Classification And Regression

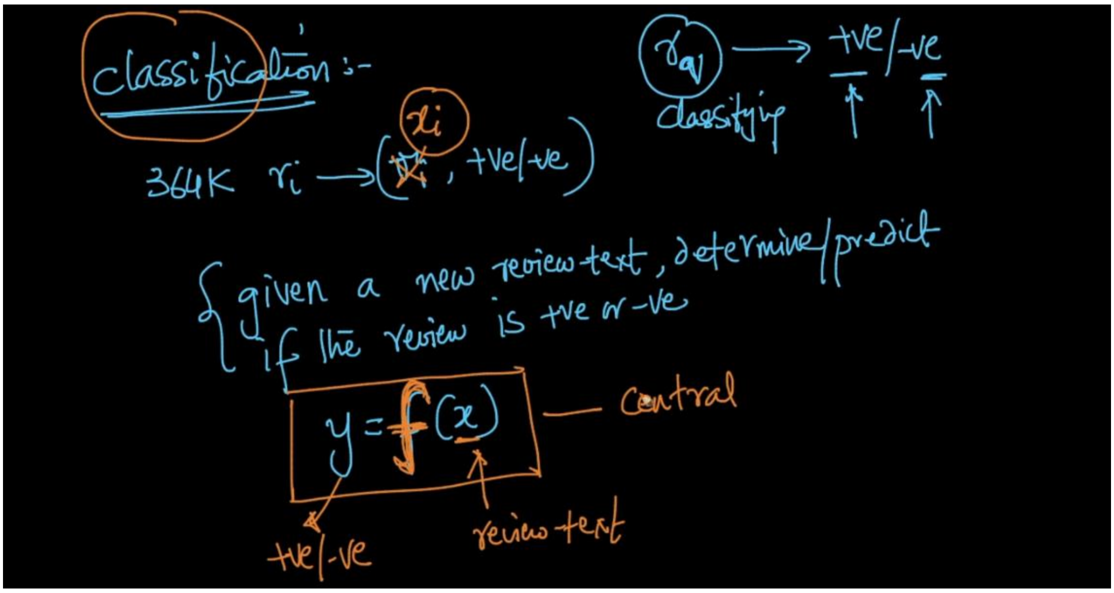
How classification works :

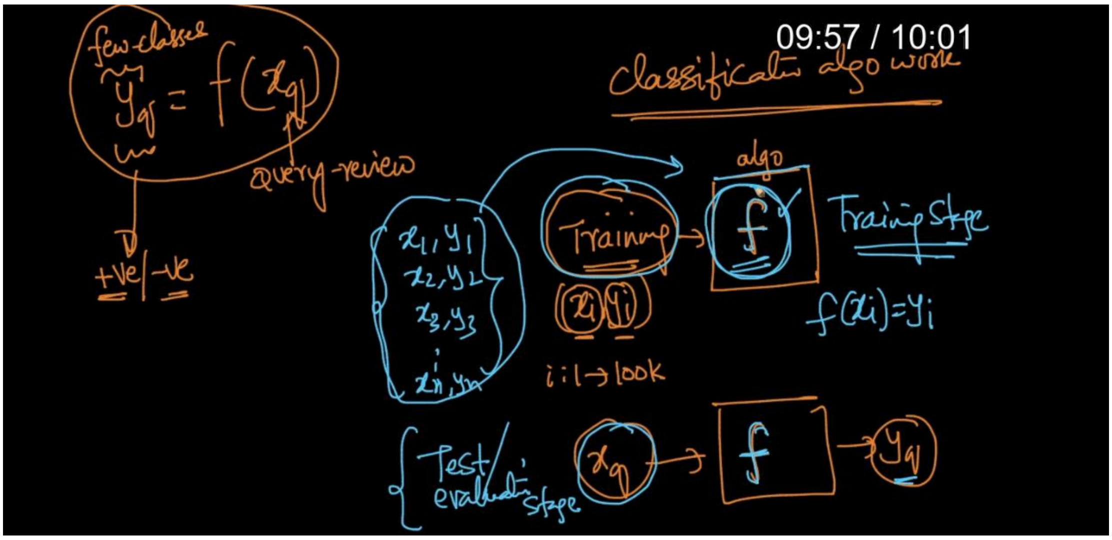
Classification is all about finding perfect function f

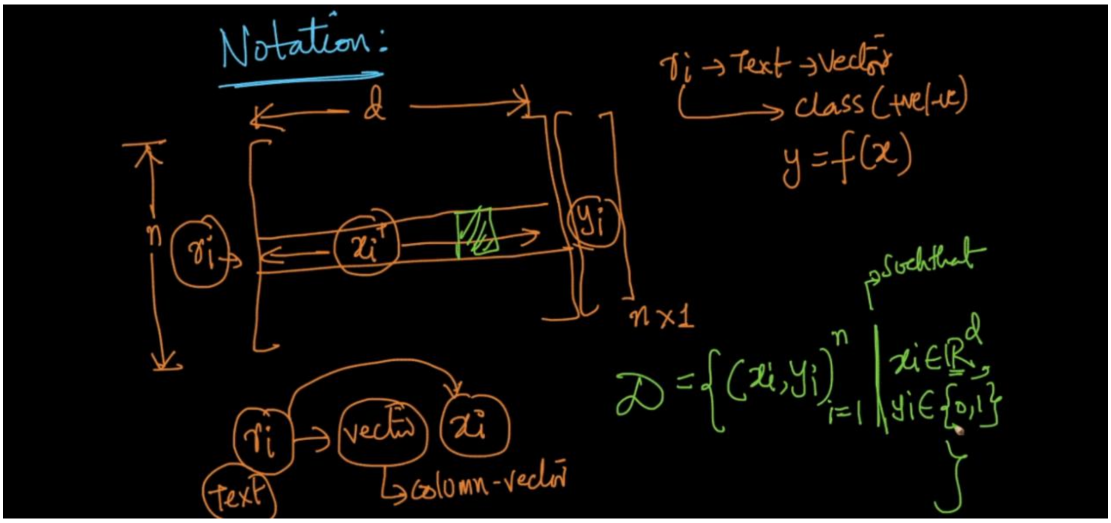
Y = f(x)

Where x is our new review Y is prediction which tell whether review is + or -

So classification is just about finding perfect function F .



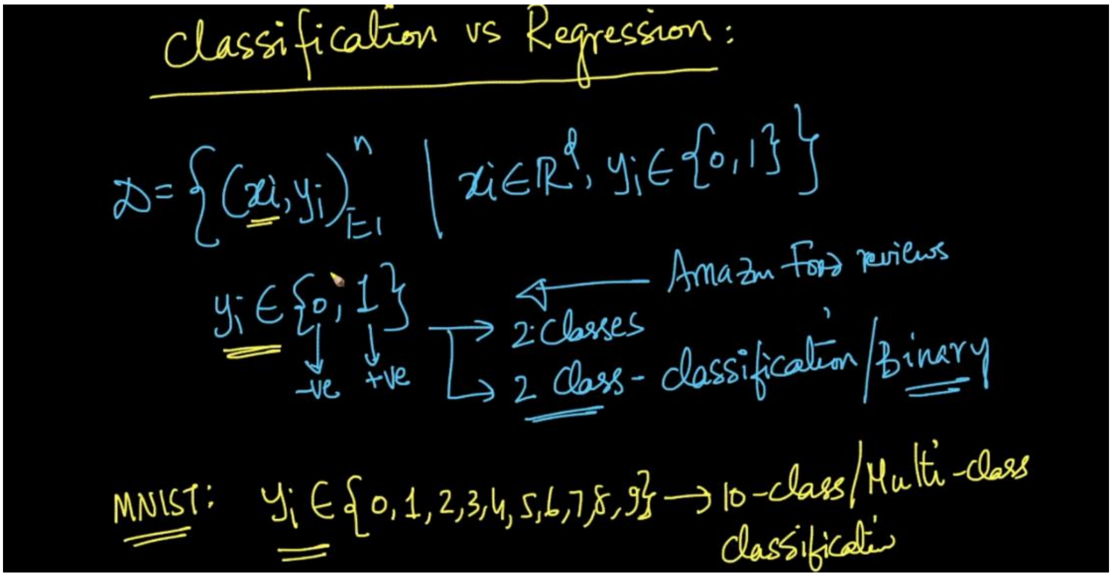


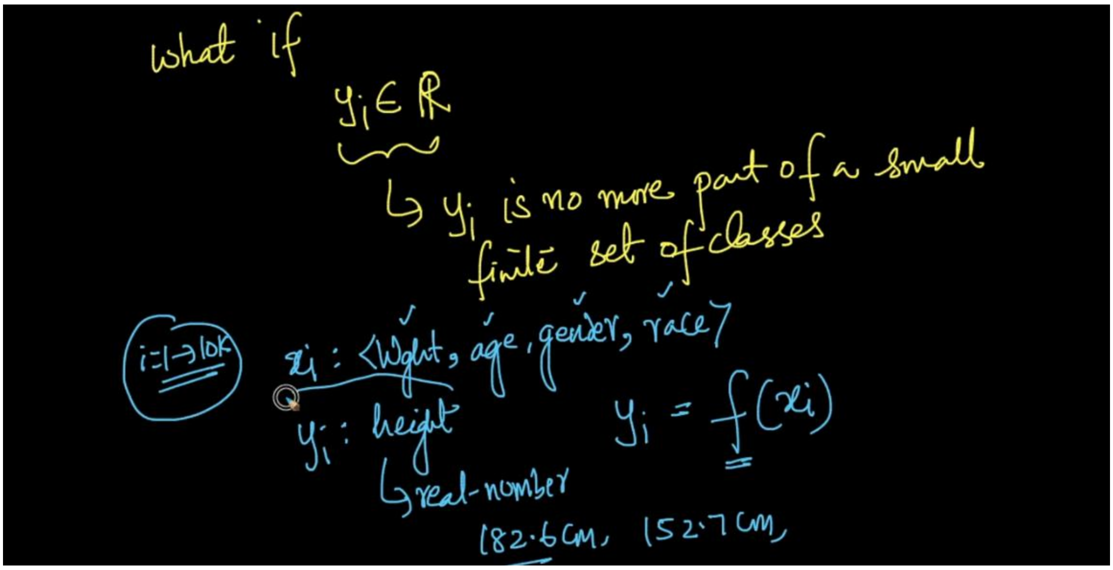


Classification Vs Regression :

In classification we have finite class like + , -
Or {0,1} or multiple class but not a real numbers .

Regression we have real number if Y belongs real number then it is called regression problem .



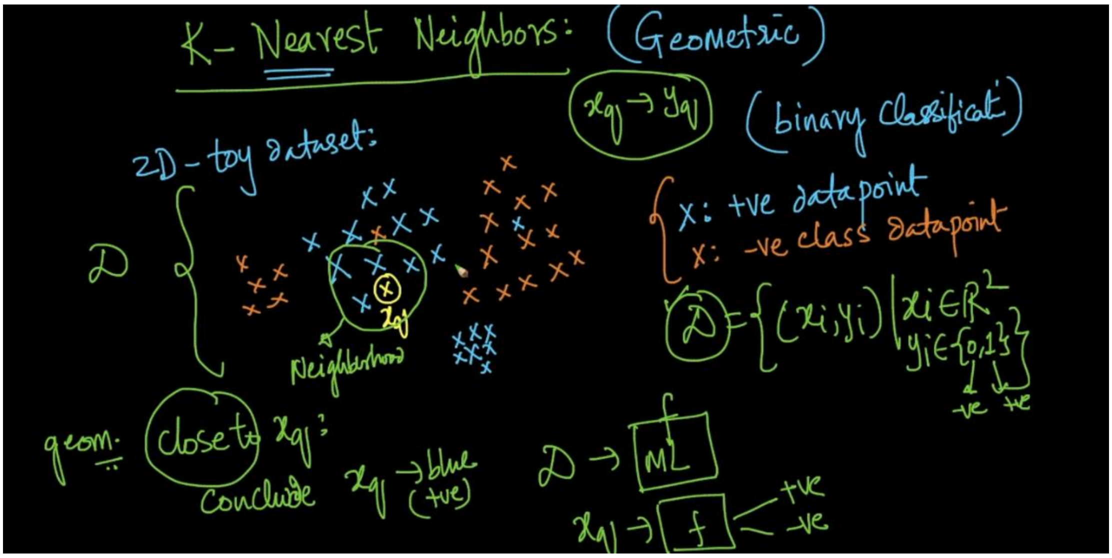


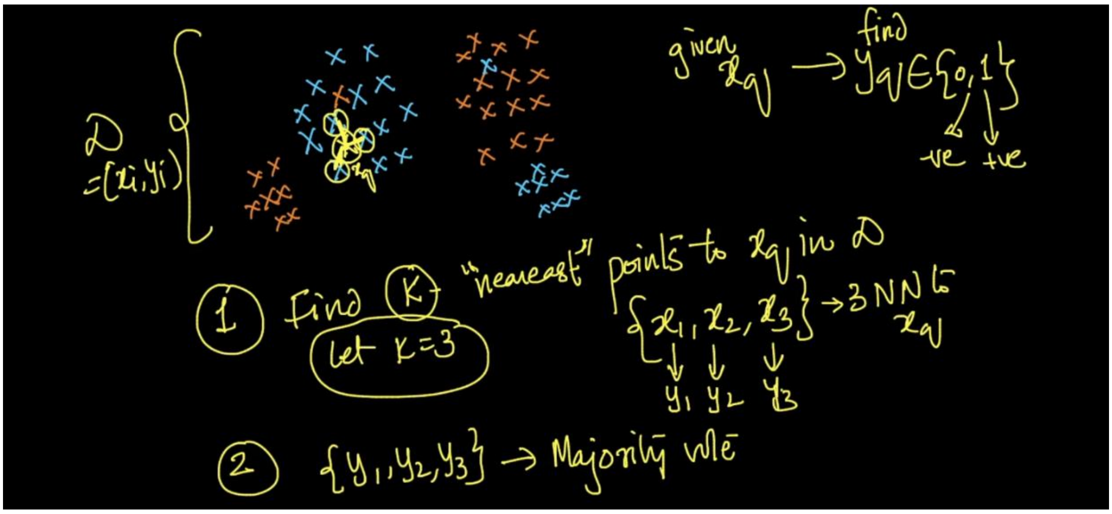
K Nearest Algorithm :

In this process we will find k nearest point for new point and will take majority voting .

We avoid k even because it will create confusion lets say we have k = 4

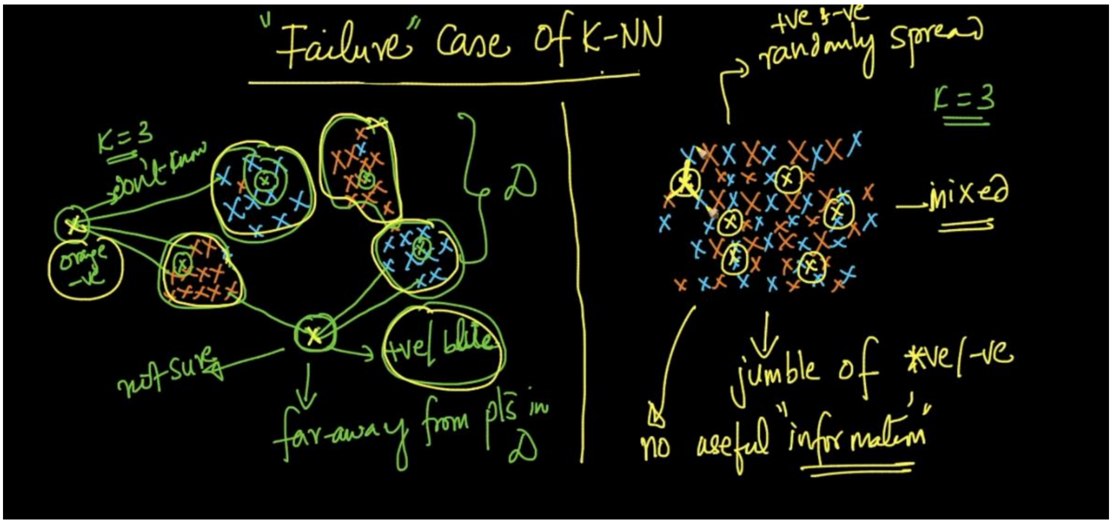
So we have now x1,x2,x3,x4 and if 2 are + and 2 are - then how can I predict new point because both have same probability .





KNN Fails :

When data point is to far from our data set we cant be sure to predict .
If data is randomly spread not well separated then also .

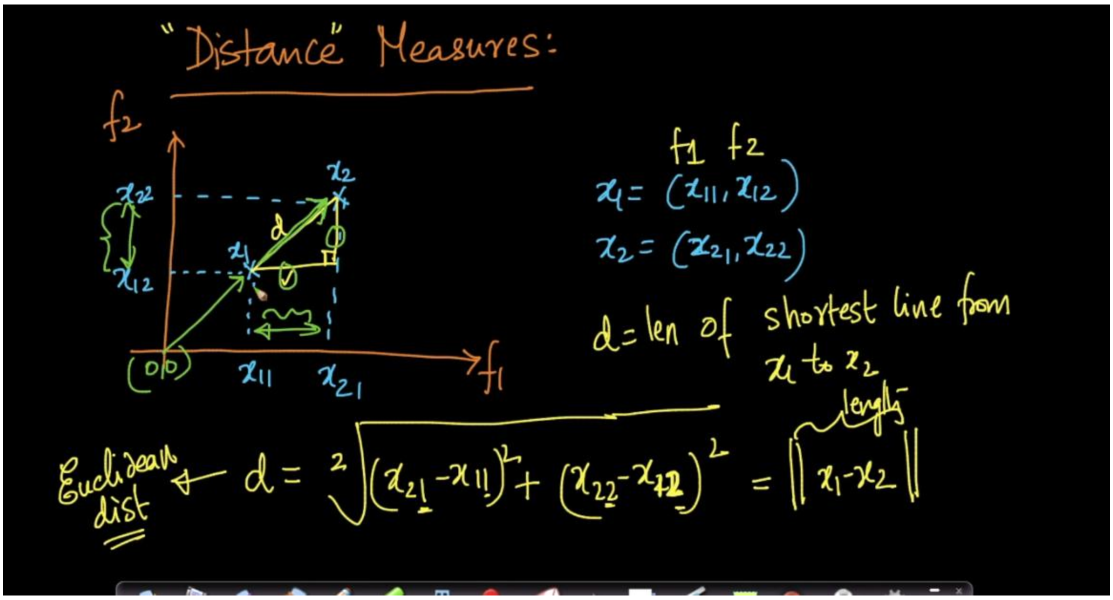


Distances are for 2 points and norms are for 2 vectors .

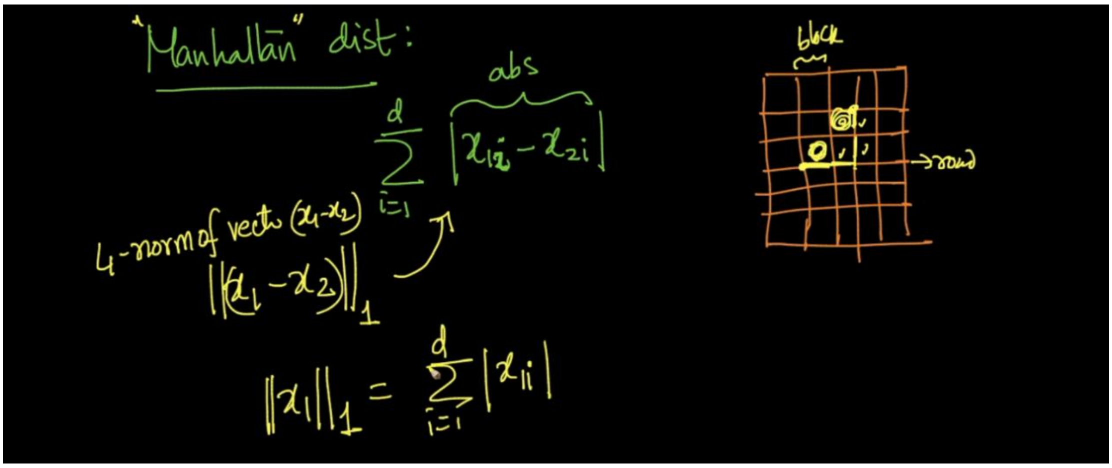
Distance :

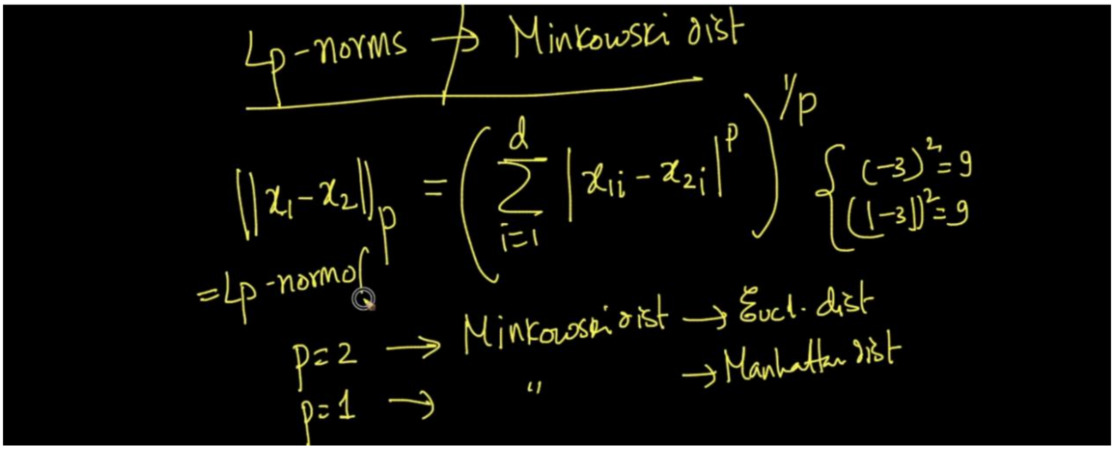
E distance between two we simply use Pythagorus therm .

For E distance we have 2 norms

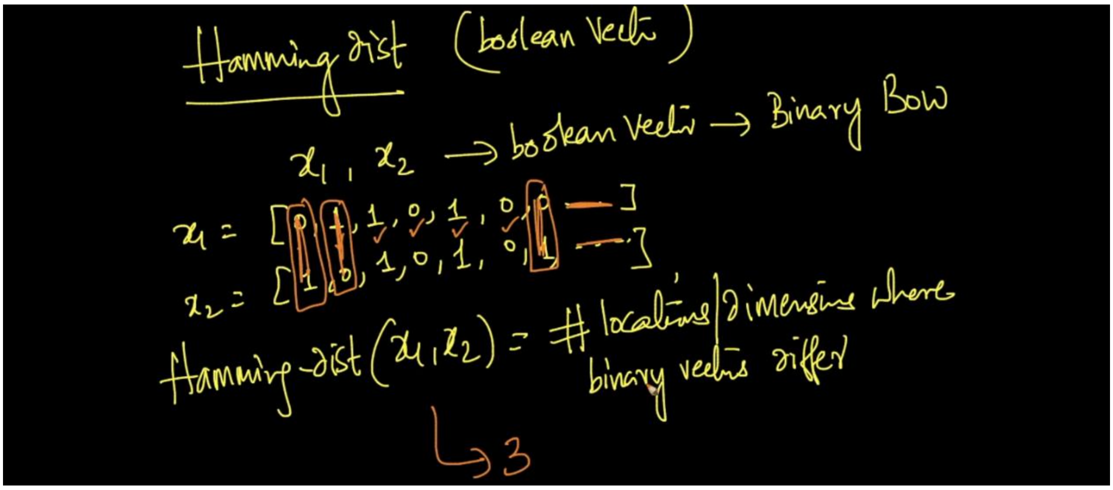


Manhattan Distance we use same formula just we remove square root . and will do sum of points .

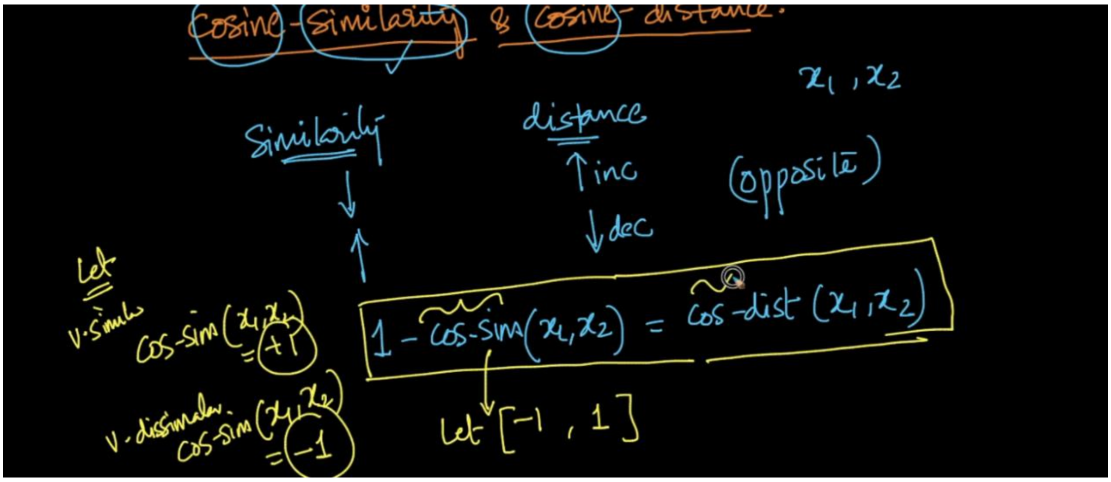




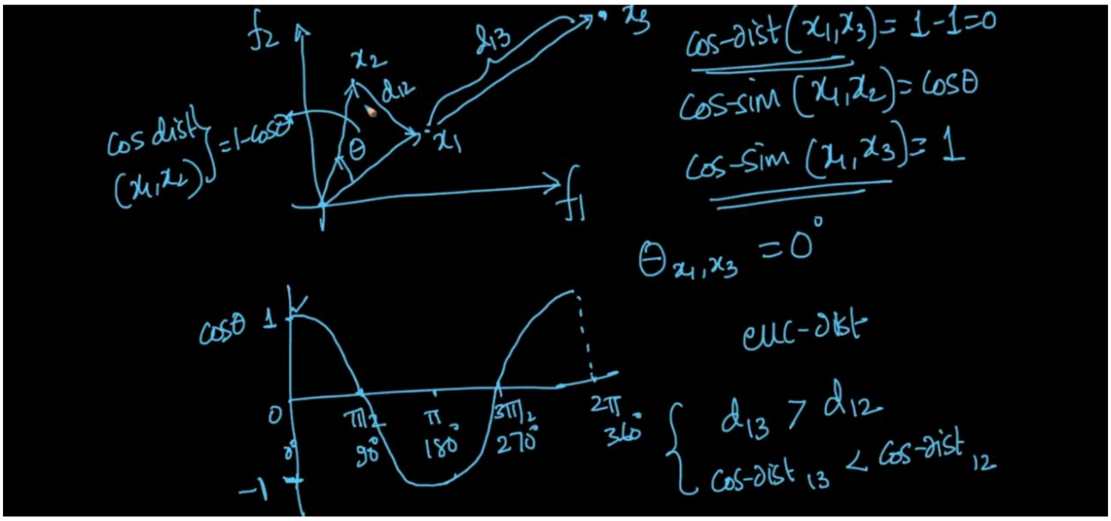
Mink distance is same like E distance if p =2 and if p=1 then it is Manhattan distance .



Hamming distance used to tell difference where or how much place 2 variables are different .



See above formula for cos similarity .
Lets say cos similarity lie between -1 to + 1 if they are very similar then cos -sim = 1 if very different then -1 .
This is rel between cos -sim and cos -distance ..



Difference E dist. And Cos :

See above image lets say we have 3 vectors x1,x2,x3 .

X1 and x2 are in same direction so angle between them is 0 .

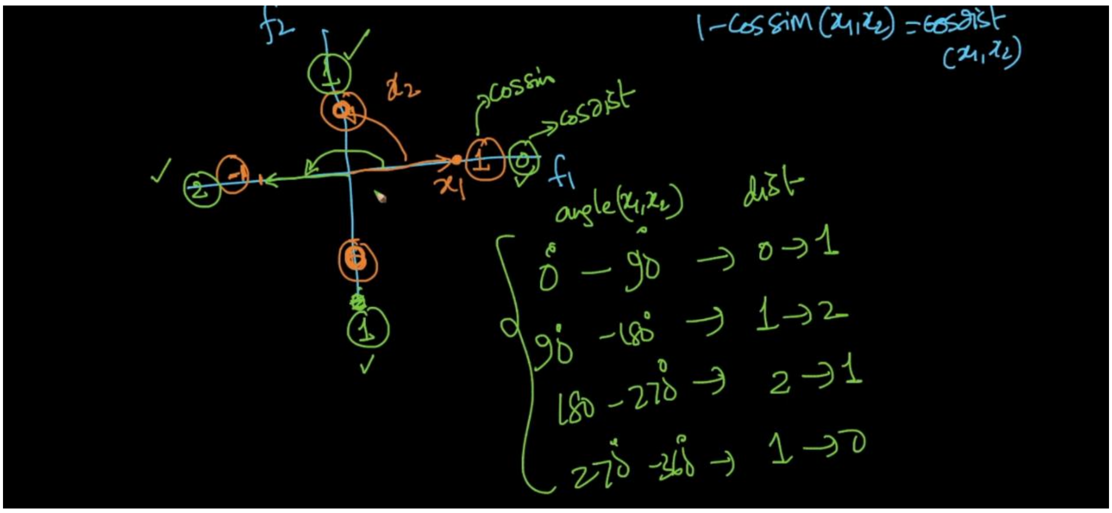
d

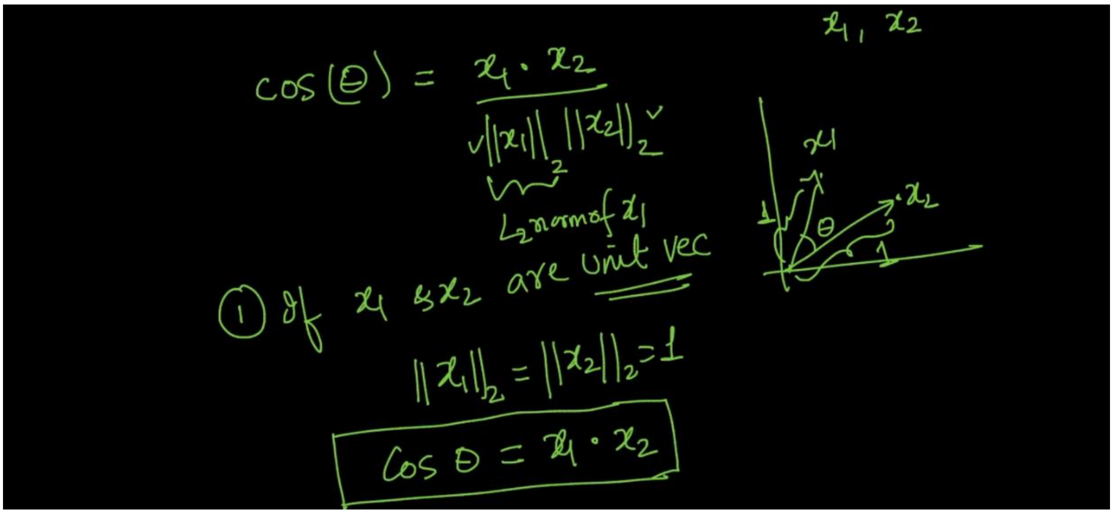
But as per E dist we consider that point is too far than x2 .

Here cos(0) = 1 means x1 and x2 are very similar .

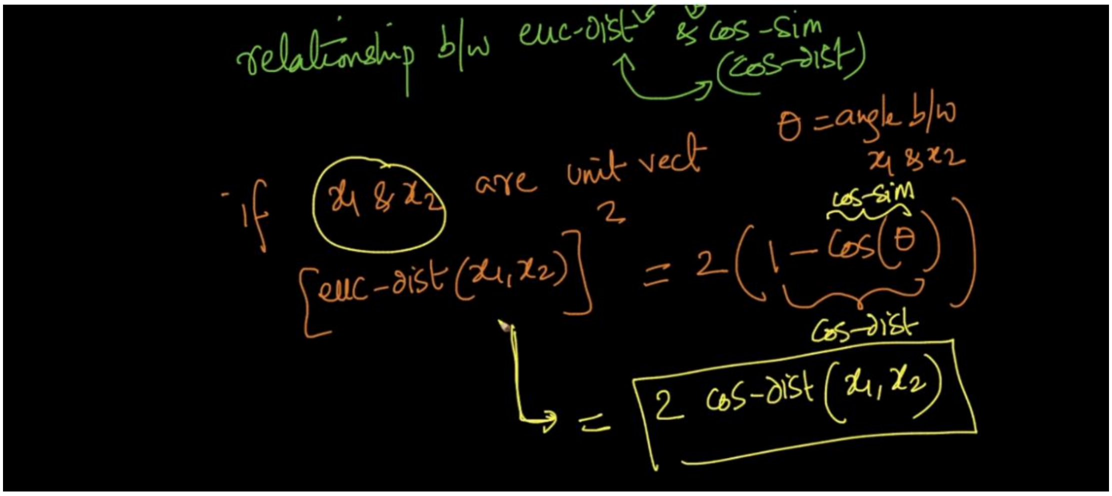
See the bottom of image E distance is very high between x1 and x3 than x1 and x2 .

But as per cos distance between x1 and x3 less than x1 and x2 ..



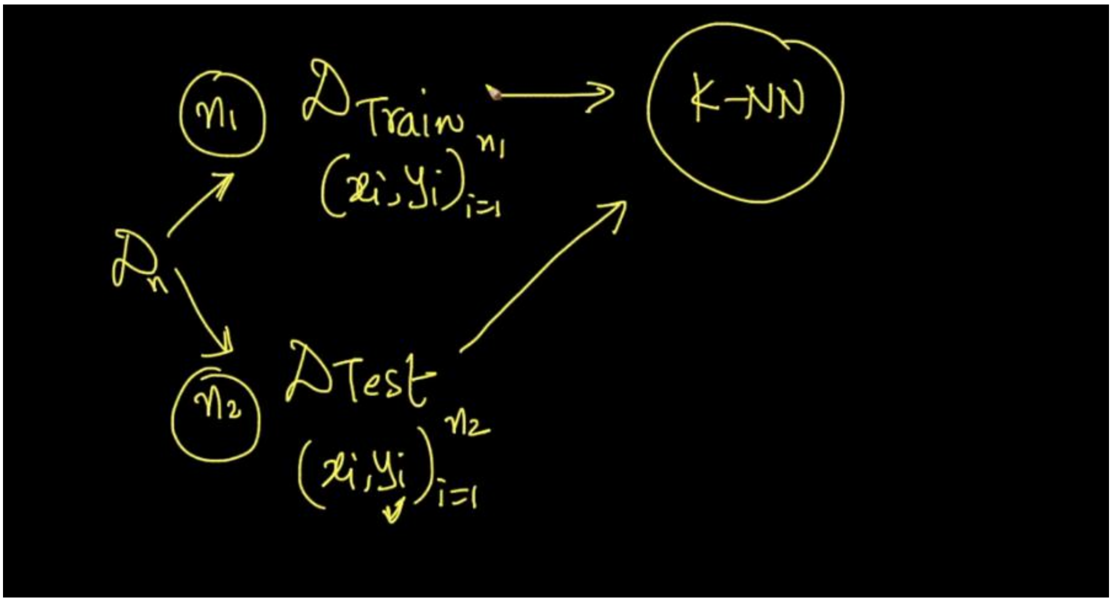


If the x1 and x2 are unit vector then relation between E distance and and cos is below :



Means E distance is 2 times cos distance . only if x1 and x2 are unit vector .

How KNN works :



Lets say we data set D . We will split it into train and test .

N1 = Train data = 70 %
N2 = Test data = 30 %

Now will train our N1 using KNN so after train we get trained model .

After that we use N2 data and will test it on our trained model so that model will predict Y’ .

Now for that test data we have already all values original Y .

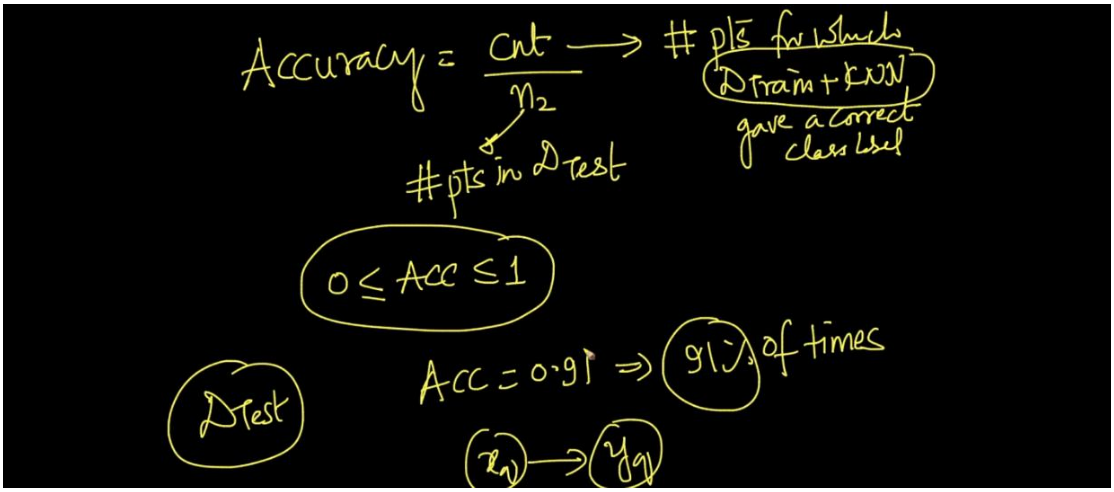
We will compare Y’ and Y if both are same then will increase variable :

count +=1 .

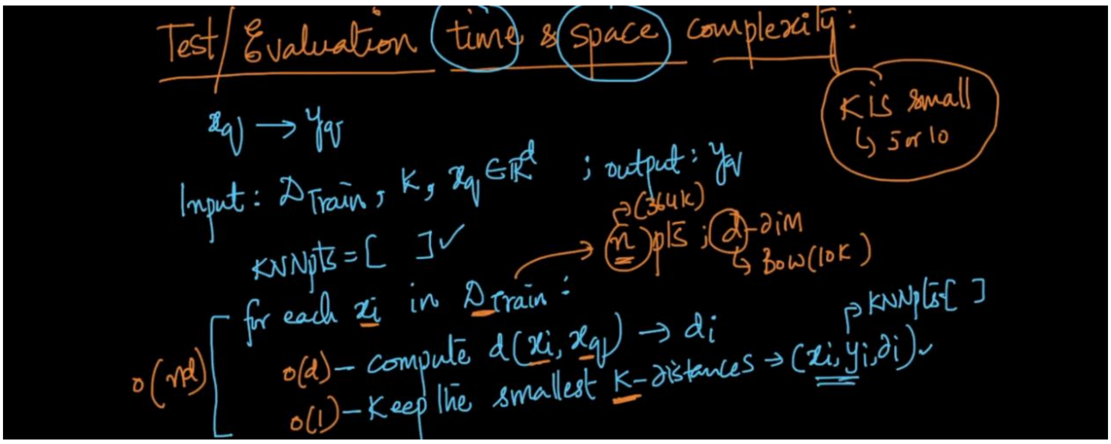
They both are equal means our model predicted correct answer .

Simply after that we will cal accuracy by :

Count/ N2



Time and Space Complexity for KNN :



Lets see above image . now we have n data means for amazon we see we have 364k data points .

And each data points is nothing but a vector of d dimension (BOW ,TF IDF of review) .

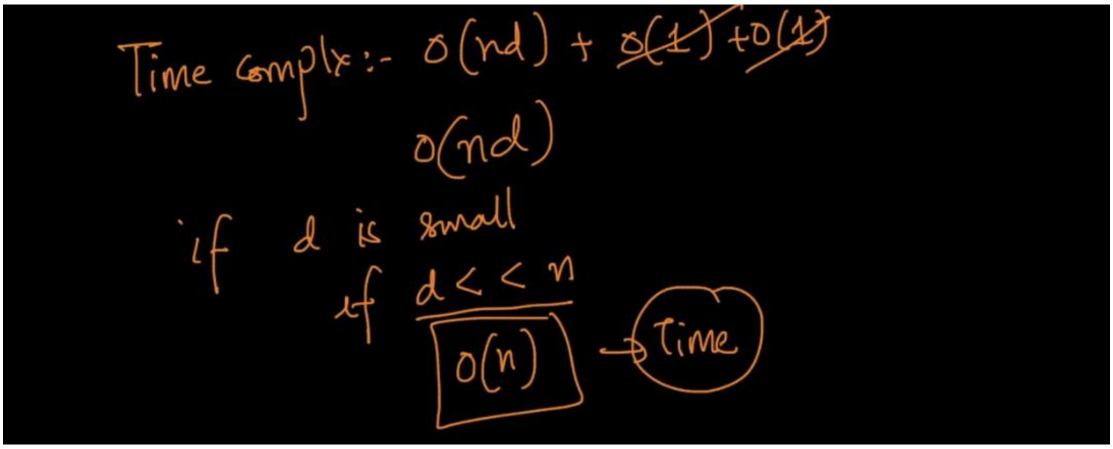
So now we get train data ,k values ,and new review to predict .

So our algorithm will run for each data point and will find distance between xi and new review(X q) .

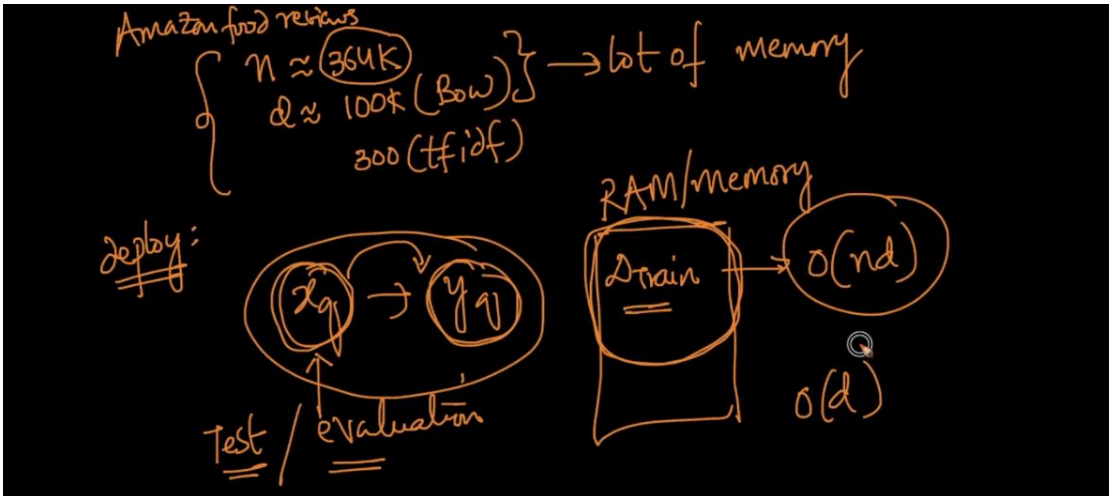
Then it will keep smallest distance like xi,y I ,d I .

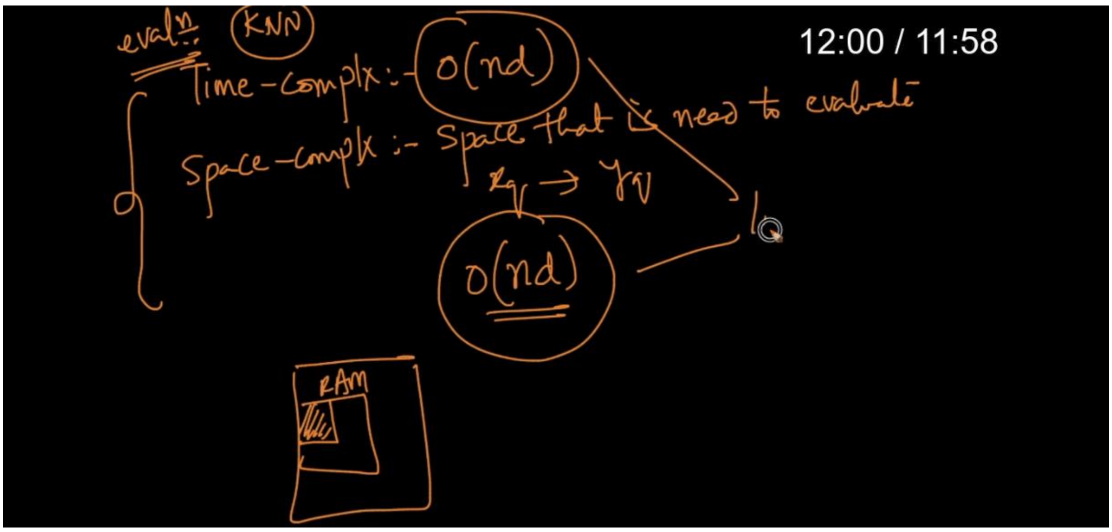
So Time complexity for each data point cal :
o(n*d)

Because we need to run loop n times means 364k time and d means dimension of vectors .



Space Complexity for KNN :



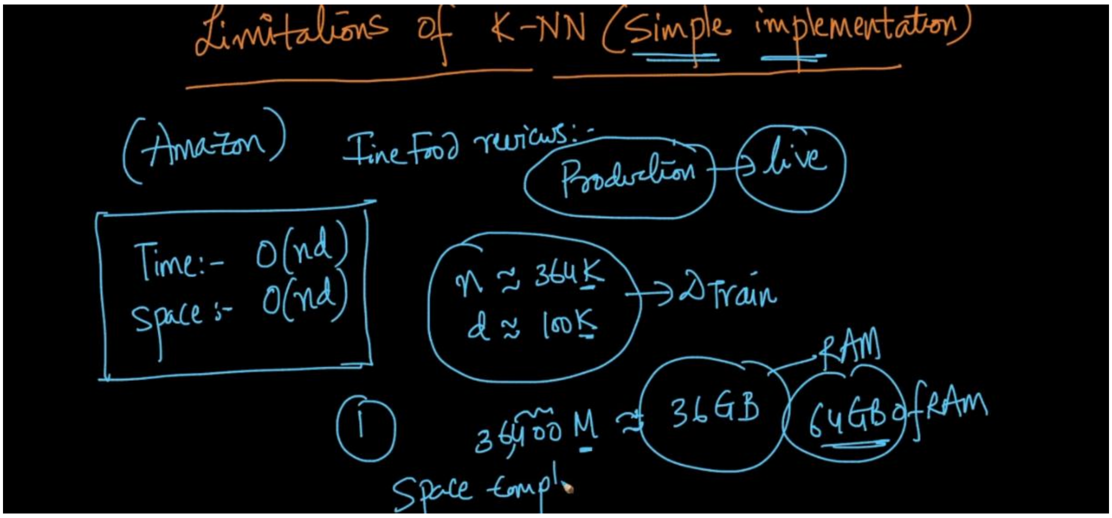


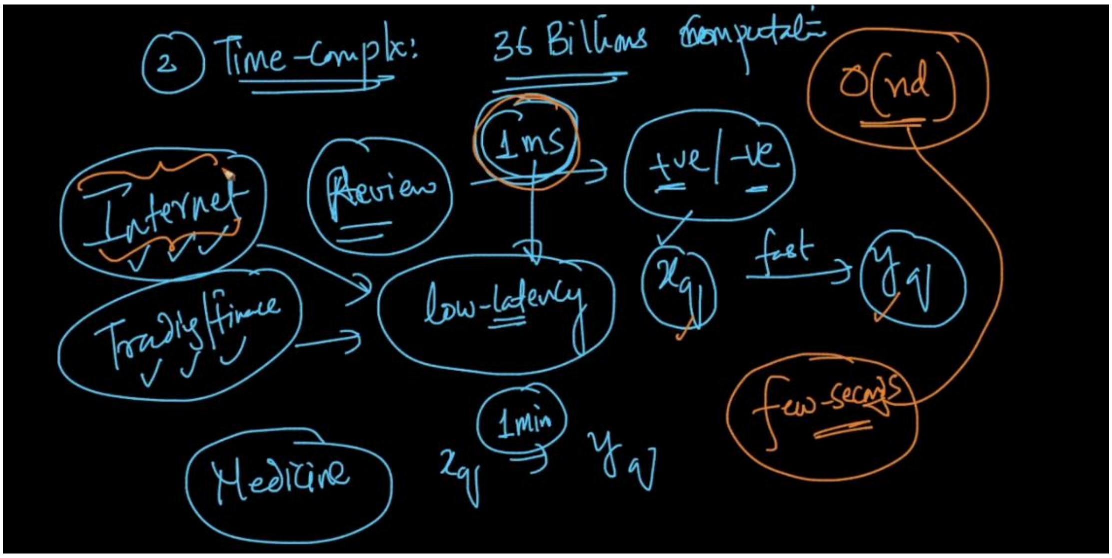
See above image space complexity means total space take by algorithm to give output .

In our amazon case we have
N = 364 k
D = 100k

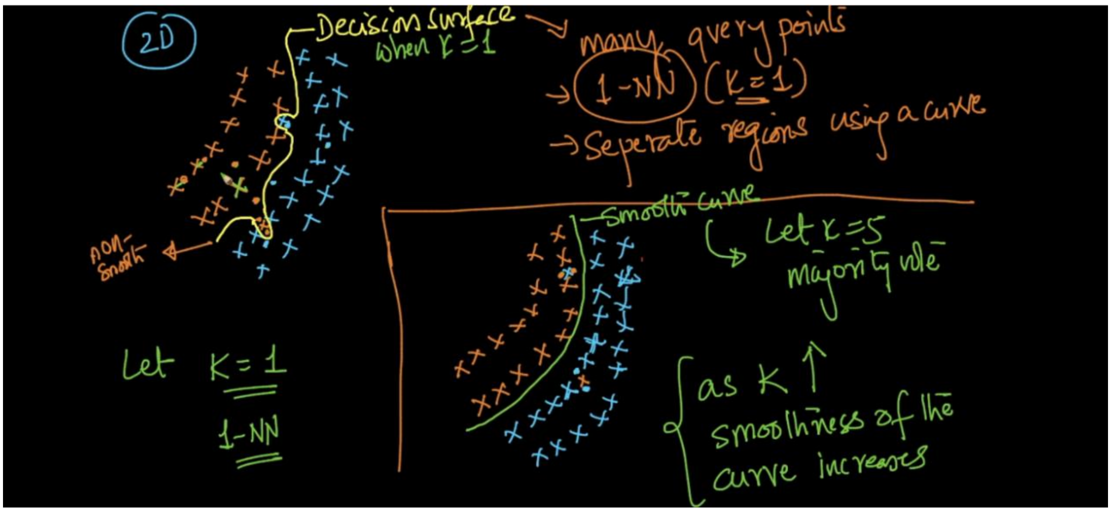
So while run algorithm we need to store all train data means 364k data * 100k dimension into our RAM it means it will take huge space .

O(n*d) In real world we use millions of data in that case it will consume huge huge memory and this is very worst condition .





Because of above 2 issue KNN not used widely . In real world we need response within ms . it will take too much time by KNN . That's why time and space comp are very very important while deploying ml model to production .



If k =1 graph will not too smooth as seen in above image .

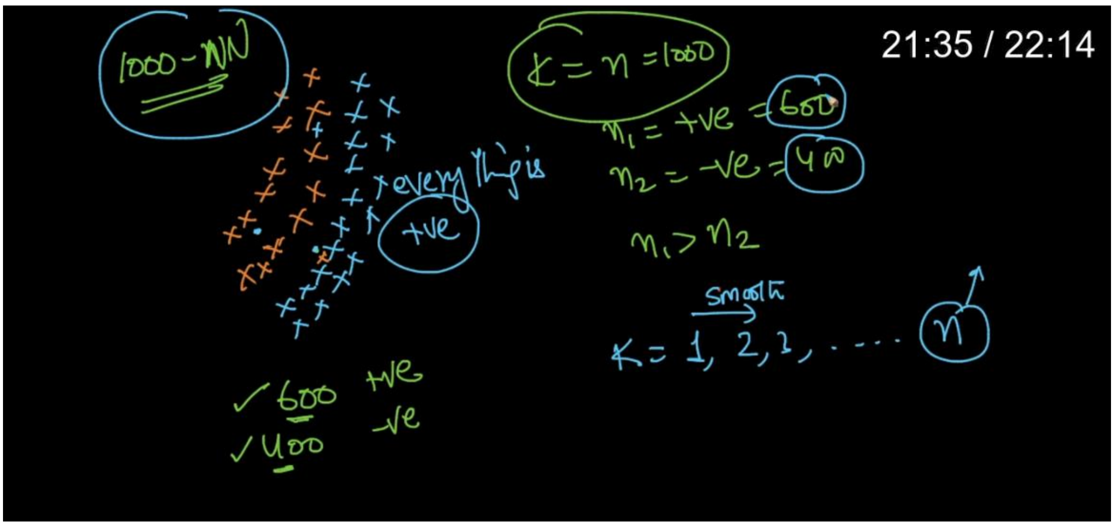
As k increases we will get smoother graph .

If we make k = n then whatever majority of data each time it will predict same .

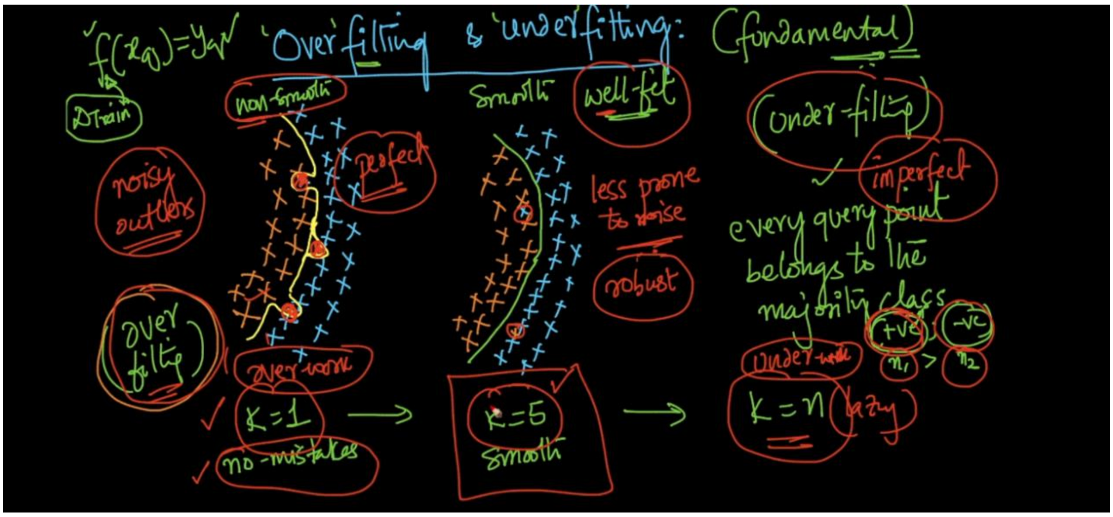
See below image we have n =1000 review .

600 are +
400 are -

Now if we keep k = 1000 then in each case our model will predict any new review as + because majority of data is + ..



Over fitting and Under Fitting :



Under fit because of very lazy model who didn't care about anything just it say its positive review .

Over fit will try to be very perfect no mistakes so because of that after train it will give wrong prediction .

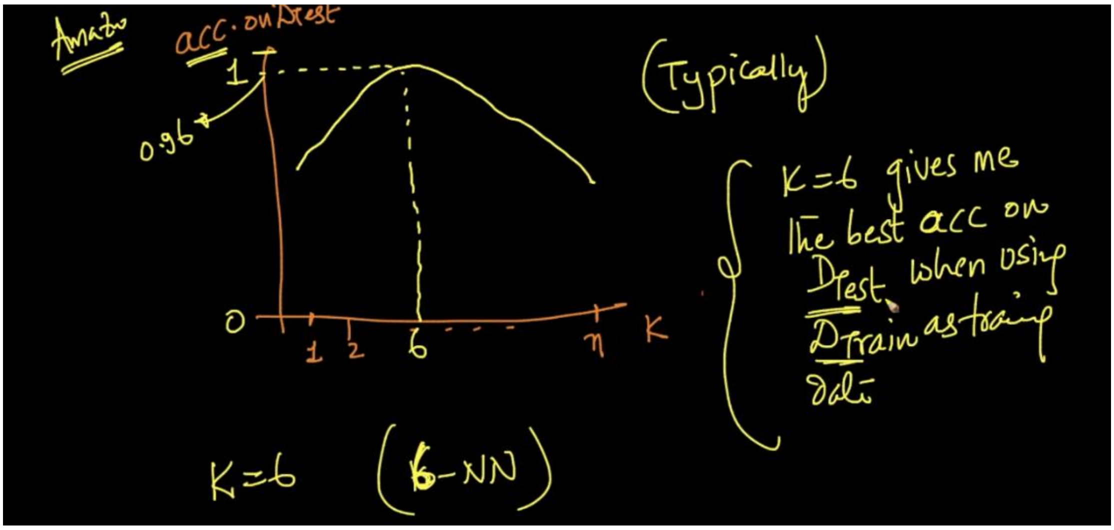
Cross Validation :

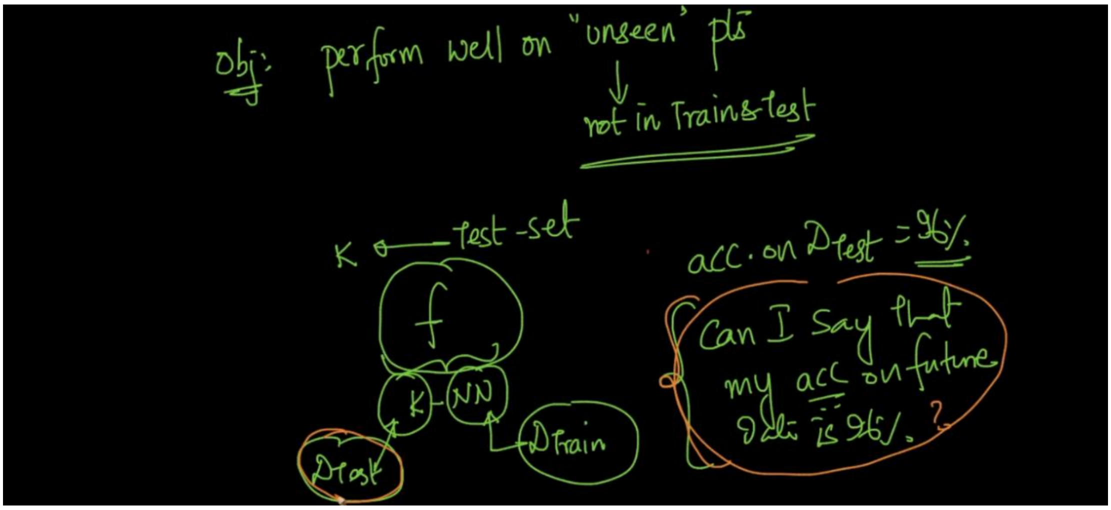
Now lets say we have Data D we split into train and test .

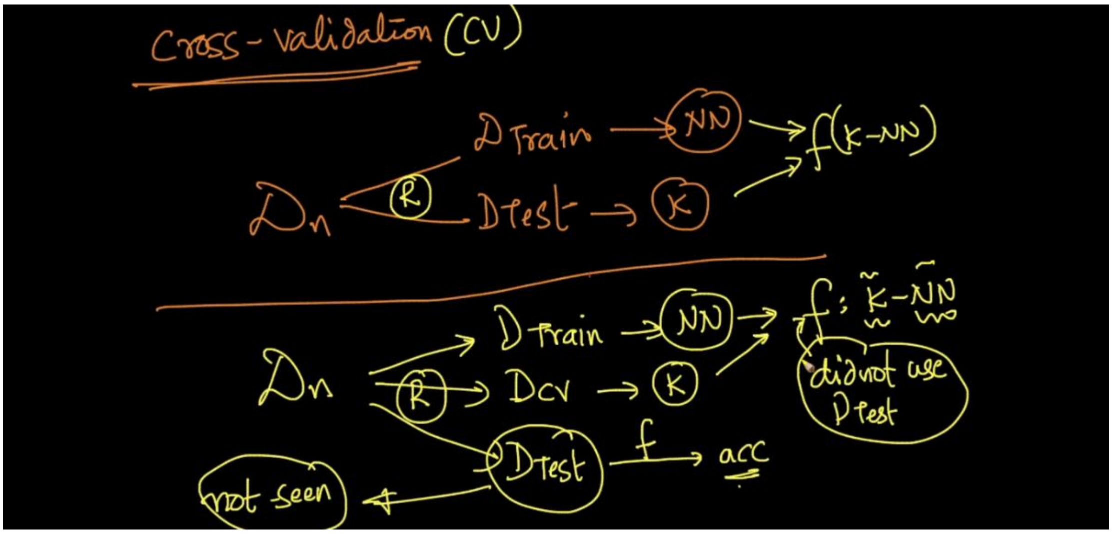
Train data we used to find nearest neighbour (NN) as we discuss above in KNN . we find nearest values for new data point or review and then will find nearest n for each data point in train data .

After that we use that trained model and will test (Test data) for value k =1 …. n and we found that with k =6 we got 96% accuracy . but just doing this I cant tell that I will get 96% accuracy on future data .

To solve this problem cross validation used .





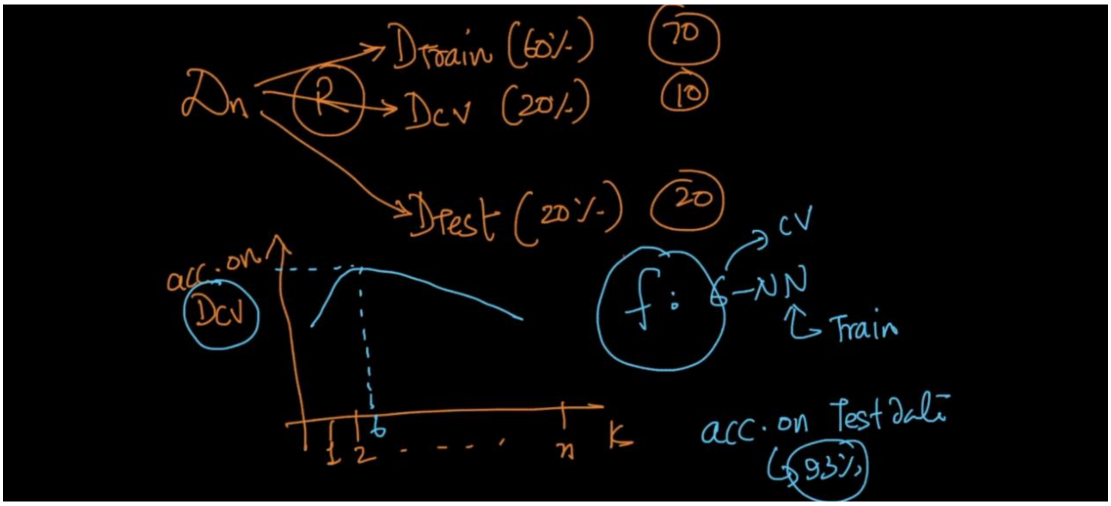


So using CV we split data into 3 parts :

1. Train
2. Test
3. Validation set

So now we find NN using train data we will find K on Validation data , and finally we test our model on test data which is totally unseen data .

After that we can say I got x accuracy using some k values and we will get same accuracy on future data also ..



K - Fold Cross Validation :

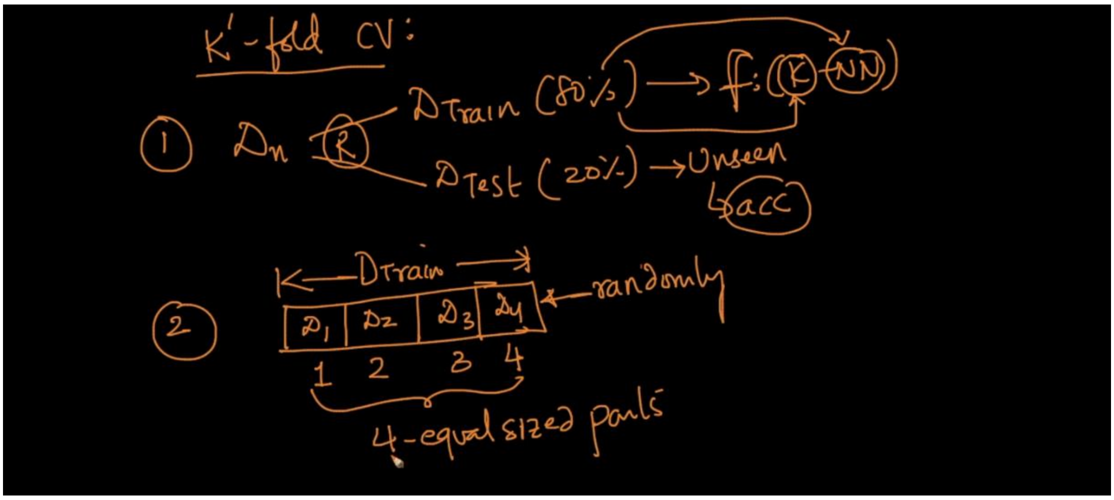
Now we see above CV but problem with that is we are using only 60% of data for training and that's not enough .

So we cant use test 20% data so is there any way we can use at least 80% of training data then ans is yes .

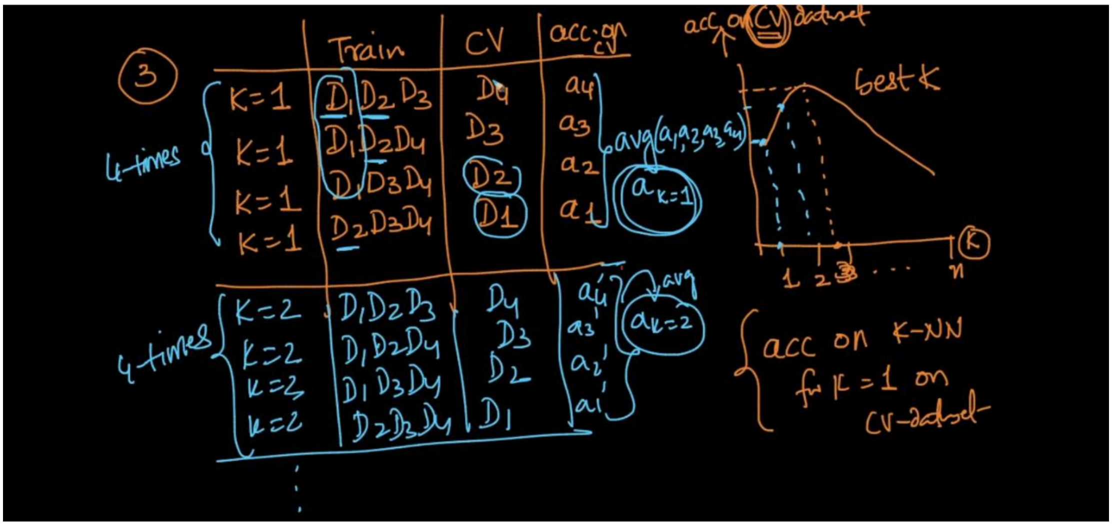
K - fold method used for that what we do here ?

Simply we split data :
Train = 80 %
Test = 20 %

Now we will again split train data randomly in 4 parts so we have now d1.d2,d3,d4 .



Now we do same K NN process to find perfect K like we start from k =1 …. n and we pick that K which has high accuracy .

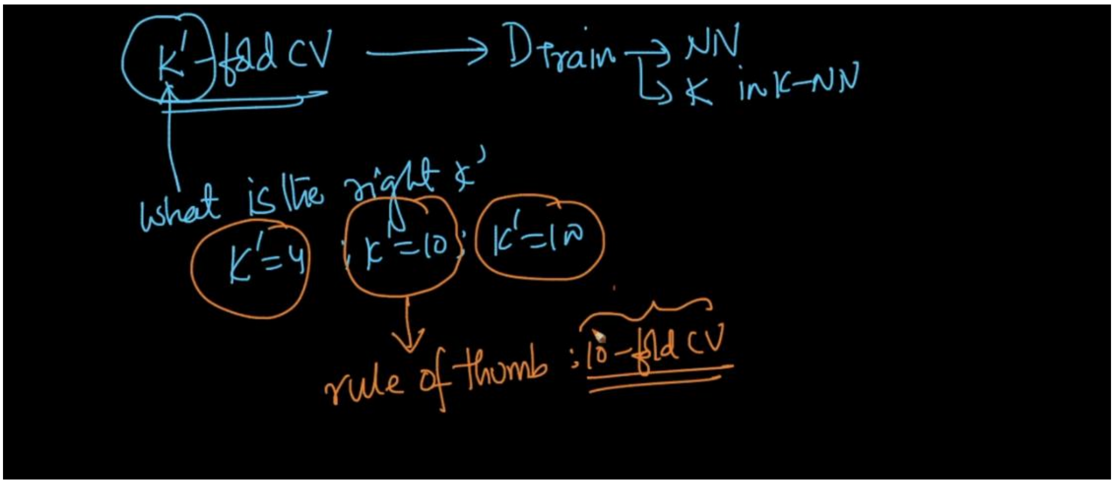


What we are doing here is we are taking first 3 part d1,d2,d3 as training and d4 as Validation set keep k =1 then we just do same alteration see above image .

We will repeat for up to k =10 its a standard value rule of thumb there is no any scientific logic but its standard .

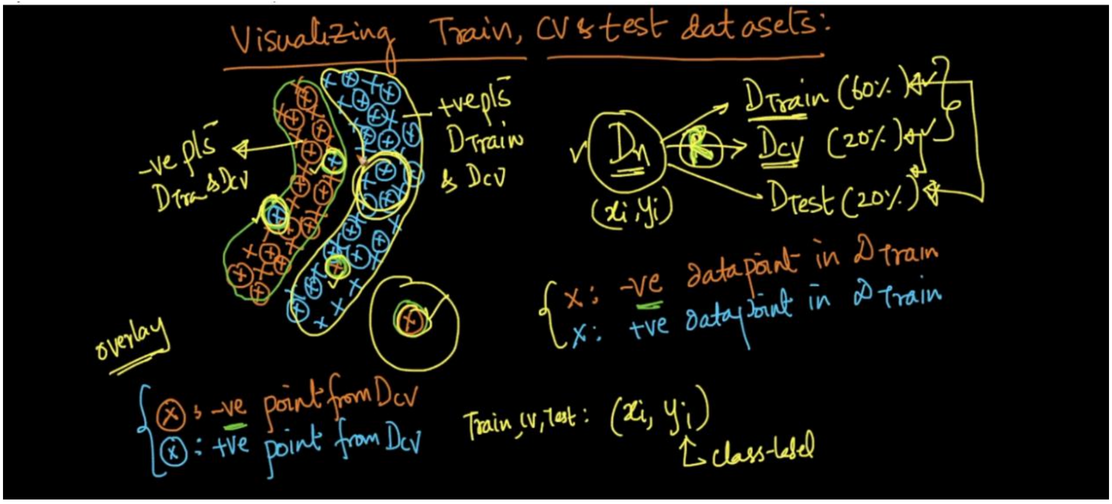
So in this way we are using all 80 % data for training .

So will use 80 % of data for finding NN .

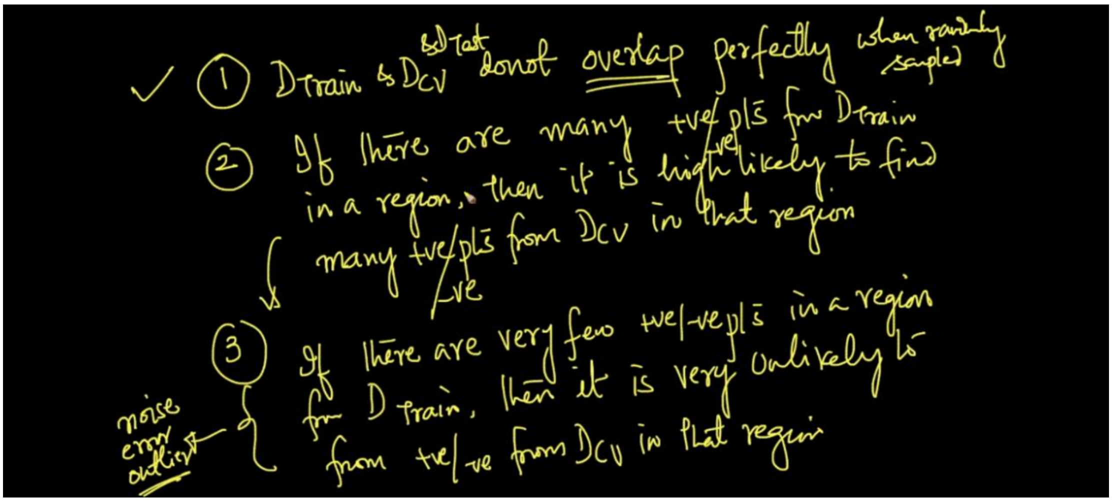


Problem with K fold is that time required for finding perfect k will increase k' times . because here we are repeating process k' times as compare to normal CV .

Means if K’ = 10 means 10 - Fold Validation and it will take 10 times extra time than CV . but still its best because its one time effort only .



When we randomly sample data train test CV then don’t expect that all points don’t expect always they are same u will surely find some outliers,error points in data set see above image ..



Train and Test Error Under fitting Over Fitting :

Now we know that we train model on train data means we find NN using train data .

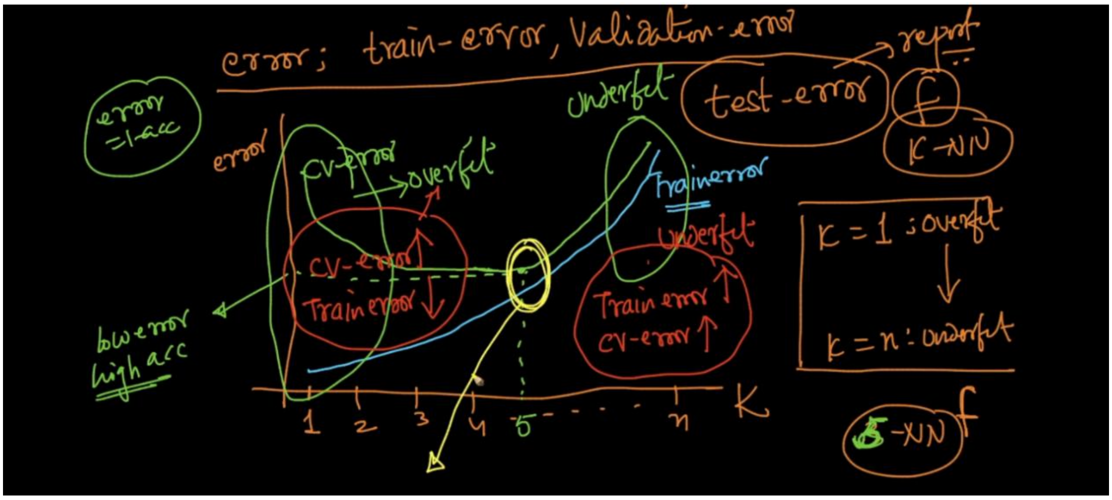
Now if we use our trained model against of train data then we will find what accuracy our model predicting on train data .

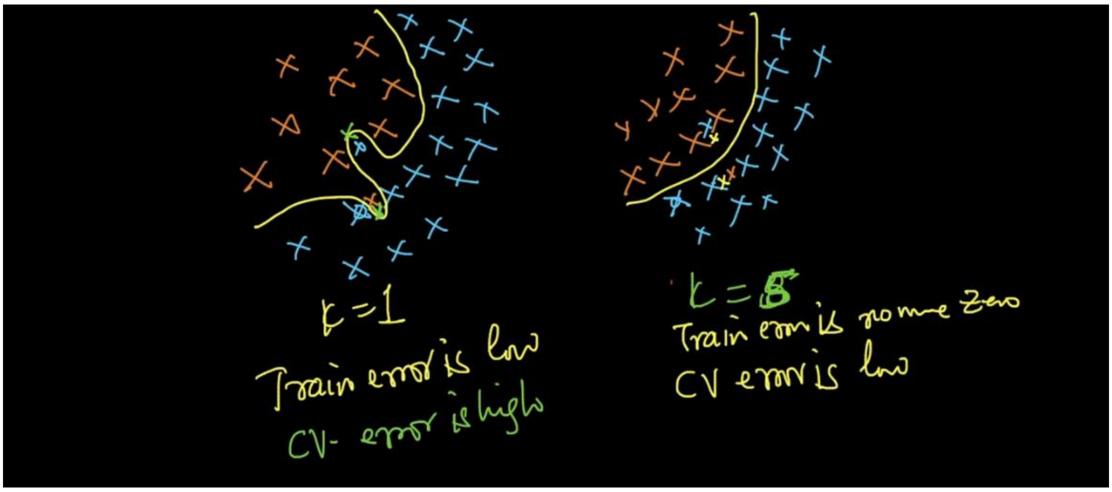
From that we find Train error :

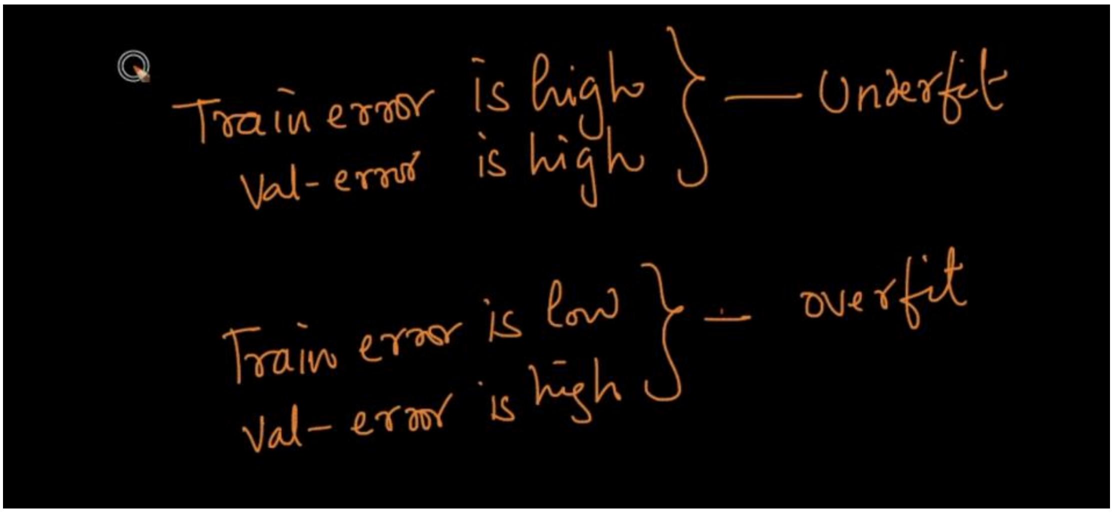
Train error = 1 - accuracy on train data

Similarly we find Error on validation set :

Validation error = 1 - accuracy on validation set







Time Based Splitting :

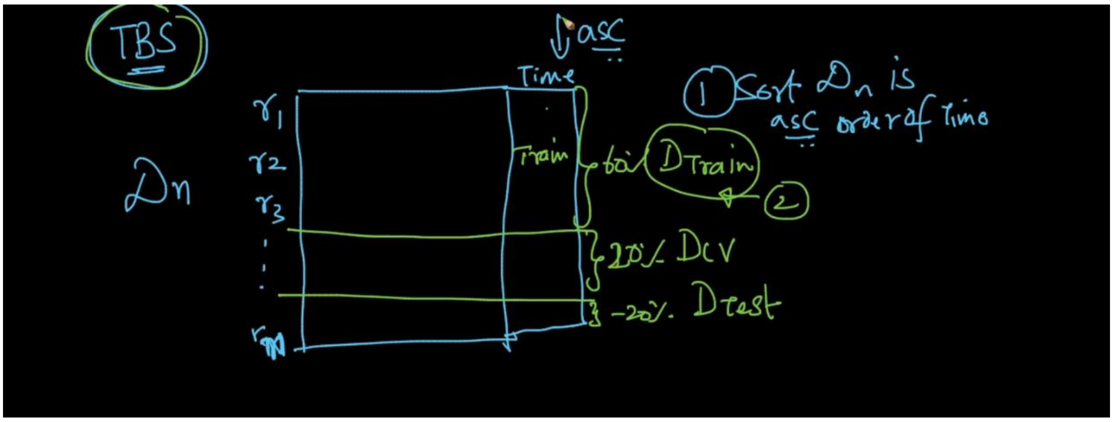
This is more preferable than random split for amazon review .

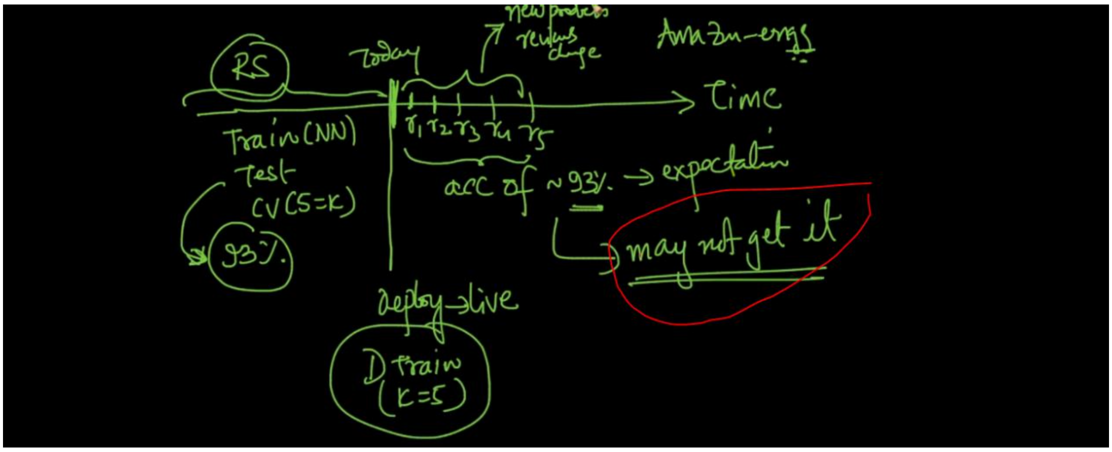
Because review are always change as time change lets say x mobile improved its quality then review will also change so we need time based splitting .

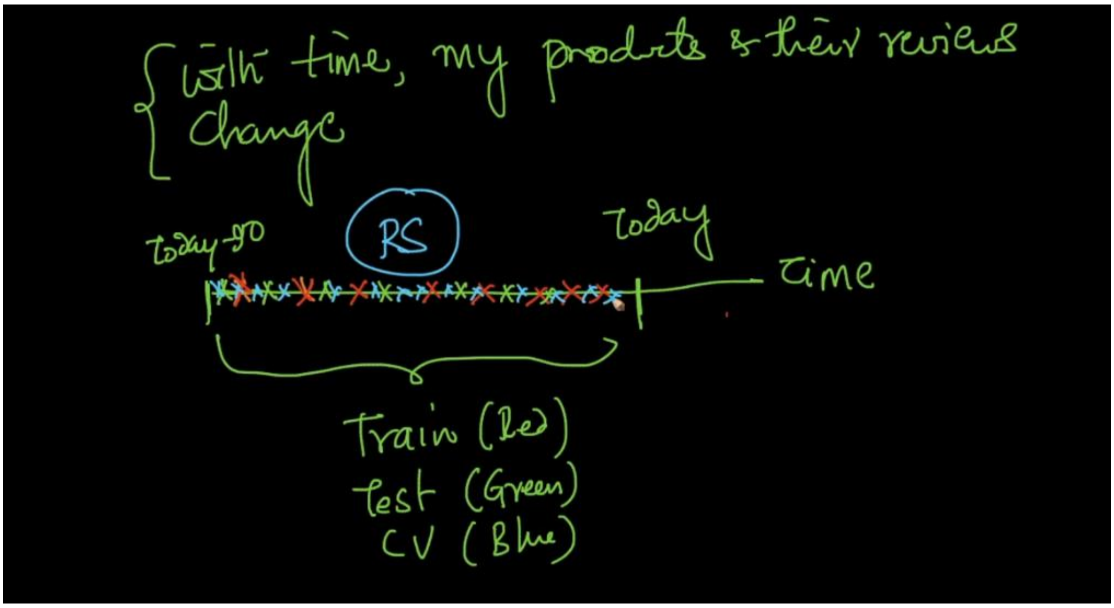
In that lets say we are using last 90 days data for train test validate .

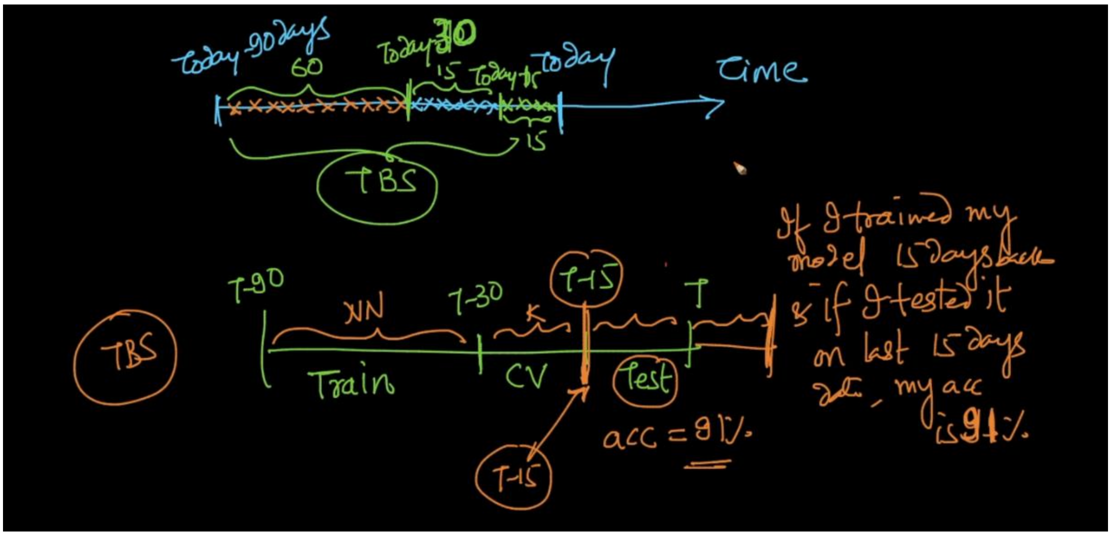
Then we use last 60 days data for train , 15 days validation and 15 days(latest) test .

And if we get lets say 90% accuracy on test then we can say that this will remain same for next 15 days . because we used latest data in random split we use any data from day 1 to 90 to test that's why its not good .

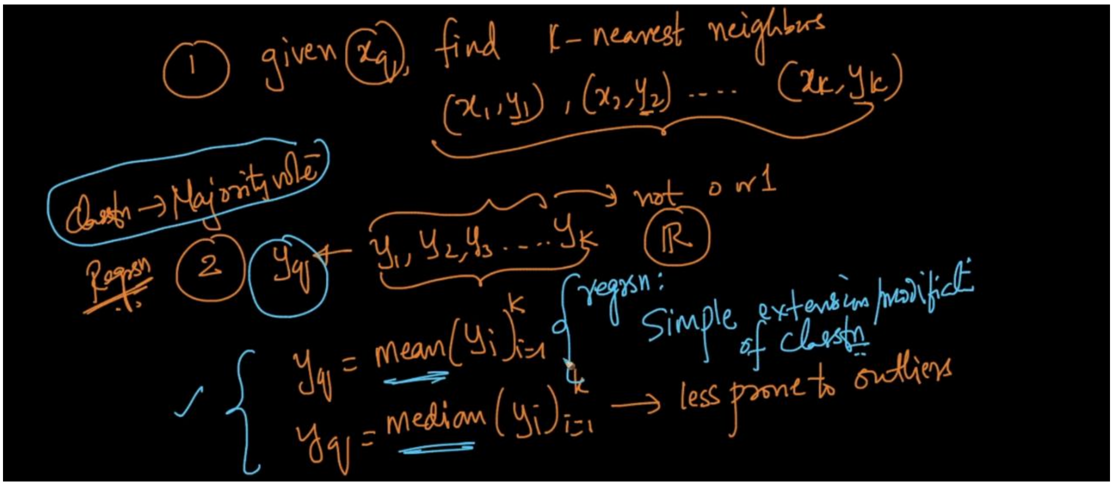



RS
Today
new products reviews change
Amazon-engg
Time
Train (NN)
Test
CV (5=K)
93%
r1 r2 r3 r4 r5
acc of ~93% → expectation
may not get it
deploy → live
D train (K=5)


With time, my products & their reviews change
Today-90
RS
Today
Time
Train (Red)
Test (Green)
CV (Blue)

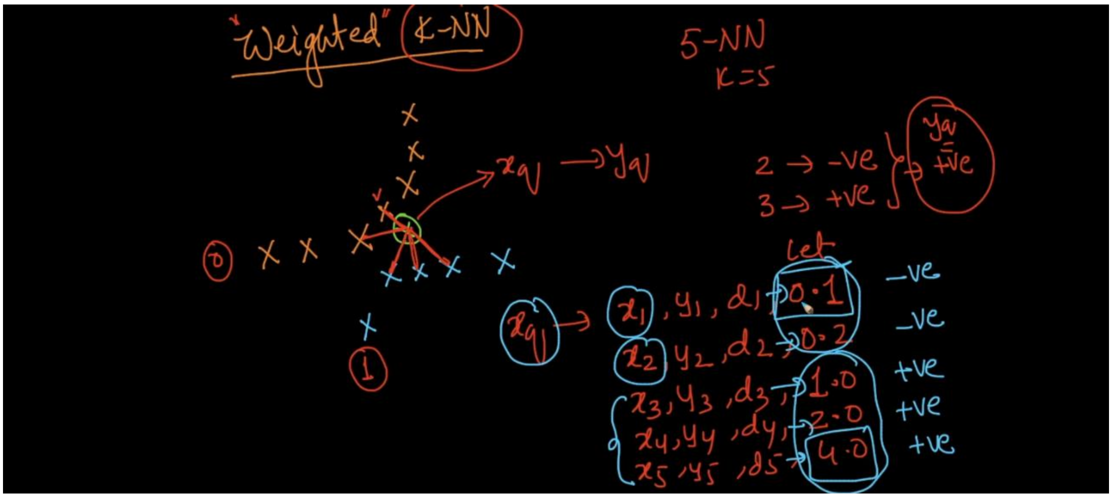

Today-90 days
60
Today-30
15
Today-15
Today
Time
15
TBS
T-90
NN
T-30
K
T-15
T
Train
CV
Test
acc = 91%
If I trained my model 15 days back & if I tested it on last 15 days data, my acc is 91%.

KNN for Regression :

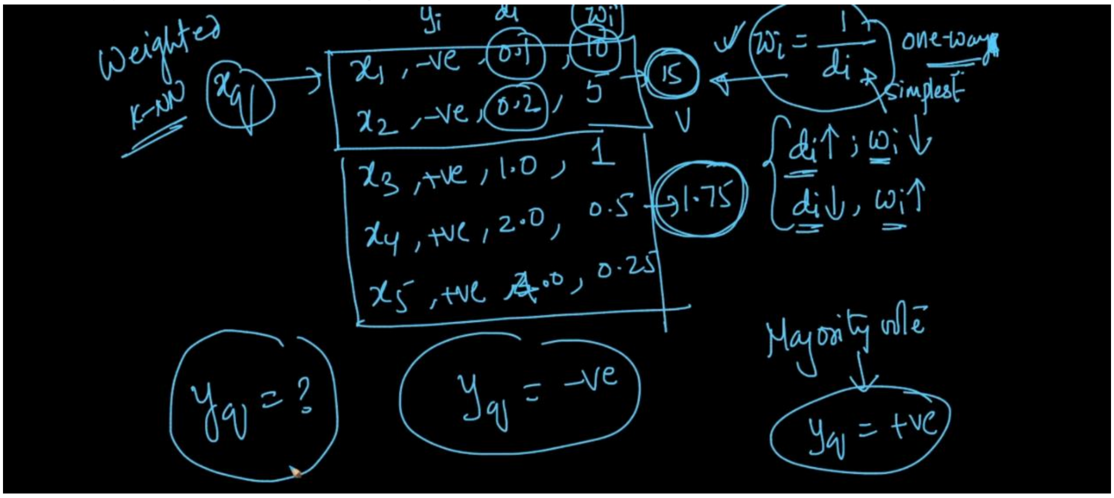


Here also we find nearest N and then will take mean or median of all y to predict new Y .

Weighted KNN :



See above image lets say we have k = 5 now we took 5 values from that 2 are - and 3 are + so per KNN rule we took majority vote hence we can say new point belongs to + class .



So here we took weighted avg

W = 1/d

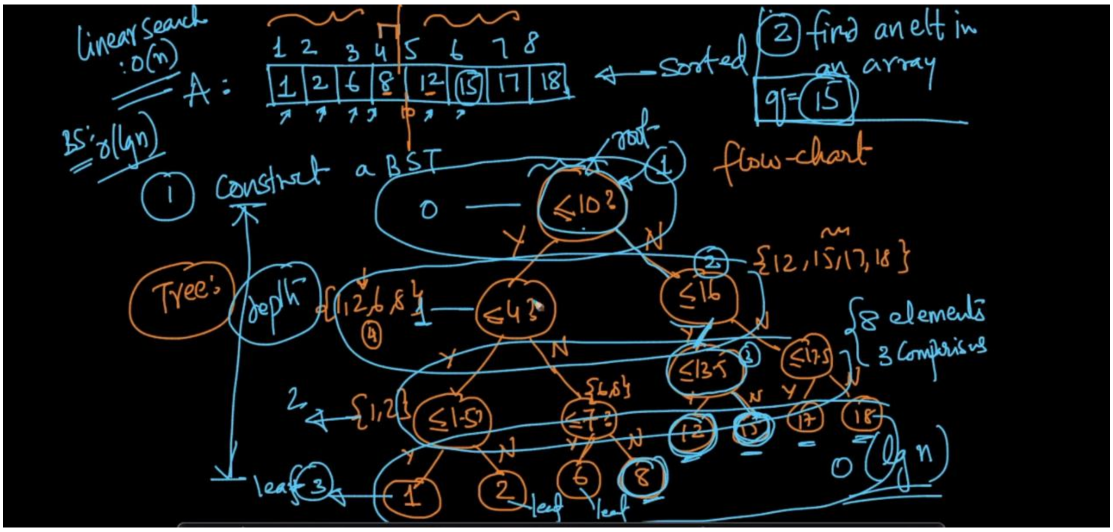
See above image we took weight and as distance increase weight decrease . so if wee took avg of - point = 15 and + = 1.75 .

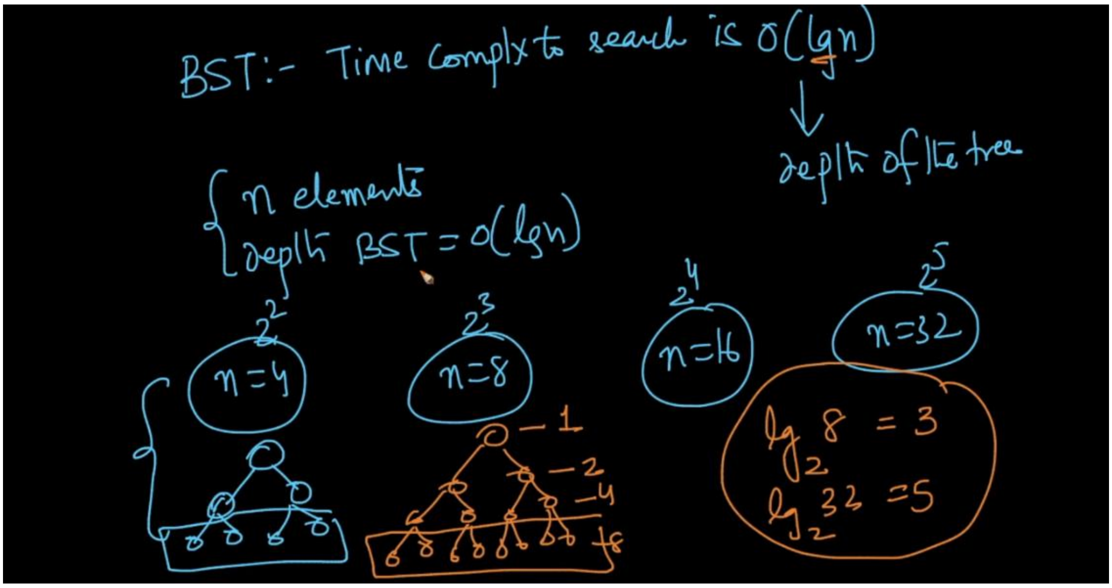
So we can say that new point belongs to - class .

Binary Search Tree :

Linear search we need O(n) comparison to search anything and in binary search we need only o(log n) .

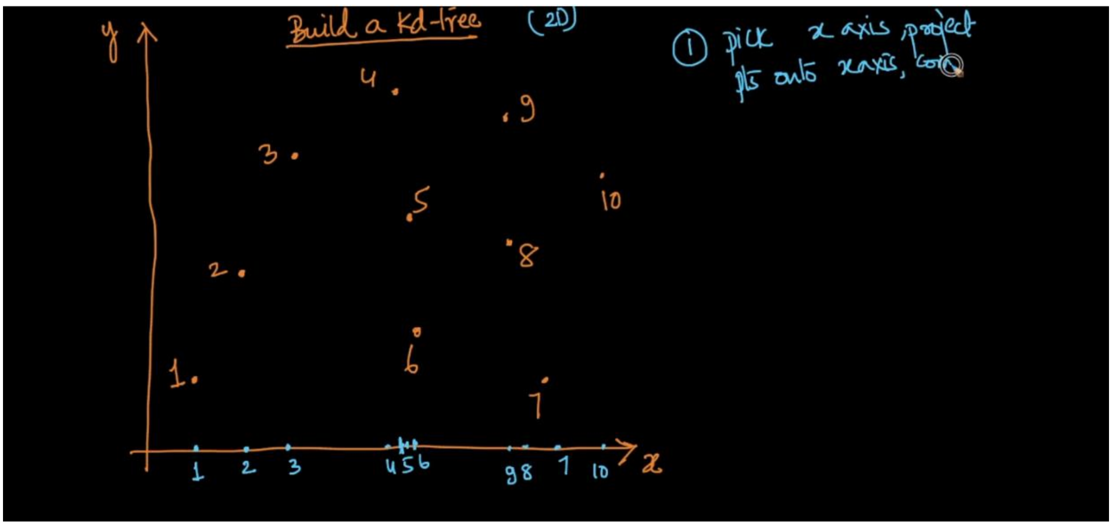
See below image in that array there are 8 elements and I want to search 15 in that array so using that BST in only 3 comparison I find 15 . which is too less time .

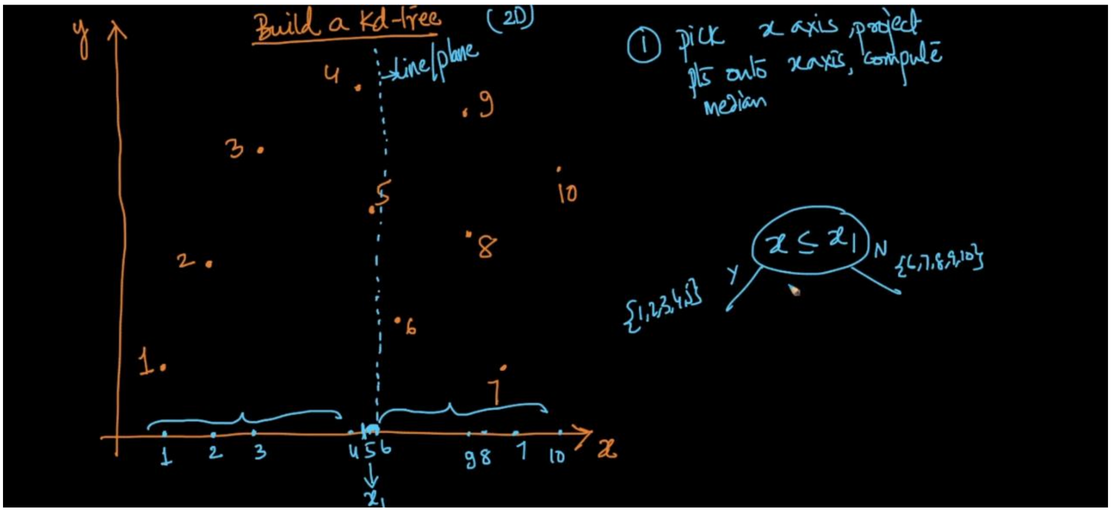




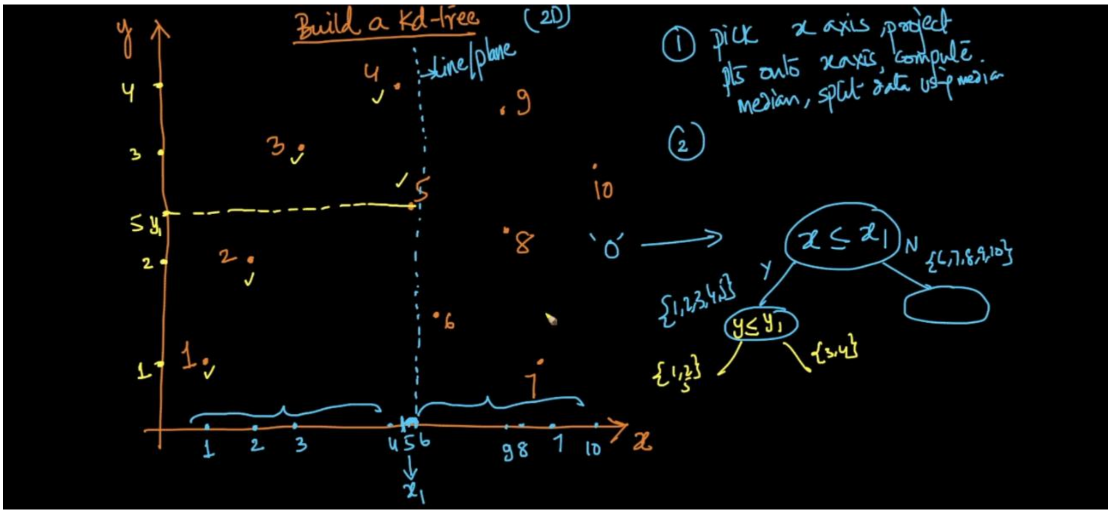
KD Tree :

Same like BST we split data into half half parts by taking median . This will help to reduce time complexity in KNN . we cant reduce space complexity in KNN .

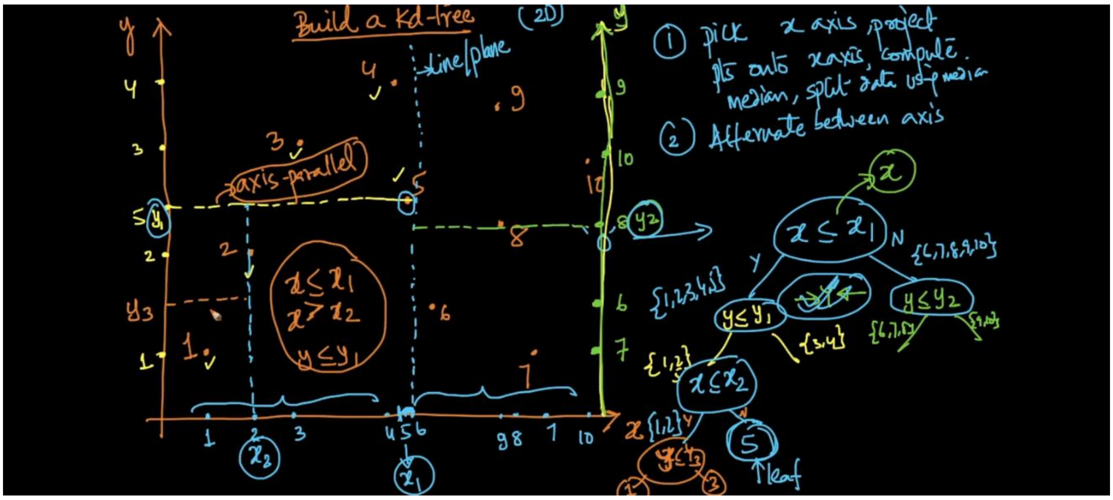


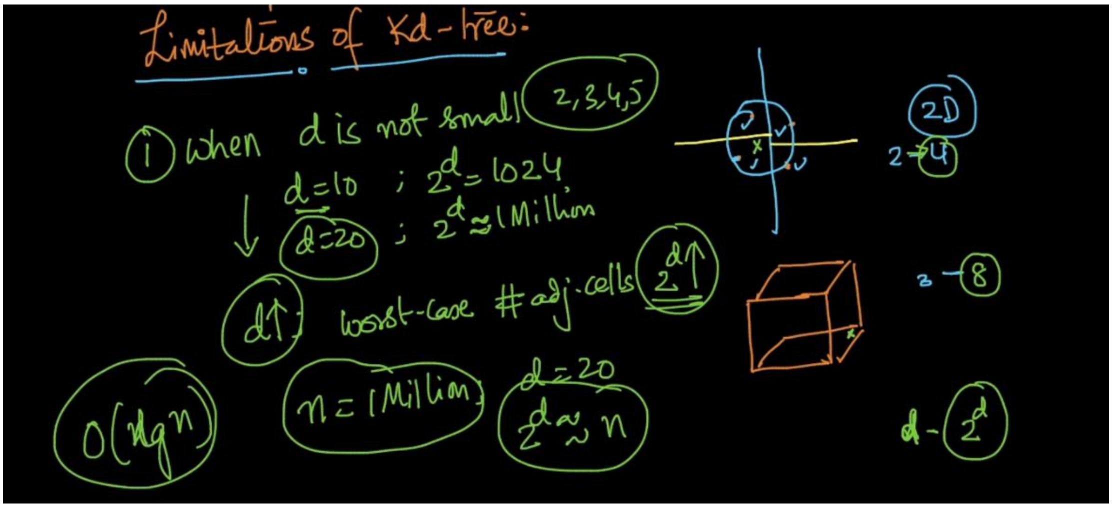


First we took X axis to split data then we use Y axis . see below image Y axis yellow color draw .



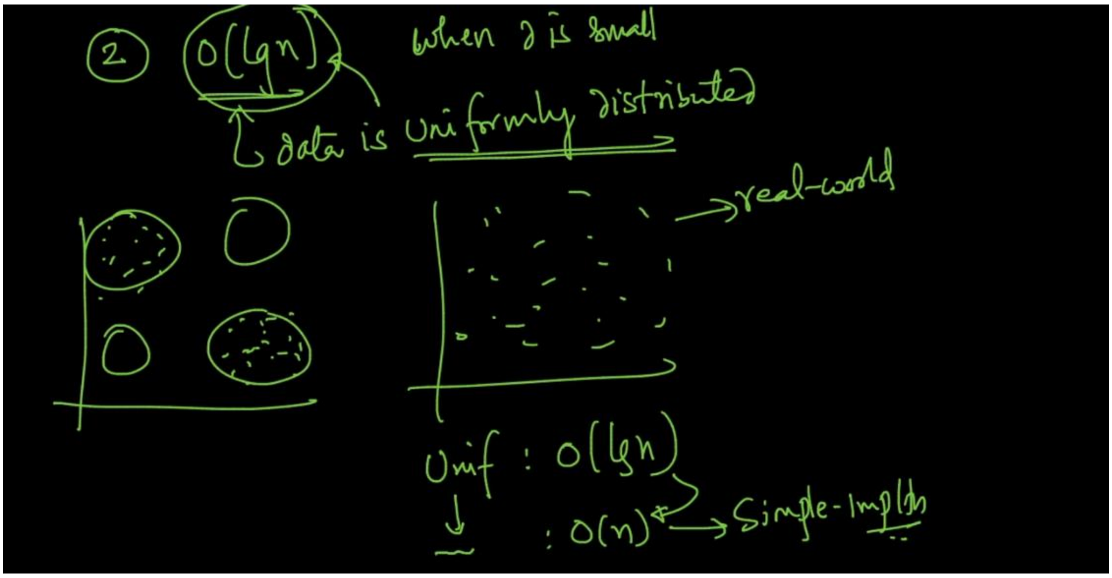
We do alternate between axis like first x then y so on ..





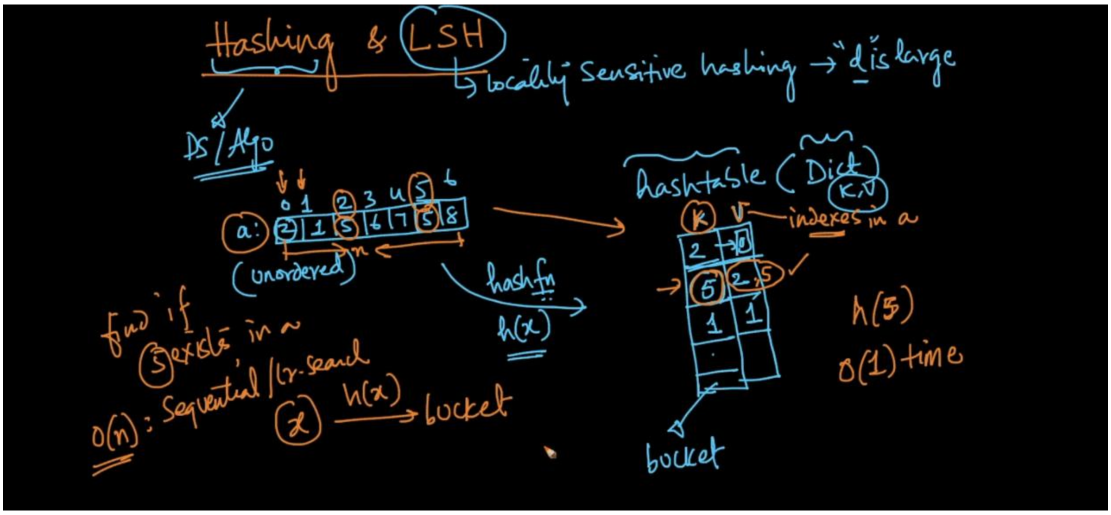
This is very big dis advantage of KD tree as dimension increase then time complexity drastically increase .

It is not designed specially for ml because we know in Ml we 100k dimension data so this will work more than worst . (Designed computer graphics because in that only 2D,3D data is there).



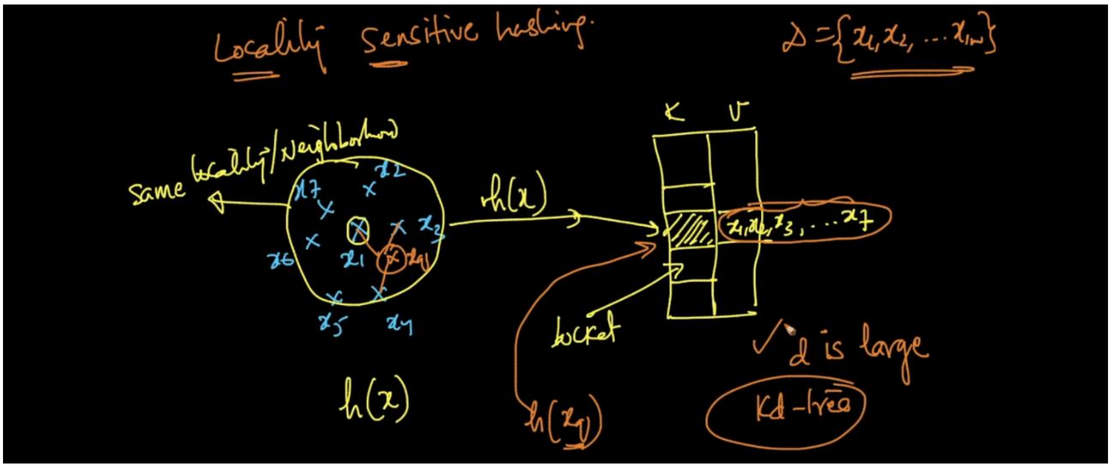
Hashing And LSH :

Its like Dict in python (key,value)



There is a hash table in that we store data . Time complexity retrieve is very very fast o(1) .

Because for each key hash function directly go that bucket and give data back .



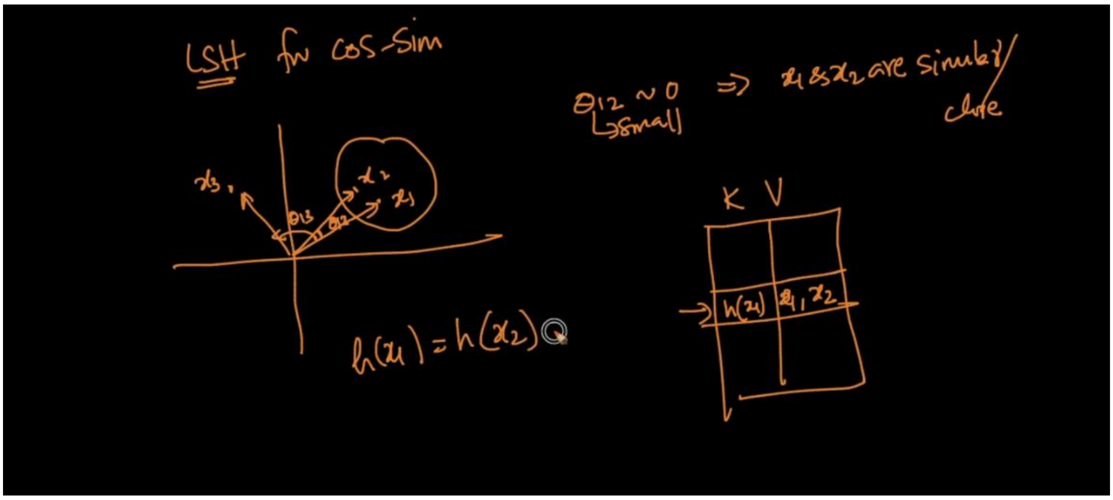
See above image LSH is very powerful method .

What we are doing here is our hash function will store all near by values to the same key .

So lets then point came x so it will directly search for that key it will reduce time complexity very well .

LSH for Cos Sim :

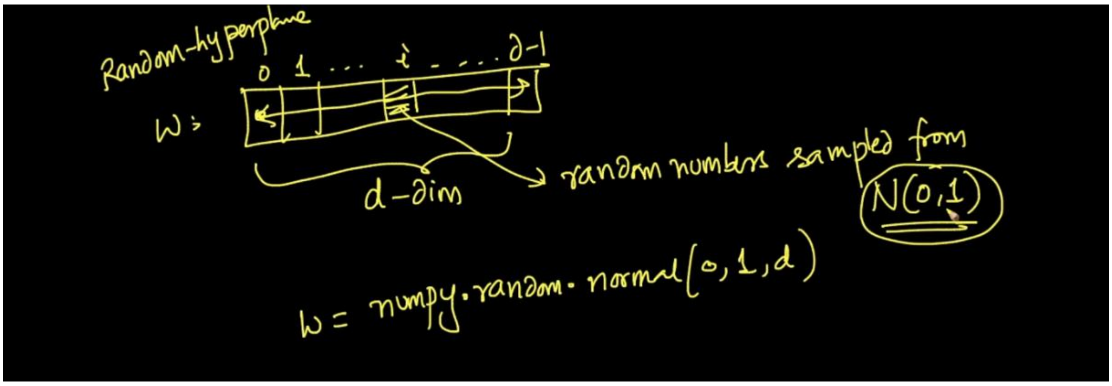
Cos sim we know that if angle between 2 point is less means they are similar .

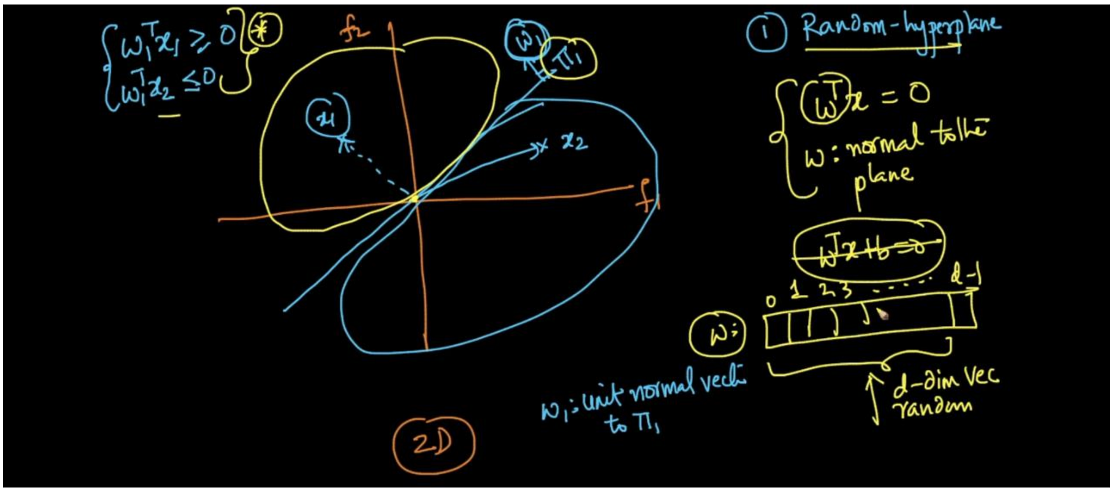


See above angle between x1 and x2 are less hence LSH will store x1 and x2 at the key .

In LSH first step we do is we break data angular means by random hyper plane .

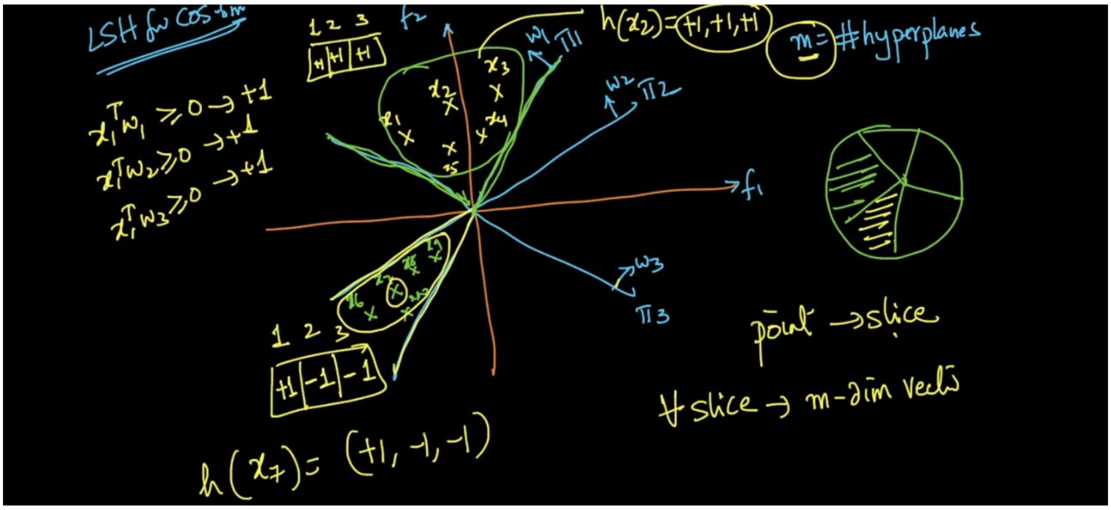
Generate random Hyper plane :





See w passing through origin is normal to plane . what w is ? its nothing but random d dimensional vector .

How To Find Hash Function :



See our data we split randomly by 3 planes . That planed split data like pizza slices .

So we create vector of n dimension means no of plane that much dimension here there are 3 planes so we use 3 dimension .

See above image we create vector of 3 values for bottom side data .

+1 means data lies in same direction of plane .
-1 means data is in opposite direction of plane .

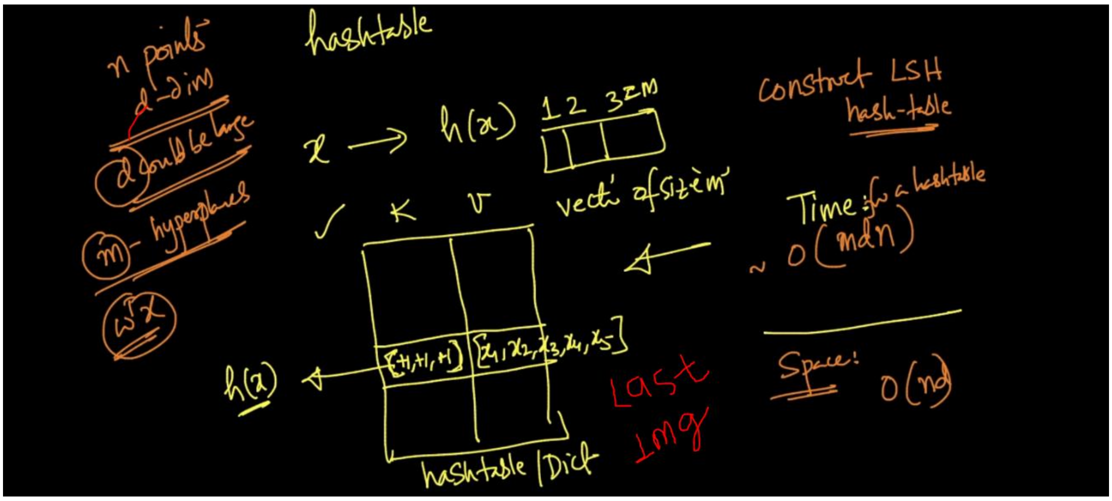
So for That bottom data our Hash function =

H(x) = [+1 -1 -1]

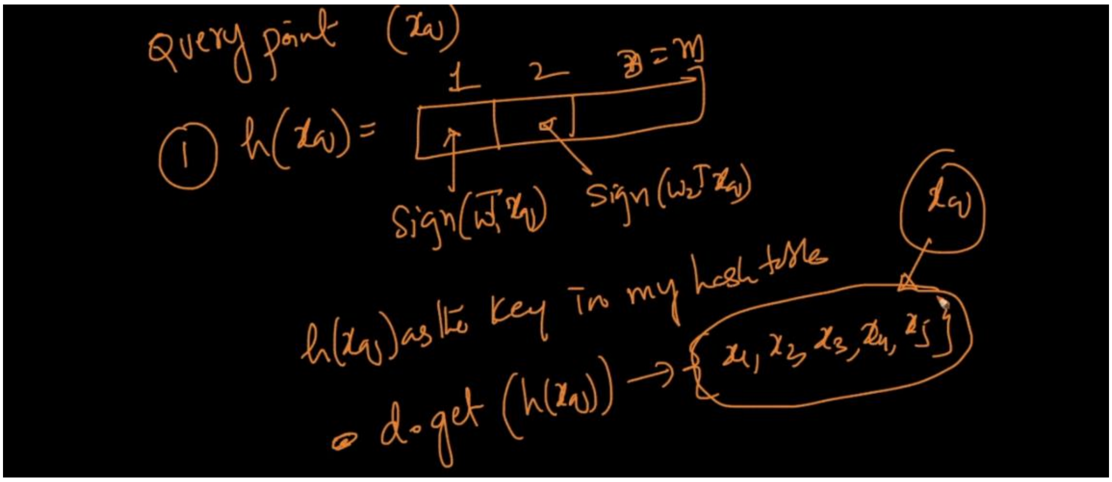
So given any point we first derive Hash function .

Lets say new point = x
Then we find H(x) First .

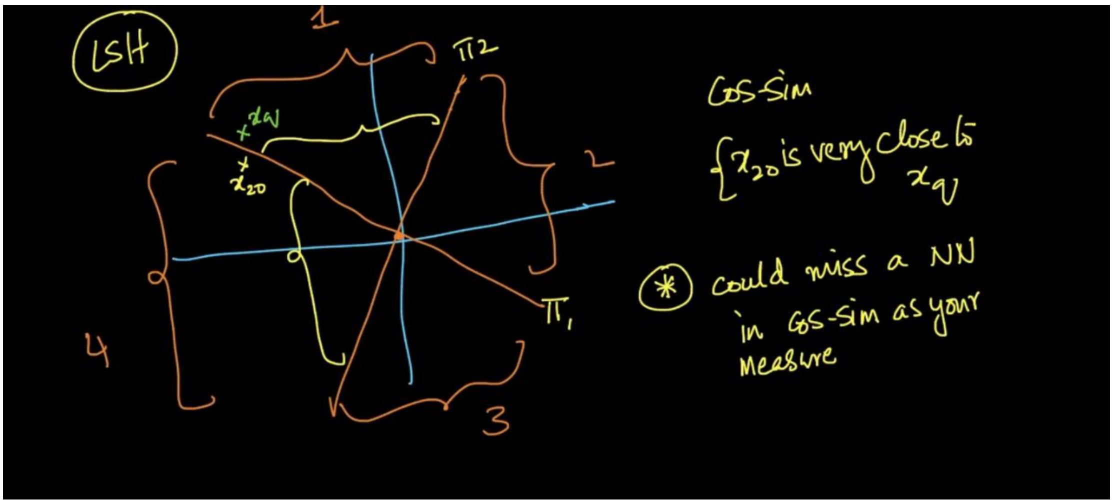
We use that h(x) as key in hash table .



So if any new point come lets say x(q) we first find h function then we get array of d dimension . then simply tell d.get(key) # python dictionary function it will simply go to that key and return all points in our example they are x1 …. x5 .



Now lets take complex problem :

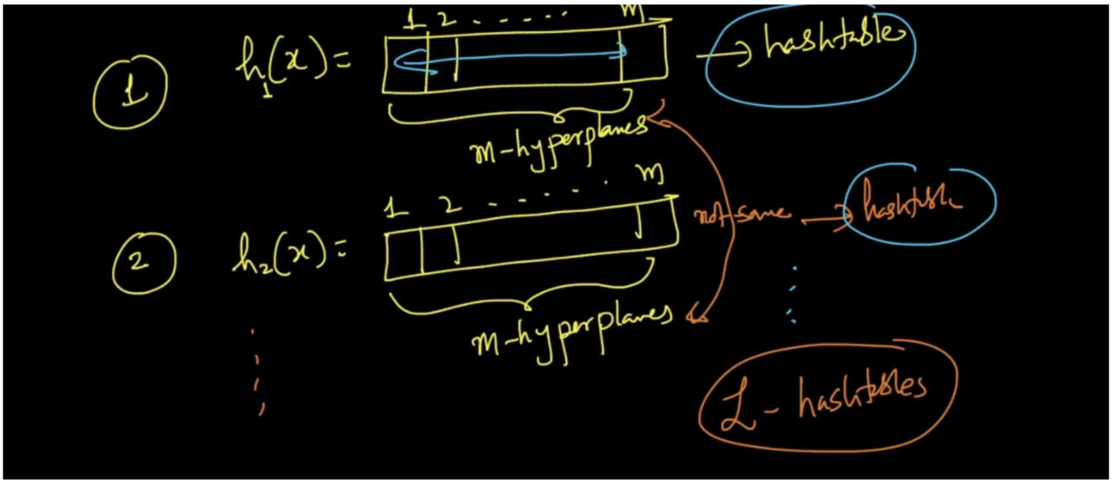


Here if we see x20 and x q are very near but still our algorithm consider them as diff because both are separated by plane so diff hash function for both .

So how to solve this issue ?

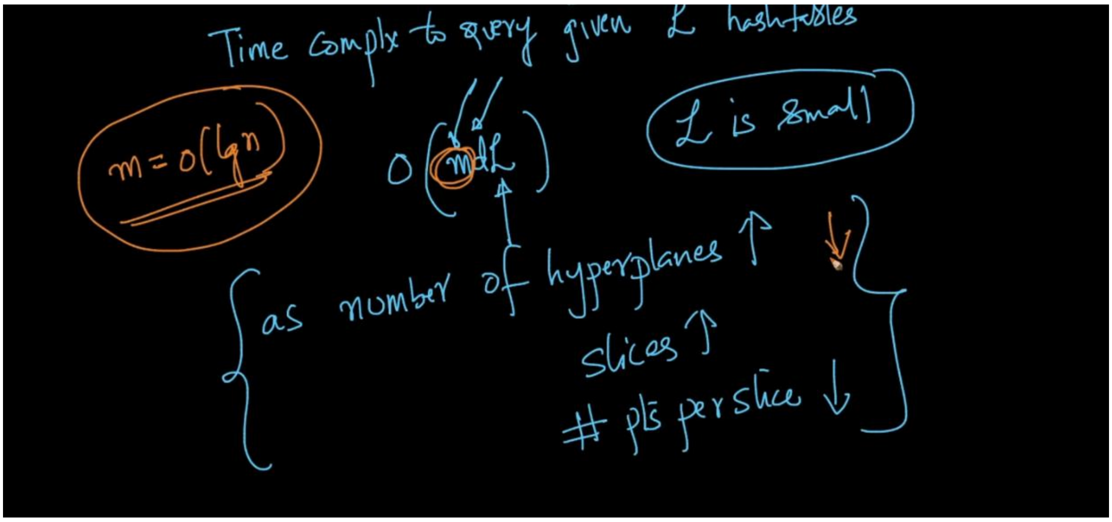
What we do here we first find h1 (hash table 1 for m hyper plane)

Then we repeat same process with different set of hyper plane each time for l times .

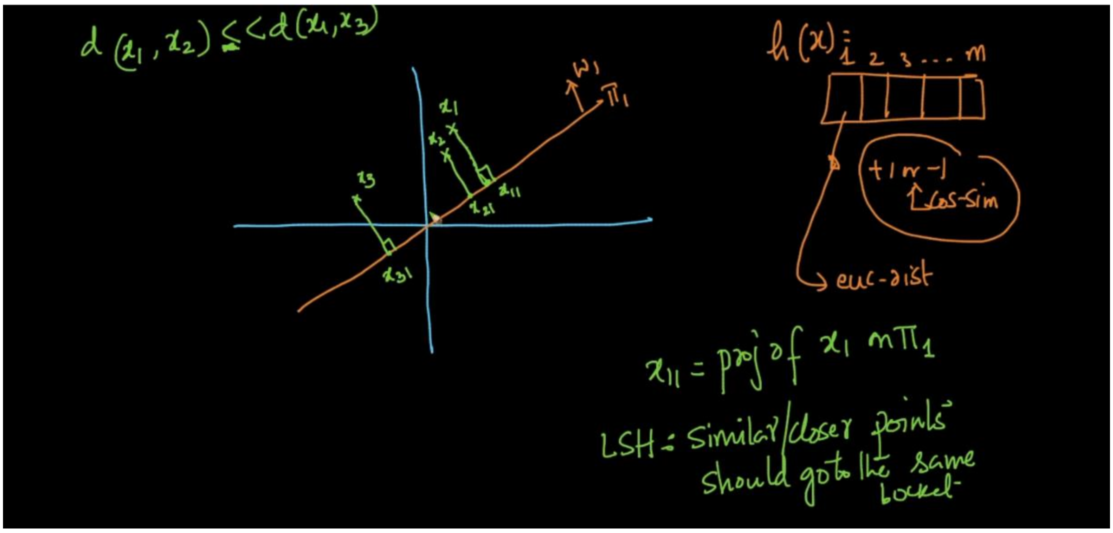


Finally we combine all points and take unions so we get all unique points in this way we can solve this issue .

As no of planes increases slices increase points per slice decrease this is not good . because we get vey less nearest points .



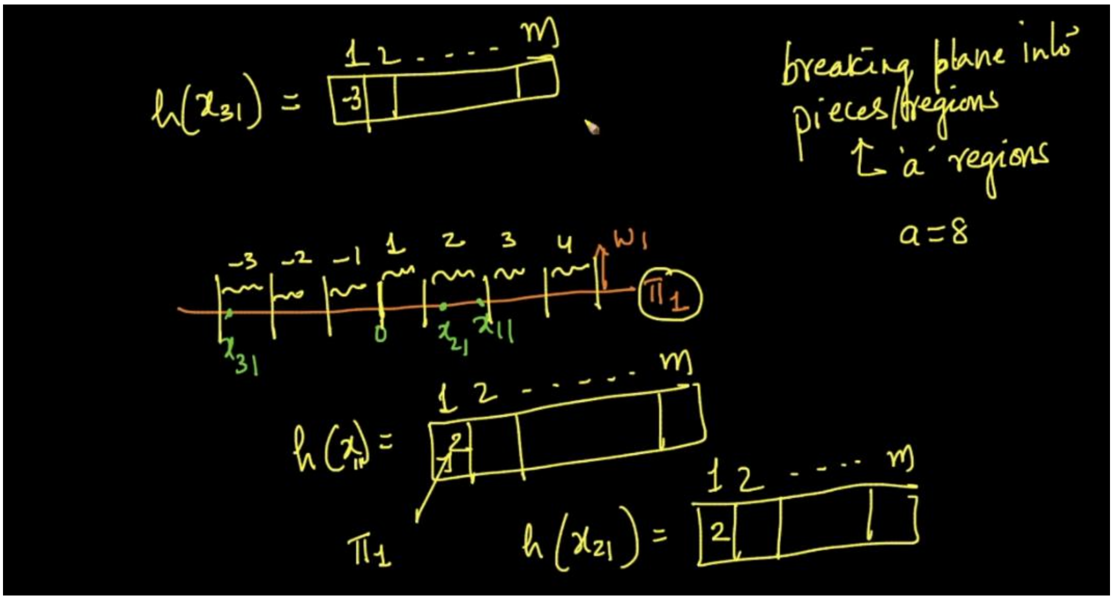
LSH for E distance   :

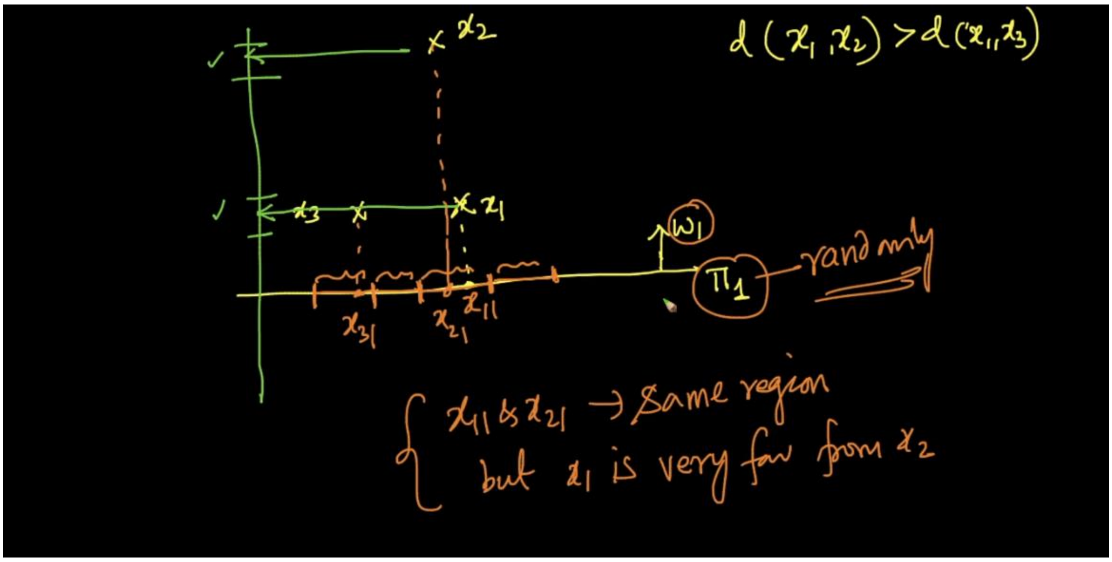


See x1 and x2 are too close than x1 and x3 .

What we do here we break plane in m regions .

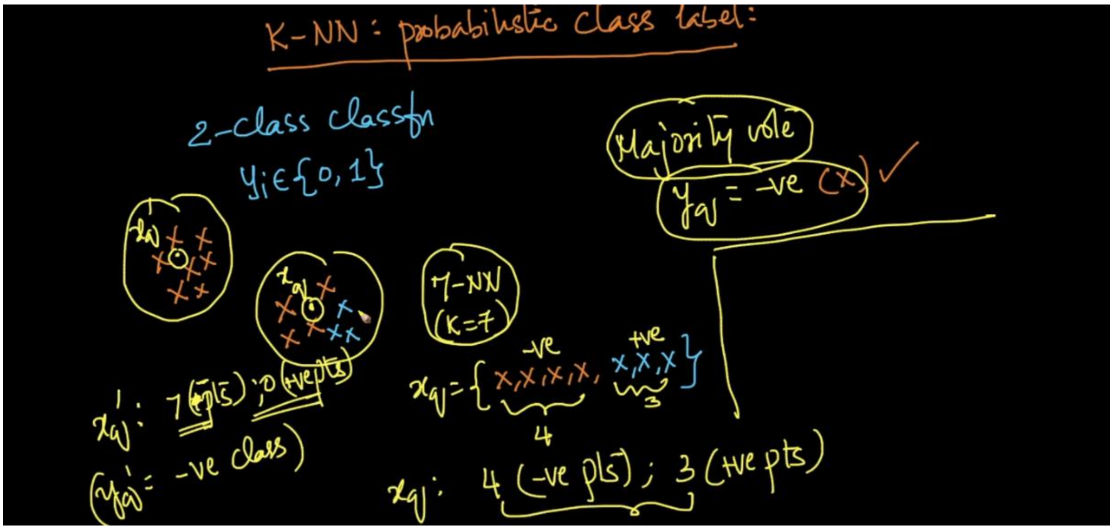
See x1 and x2 belongs to same region 2 as they are very close . Similarly we create vector as key for hash table . Here we get any value like -2,3,etc not only +1 and -1 .





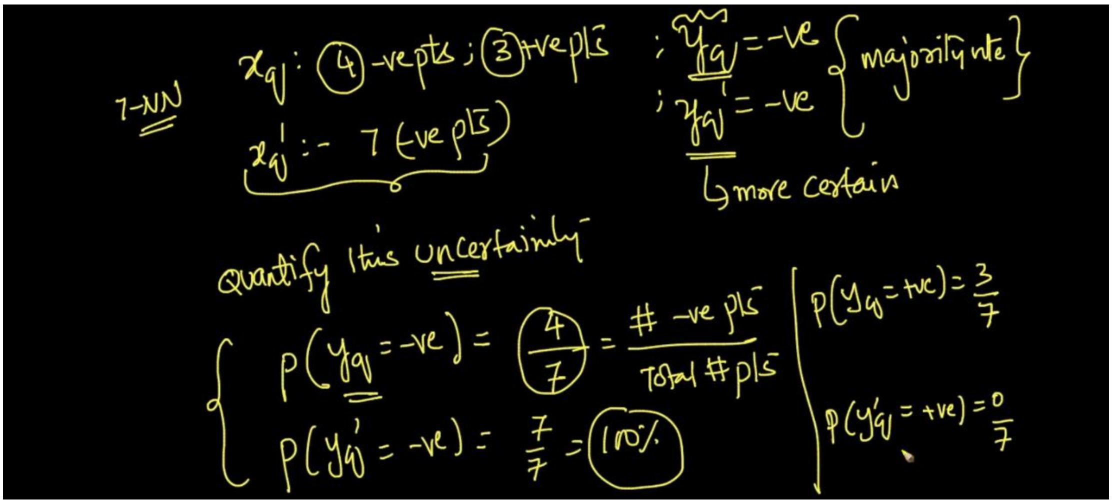
See above image x1 and x2 are too far but LSH says its very close in this way this will do mistakes .

LSH is not perfect its probabilistic random algorithm widely used in Computer vision .



In KNN we took Majority vote instead of that we can say probability and that will be very real .

See above 4 points are - and 3 are + then we can say
P(y =   -)   = 4/7



✓ probabilistic class label

$$x_q \rightarrow y_q \left\{ \begin{array}{l} p(y_q = +ve) \\ p(y_q = -ve) \end{array} \right\} \rightarrow \text{certain}$$

$\rightarrow 0.55$ ✓

$$x_q' \rightarrow p(y_q' = -ve) = 1 \checkmark$$