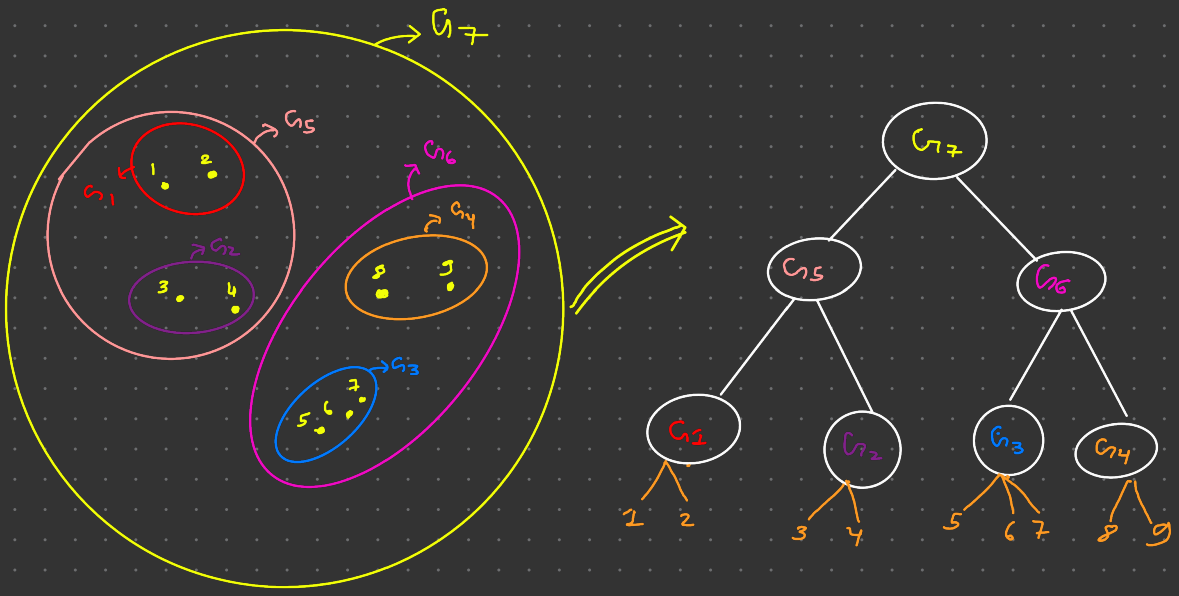
↓

No. of comparision ↓↓ → Time complexbility ↓↓

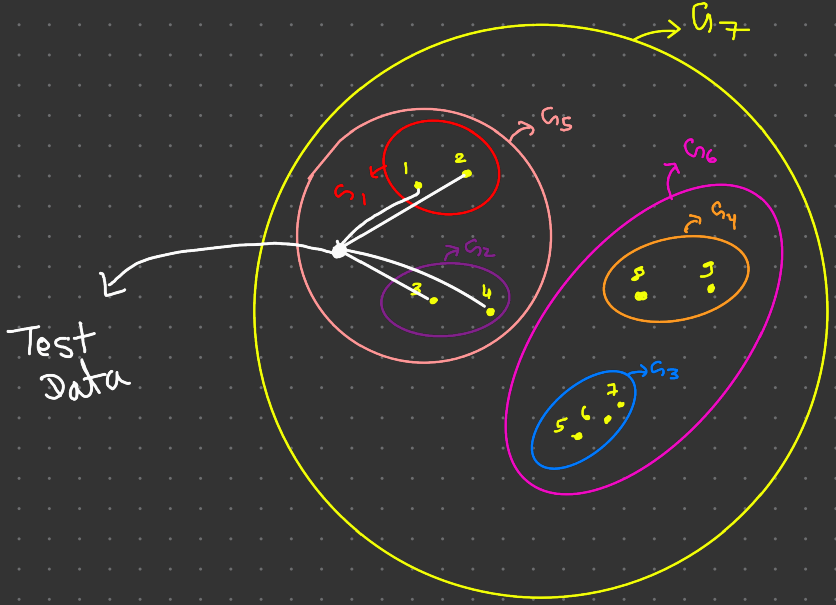
2. Ball Tree :-

Ball tree partications data points into balls, hierarchically, forming a Tree. This structure speed up Finding nearest neighbours by eliminating Distance regions. Here we <u>don't use Back tracking</u>.

ex :-



Deals with new Test Data :-



Here my New data Test Data Present in G_5 Group So I just Find my K nearest points and for this I don't need to go through other Data points I just go through G_5 Data points And I got my K nearest data points easily