Classification Algorithm

How to deal with imbalanced data ? :

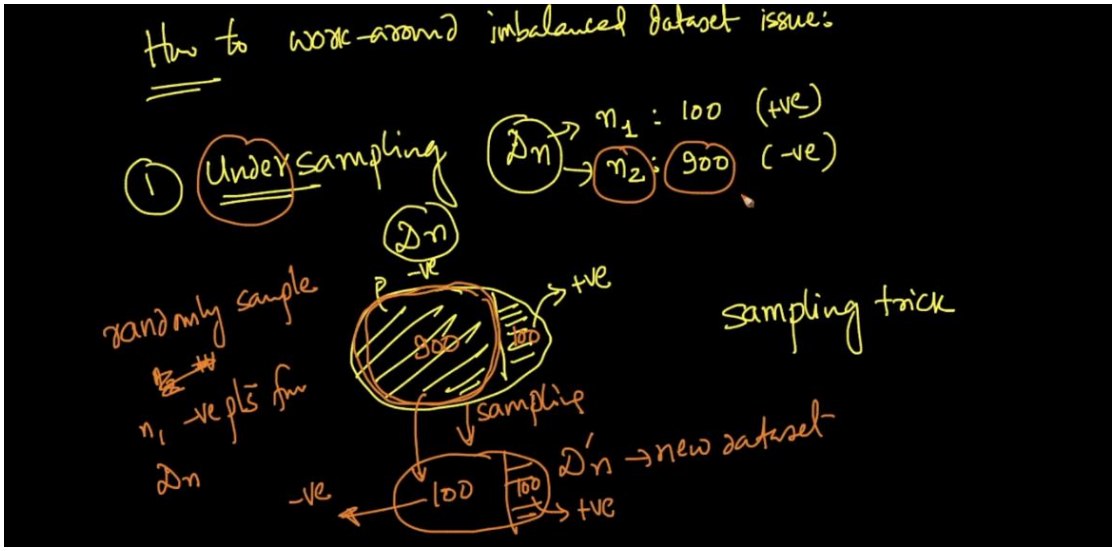
Imbalanced data means data is not proper for example lets say we have cancer data set 1000 values . In that 100 are + patient and 900 are - patient this is called not balanced .

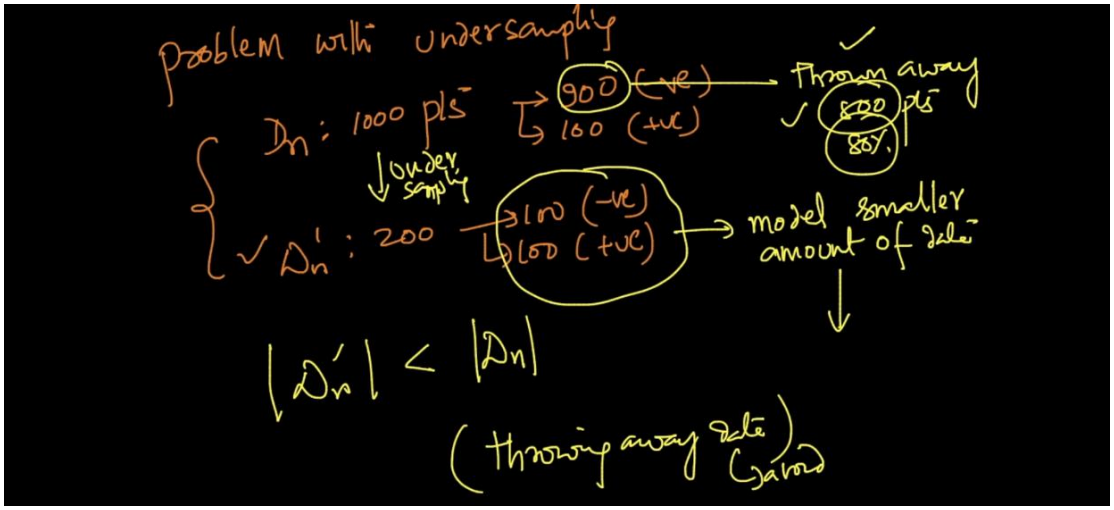
There are different methods used to handle Imbalanced data :

1. Under Sampling :

In this method what we do is we can make data balance by removing 800 (-) points .

So we get now 100 (+) and 100 (-) . But this method is very bad because we are missing too much data so our model will not work well.



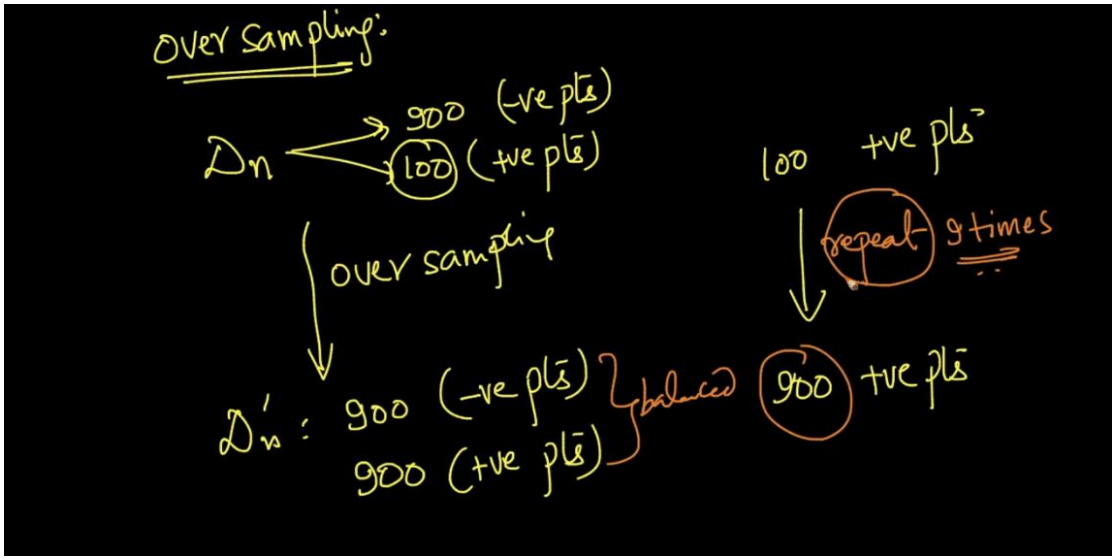


2. Over Sampling :

In this method we make data equal just by repeating same data for minority class .

In above example we repeat (+) points 8 times so we get + points 900 and - points 900 now our data is balanced .

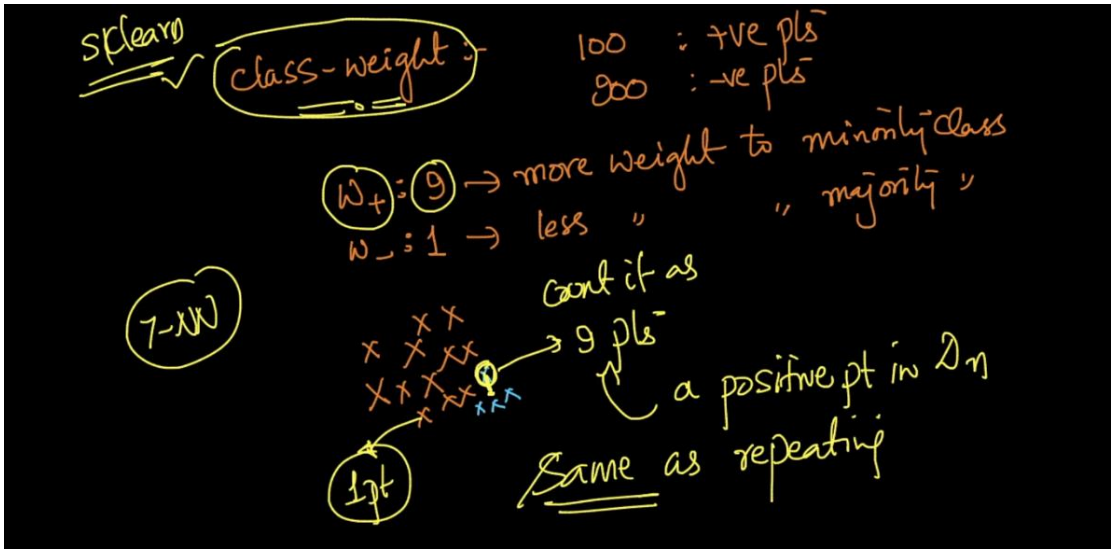
This method is too much better than previous because here we are using all data .



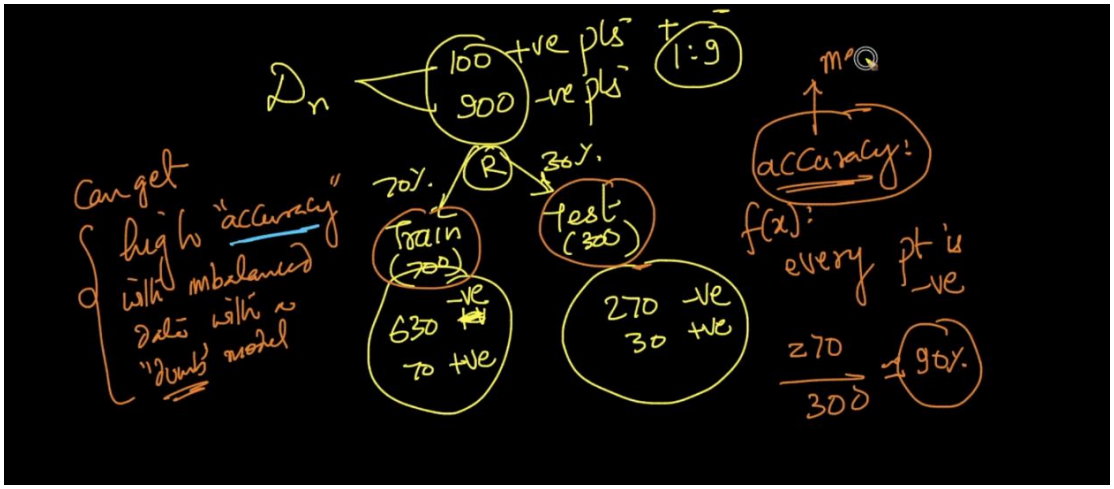
3. Class Weight :

In this method what we do is we will more weight to the minority class (+ class in our example) .

That means 1 + point = 9 (-) point in this way we can balance data its same like 2nd method .



Imbalanced data set occurs lots of time in real world problem .



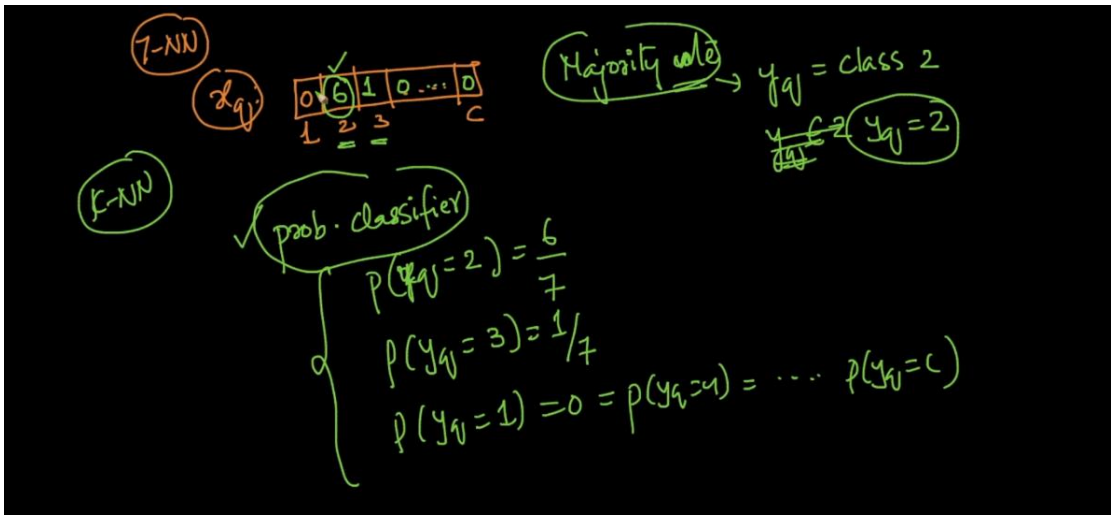
See above image we get 90% accuracy on test data its very wrong because of not balanced data .

M class Classification :

Algorithm like KNN can very simple do multiple classification but logistic regression algorithm can do easily .

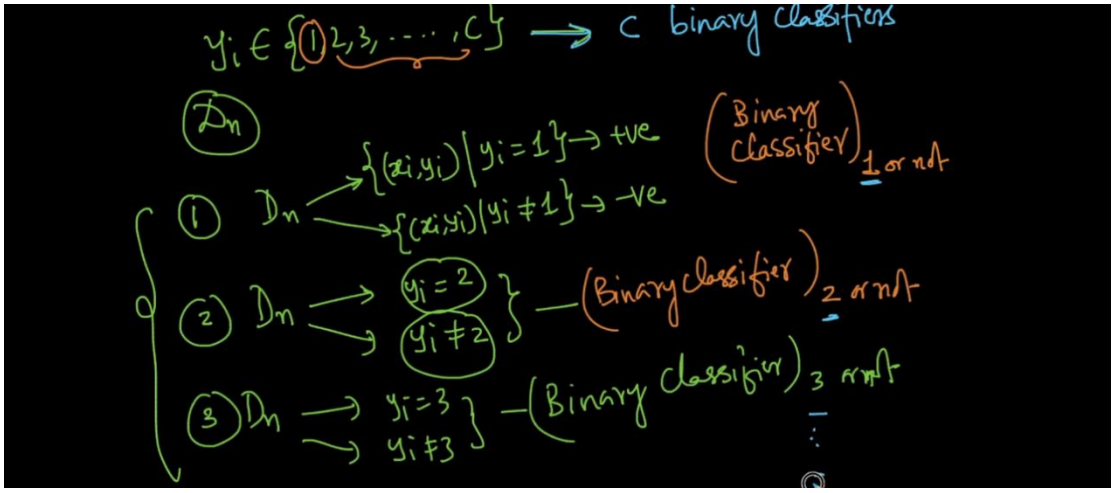
Lets say we have x class we are using KNN then we can easily predict lets say k = 7 .

Then we simply see say how much nearest value to class 1 ,2 . …. x . and we take majority vote and we classify .



Binary Classifier :

Not able to classify multiple class but we can solve that problem see below image .



We will make c classifier and will make classification . this will increase time and space complexity too much .

Similarity matrix means we create matrix for two variable and say how much they are similar .

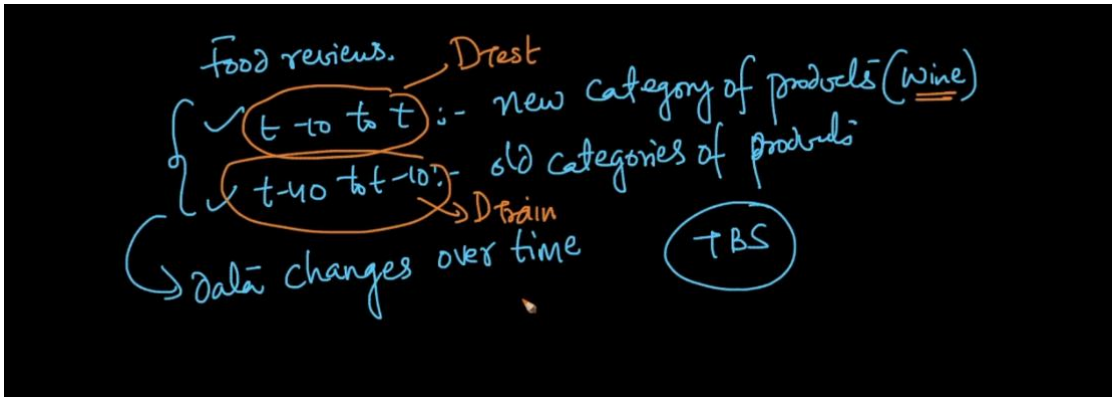
Ex. Drug1 and Drug2 if we go to the pharmacy and ask then how much they are similar we get values so that is nothing but similarity matrix .

Train and Test Data :

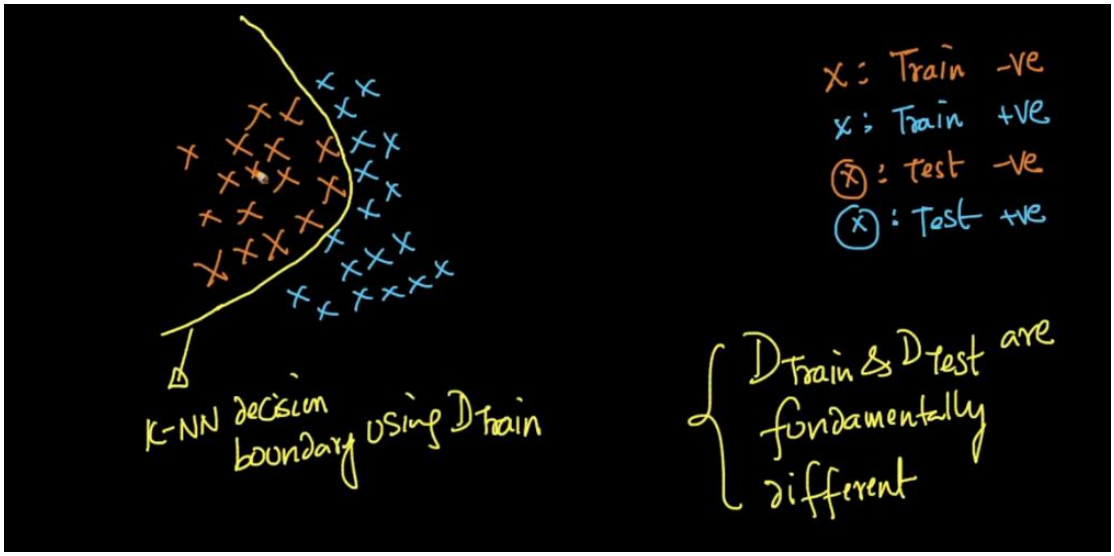
Lets take food review example as we have time based splitting so we have train data for today-60 days . and test data for last 10 days .

Now in last 10 day amazon add new category bear . so review for this product will be very different as

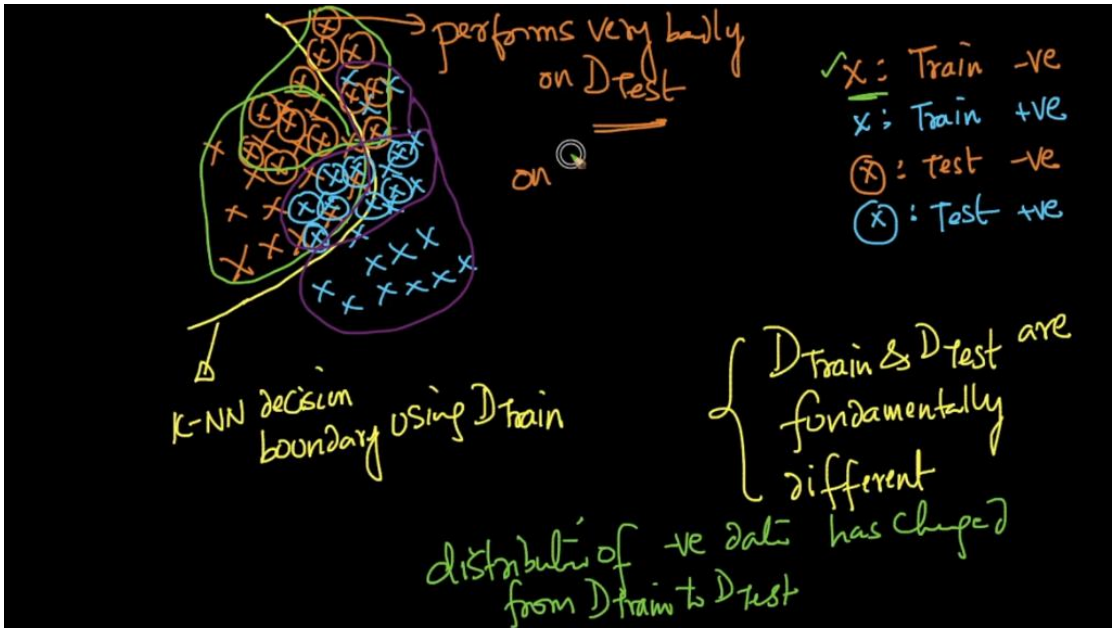
it is alcoholic food . but in train data we don't have this data ..

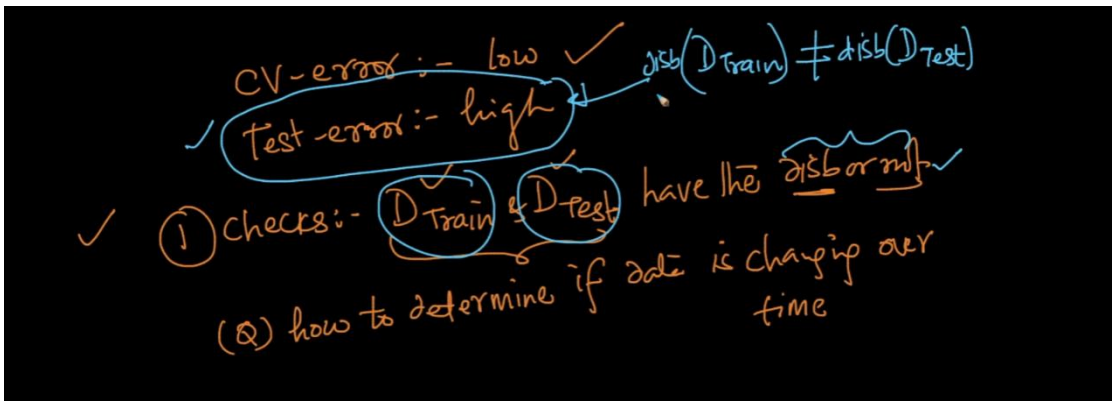


Because of this our whole distribution of data will change from train to test data .



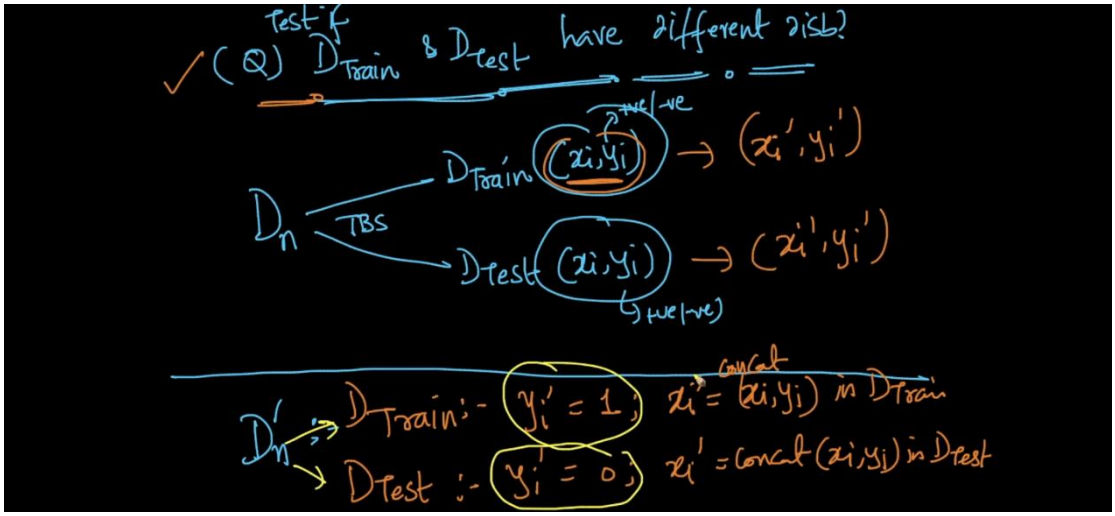
See below image now that yellow line work only well for train data but if we test same model on new data it will give very high error .



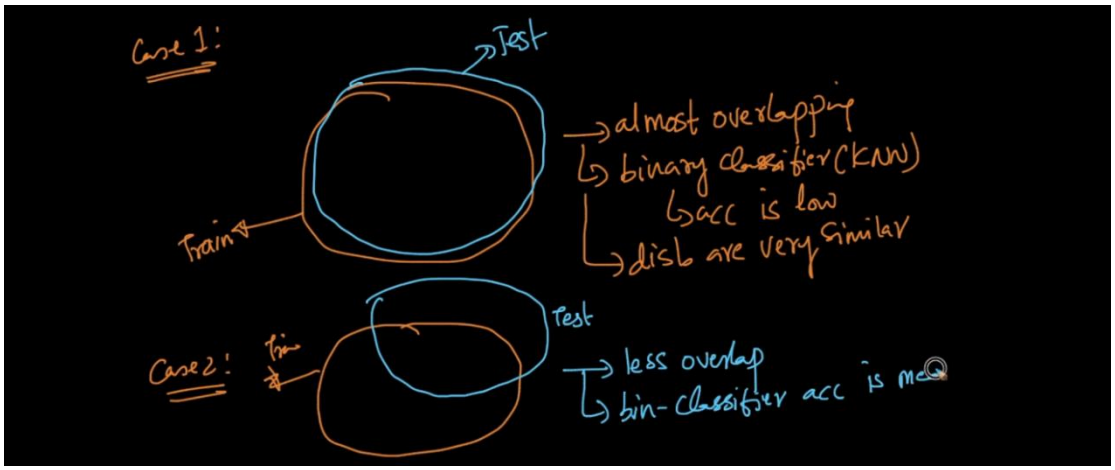


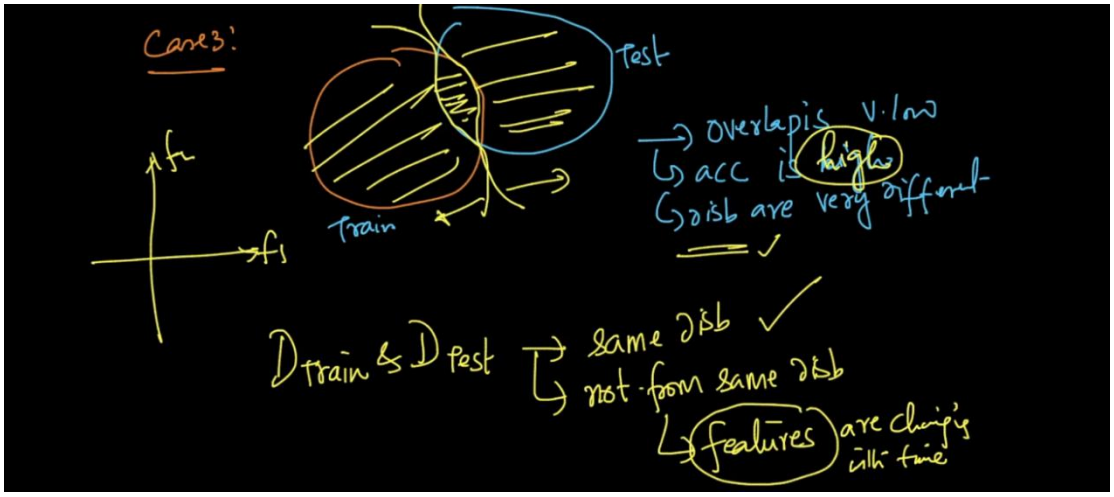
If distribution changes very speed then no model work very well so we need to check first whether both train and test follows same distribution or not .

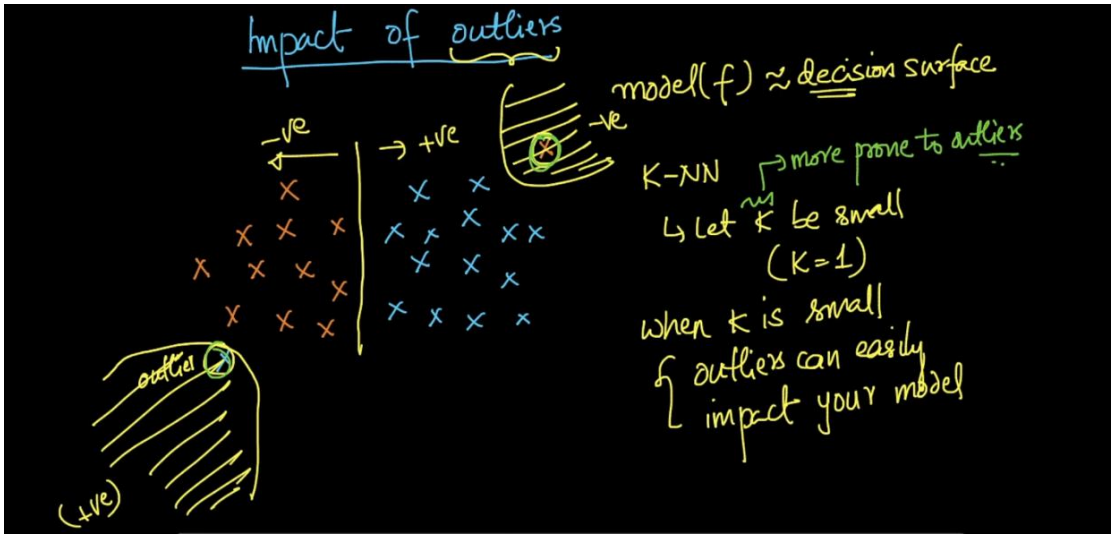
So we create new data set D' with all train data as label =1 and all test data label = 0 .



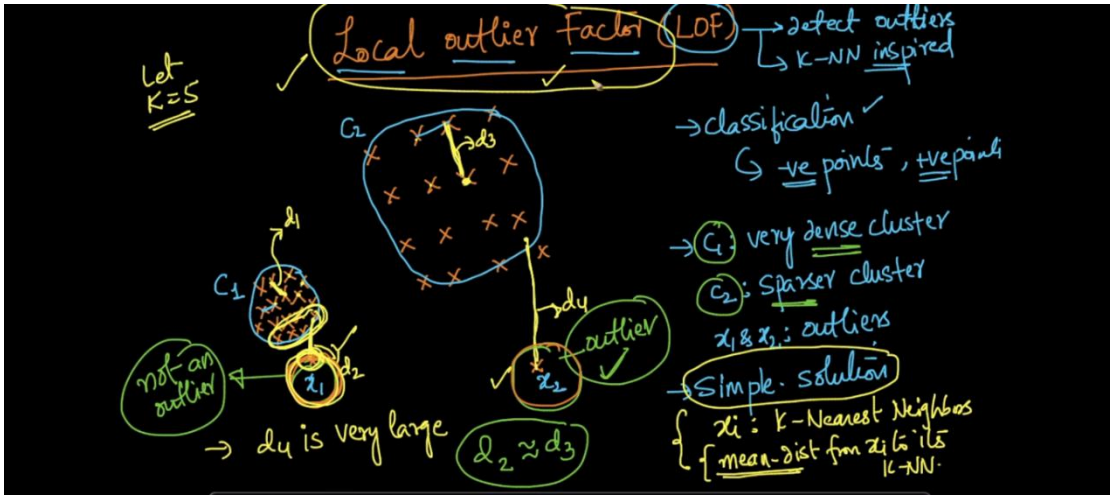
Then we can apply binary classification on new data to check whether they are from same dist or different .







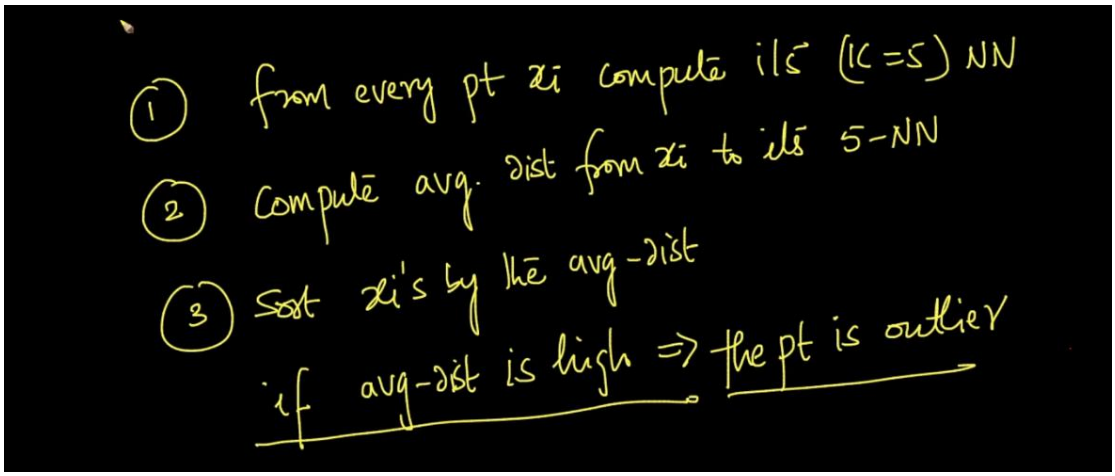
Always chose high K because if we chose k=1 and there is outlier then output may be affected .

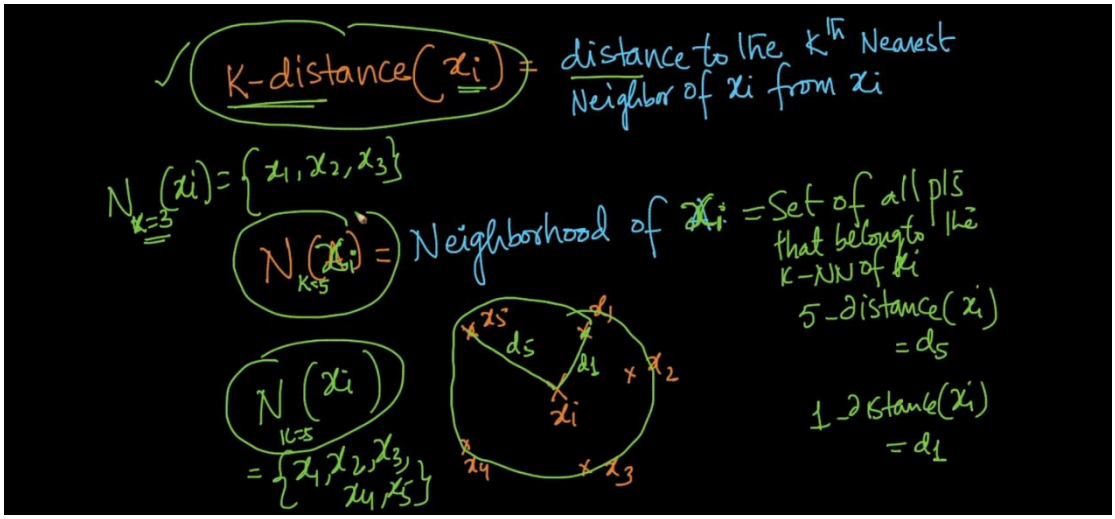


See above image LOF :

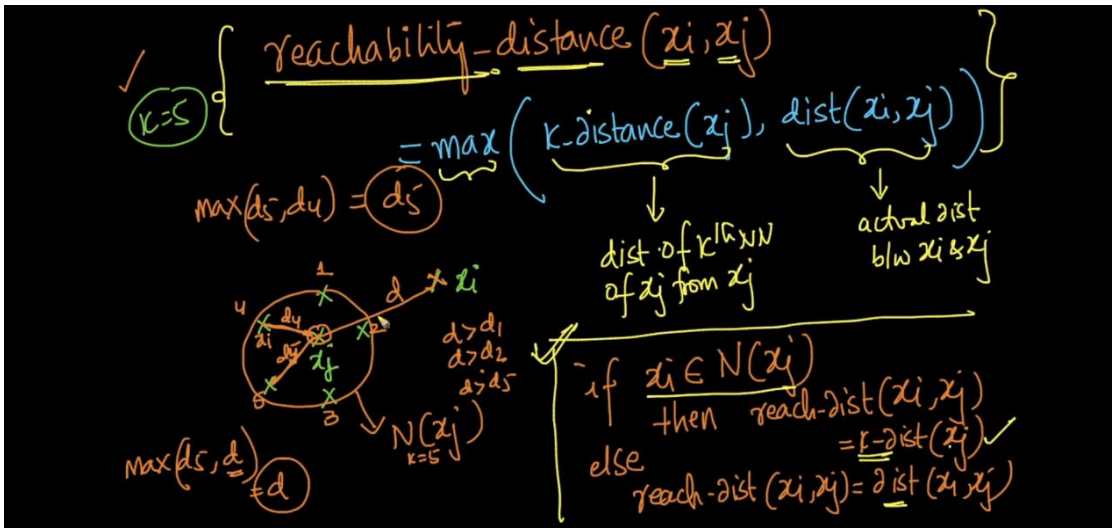
Lets say we have k = 5 so for each point we will find 5 NN and will take average of that point that is nothing but distance d1 . similarly we find d2 for cluster c2 and d3,d4 for 2 outliers . (Average distance for 5 neighbour)

Now we sort all d1 .. d4 . in this method problem is x1 not consider as out l. because d2 and d3 are almost same so if we say x1 is out l. then whole c2 will be out l. hence this method will not work well .

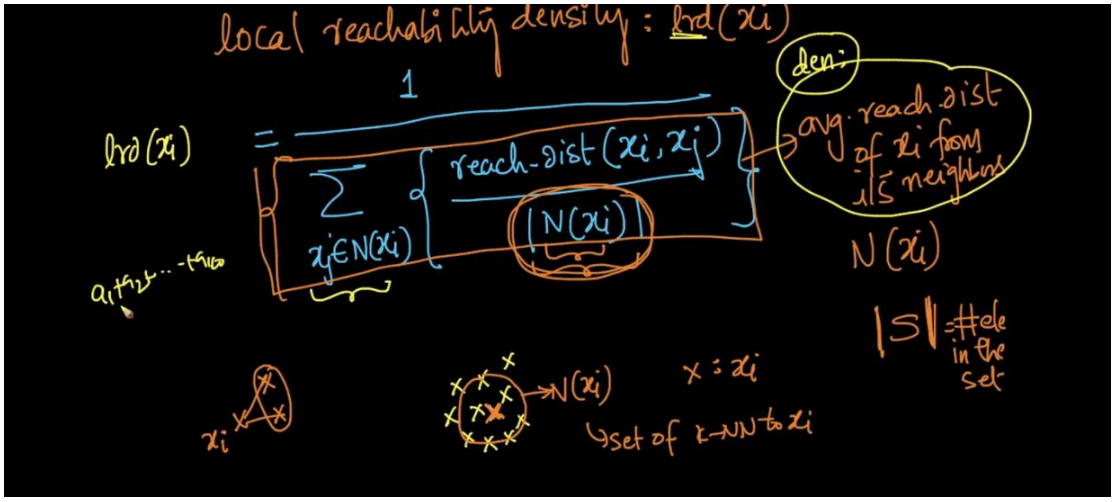


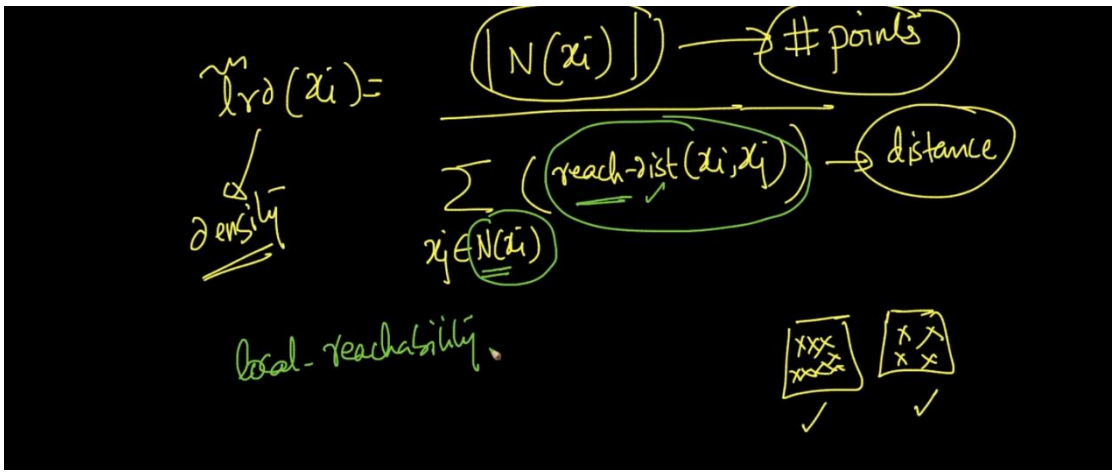


Reach ability Distance :

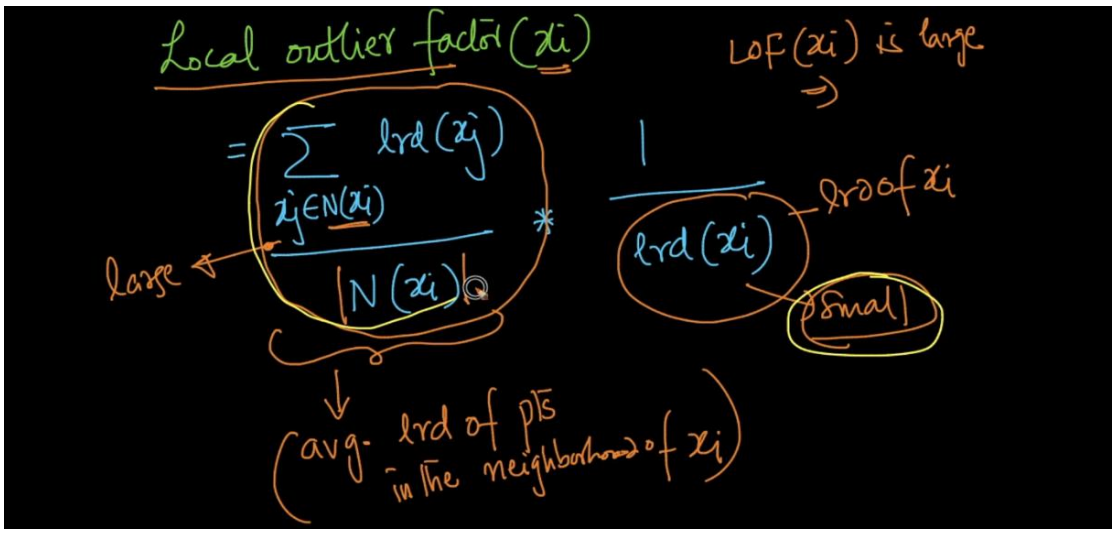


See above image . we will take max distance from formula .


local reachability density : lrd(xi)
lrd(xi) = 1 / (Σ xj∈N(xi) reach-dist(xi, xj) / |N(xi)|)
den:
avg. reach-dist of xi from its neighbors
N(xi)
|S| = #ele in the set
xi
N(xi)
x : xi
↳ set of k-NN to xi

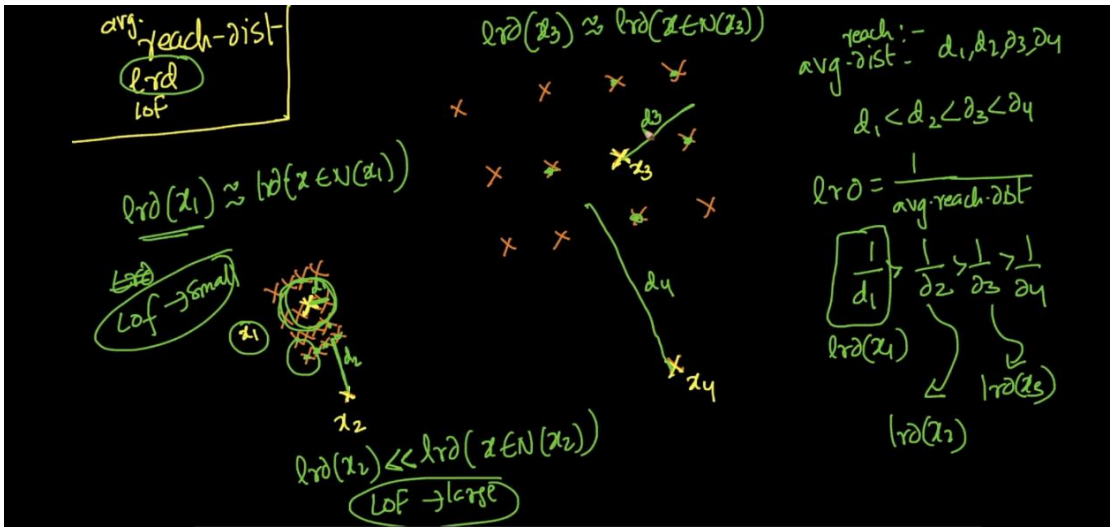

lrd(xi) = |N(xi)| / (Σ xj∈N(xi) reach-dist(xi, xj))
# points
distance
∝ density
local-reachability

LOF :


Local outlier factor (xi)
LOF(xi) is large
=>
= (Σ xj∈N(xi) lrd(xj) / |N(xi)|) * (1 / lrd(xi))
large
lrd of xi
small
(avg. lrd of pts in the neighborhood of xi)

See above image LOF formula . LOF will be high for any point if any one of the term is high in formula .

If both term are small then only LOF will be small .



See in this image we have 2 out l. now we say point x2 and x4 are out l .

Lets take 1 point from both cluster x1,x3 . If we find avg reach distance which is nothing but avg of all neighbour points . we get d1,d2,d3,,d4 .

Now LRD of d1 is high and density of NN around x1 is also high so LOF will be small .

Now LRD of x2 is much less than x1and and density of its NN is high see in fig . So high + low = Large LOF .

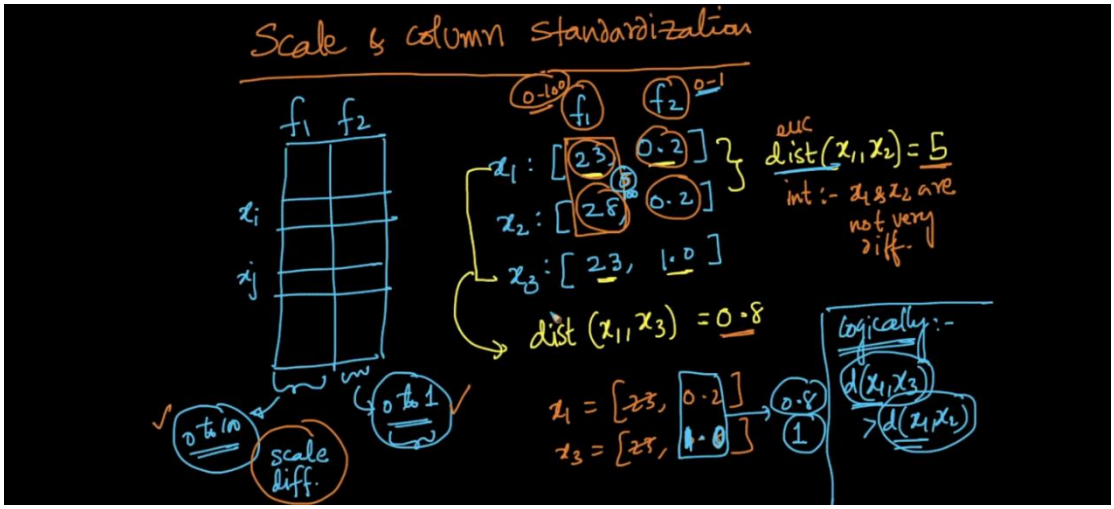
Same LRD of x3 is small and density of its NN is small (Not too dense) so LOF will be Low .

LRD of x4 is much less as compared to its NN density . and density of its NN is high so LOF is high .

So we can consider points as out l . whose LOF is very high .

But there is no scientific logic how much high LOF Is bad we need to take that decision as per our data ..

Column Standardization :



Now see above image lets say we have feature 1 and feature 2 f1 and f2 and both are from different scale means one lies from 1 to 100 and f2 lies from 0 to 1 .

If we measure E distance as per that we found distance between x1 and x2 is too much greater than x1 and x3 .

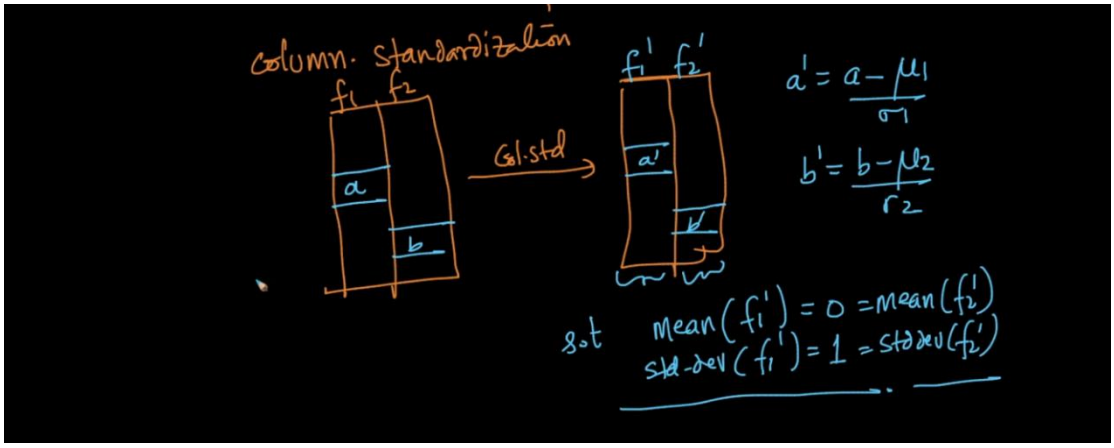
But logically this is wrong this issue because of different scale and due to this we get wrong NN in KNN .

If we see distance between x1 and x2 = 5 but range of f1 and f2 lies from 1 to 100 so as compare to that is very small .

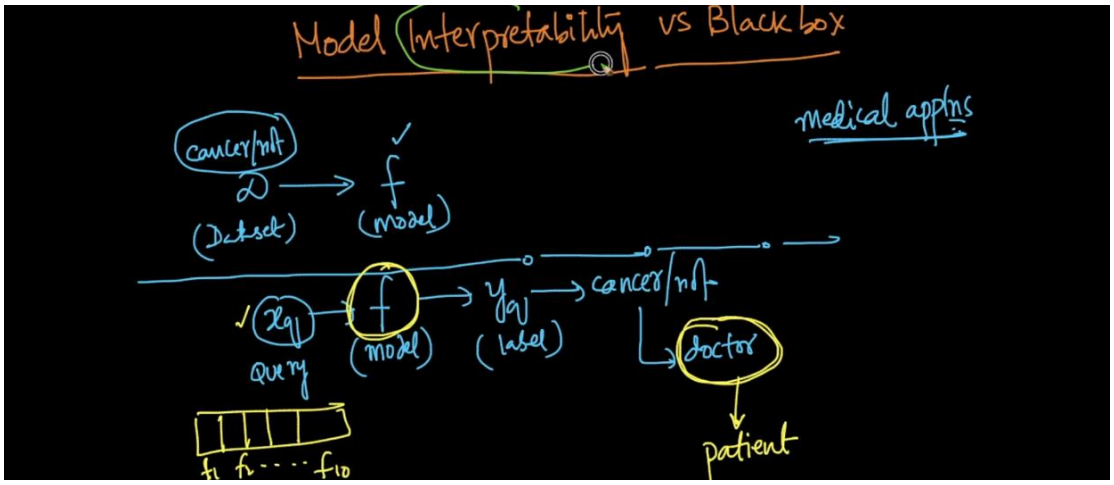
Other side distance between x1 and x3 is 0.8 and max distance is 1 so it is too high ..

So to avoid this type of problems we need standardization of col . See below image .

So each time when we are applying KNN need to do 1st is standardization of data .



Model Interpret and Black Box :



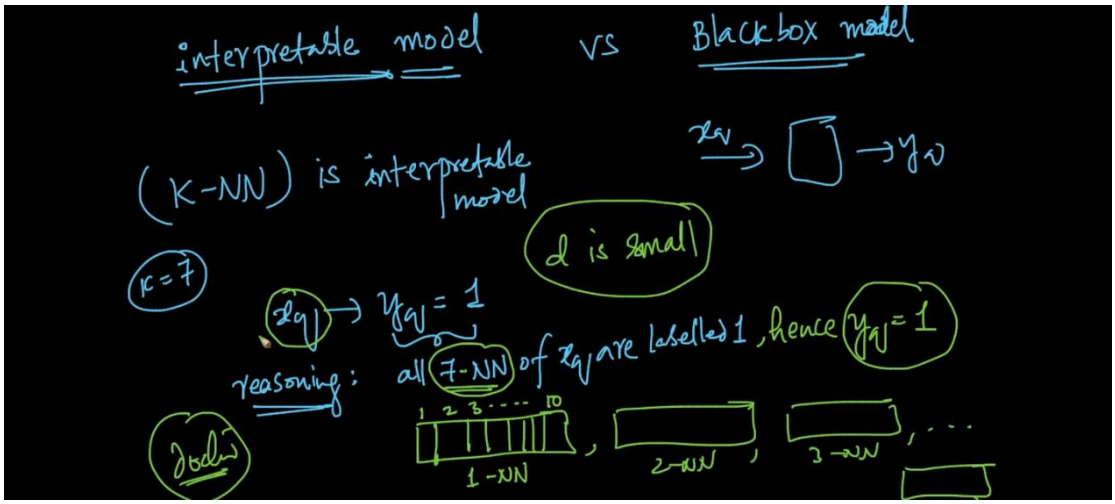
As we know that we have a data we train it and create a model ,when new data come we just use trained model to predict output .

Lets say we trained model on cancer data set now new patient come and our model predict he has a cancer . but patient and doctor both need reason why he has a cancer on what basis our model is telling that because doctor don’t know any maths behind that we need a reason .

That is nothing but interpret model not black box .

Means it will give perfect reason and will tell I found f5 (test 5) result is very high and f8 is very low and because of this I say this patient has a cancer .

Then doctor can easily check test 5 and 8 and he will understand whether our model predict correct or wrong .

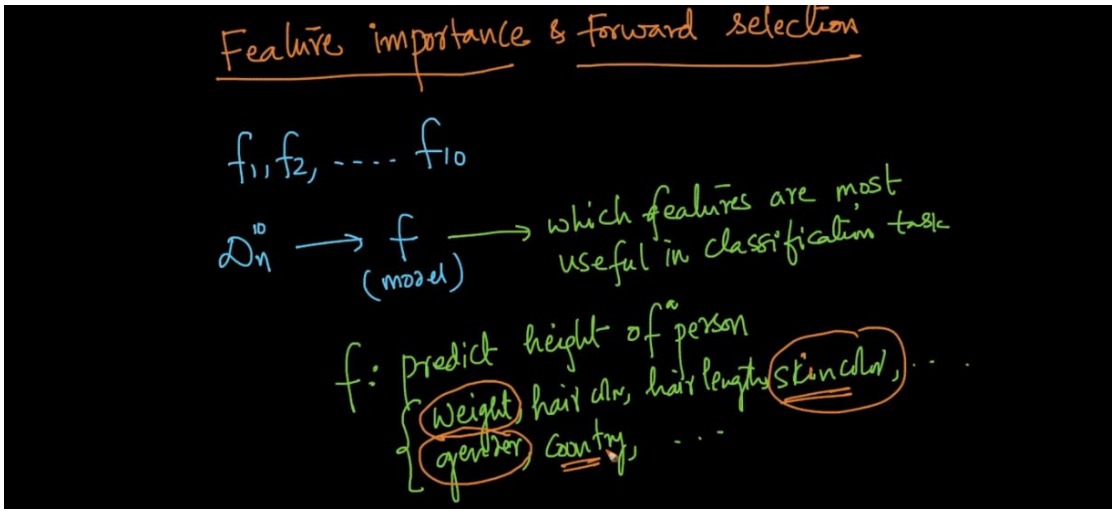


KNN is interpret model if d (dimension is small in our case here d = 10) so we can say it is interpret model ..

What KNN will do in this case is it will find lets say we decide k = 5 , so it will find 5 NN who has same

condition like new patient and as per that KNN will say Cancer or Not.

Feature Importance and Selection :



In real time we cant say we have 10 features only we get 1000 sometimes more than that so using all of the feature will decrease model performance .

So now what we can do is we will try to remove non importance feature lets say from 1000 to 100 .

One way is PCA TSNE but they didn't care about classification one very powerful method is Features forward selection .

What we do here is we take f1 and trained our whole model and will check accuracy on test lets say we get accuracy a1 .

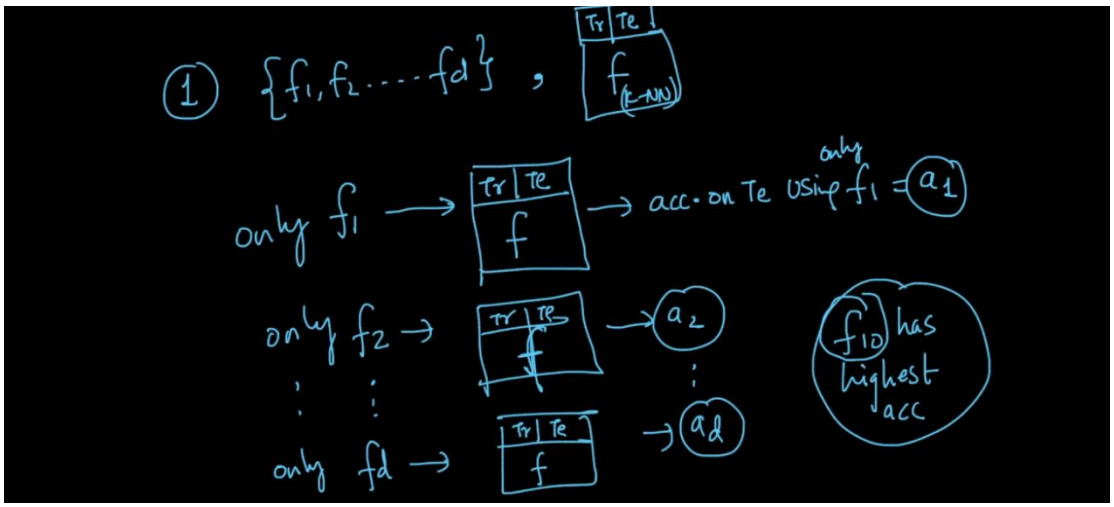
Similarly we repeat up to d features (1000) now we have 1000 accuracy we will check which one is highest then we select that feature . lets say f10 has very high accuracy so we select f10 .

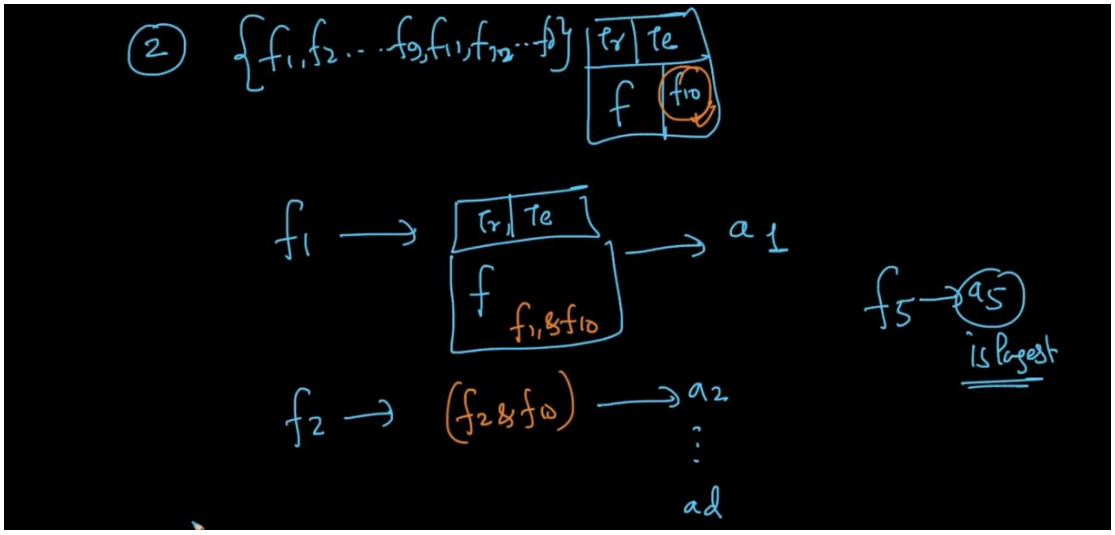
Then we take f10 + f1,f10+f2 ……. f10+ f d. then we chose highest accuracy feature lets we get

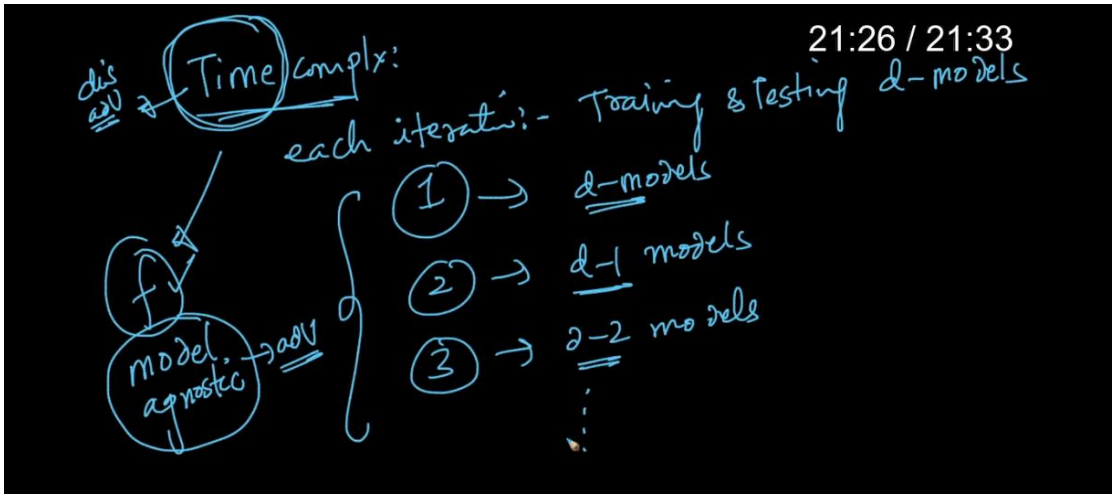
F10 and F5 .. similarly we repeat this process until we get 100 feature in this way we find best 100 features .

This method works on any any model .

Backward selection is exactly opposite to it we will remove 1,1 feature and will check how much effect on accuracy but forward is best .







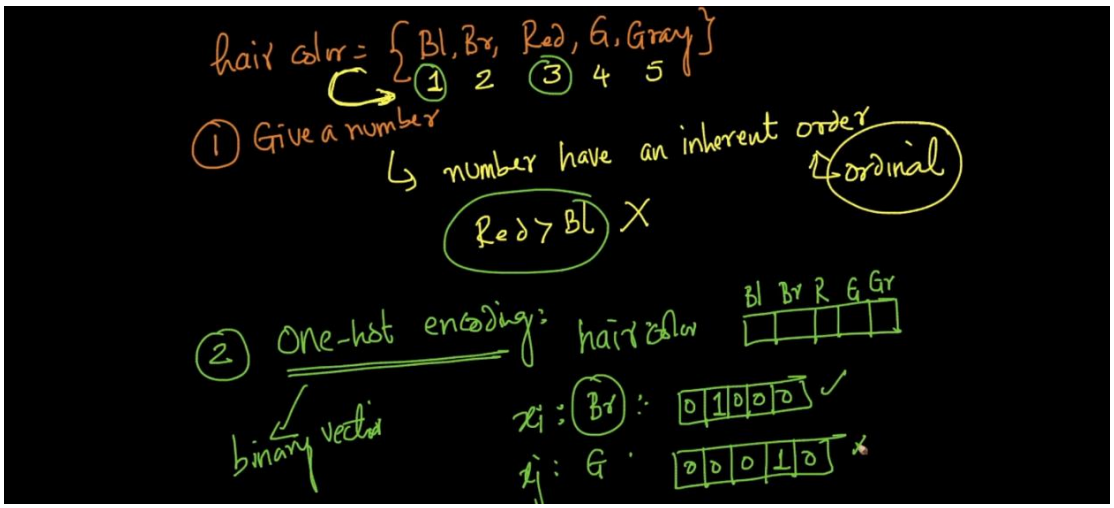
Only the problem is it will increase time complexity very very high but still its good because its one time effort after that we get best features .

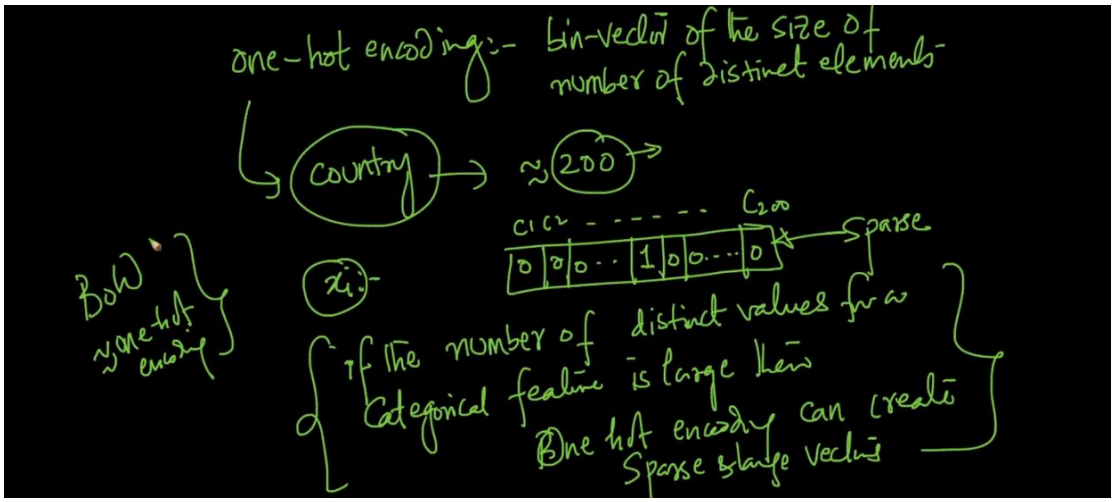
How To deal with Categorical Data :

Lets say we category data for hair,color,country etc . and we want to predict height .

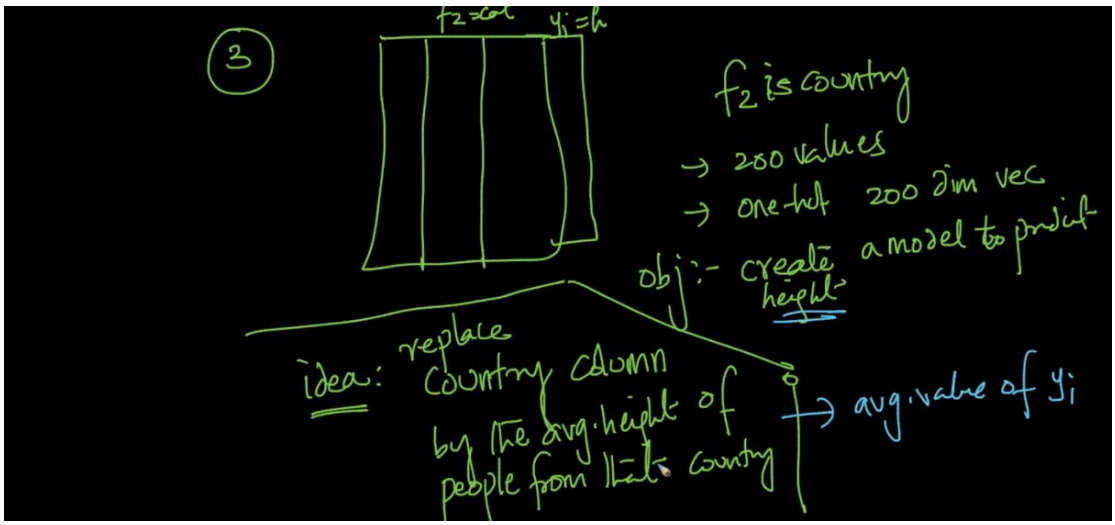
One way is we can convert feature into one hot . but the problem is lets say I have country feature and almost we know 200 unique name for country .

So one hot vector will be size of 200 and in that only one values will be 1 and 199 will be 0 this is like BOW problem sparse .



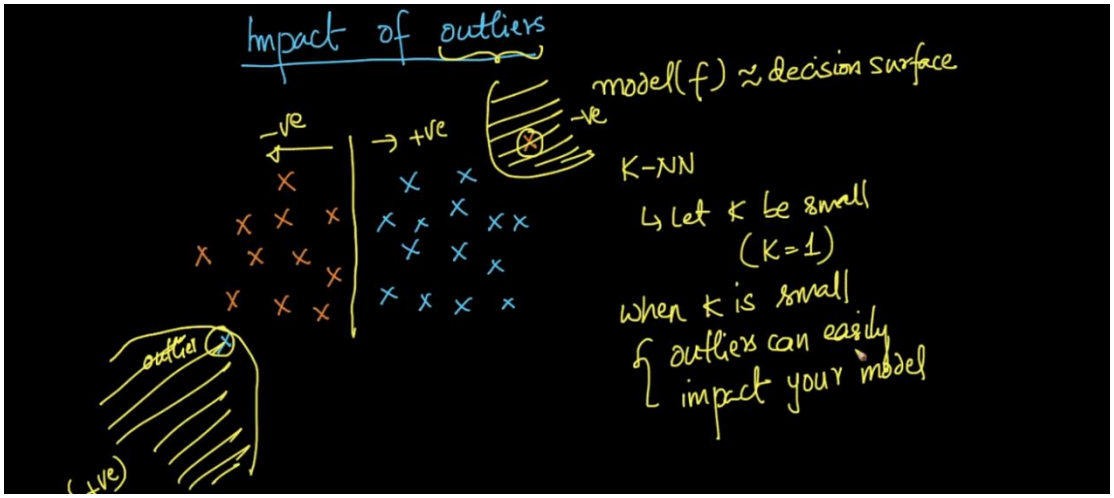


Another method is replace column with its mean value . like take height of people from India
And find mean of that .



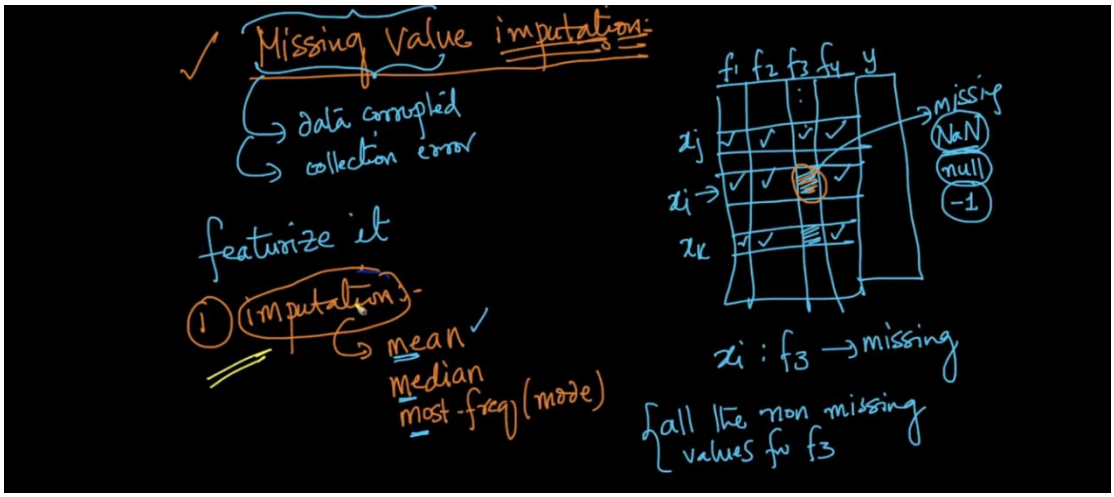
Else we can convert by domain knowledge .

There is no any fix logic for that we need to check as per our data . also we can experiment all method and check which one is best .



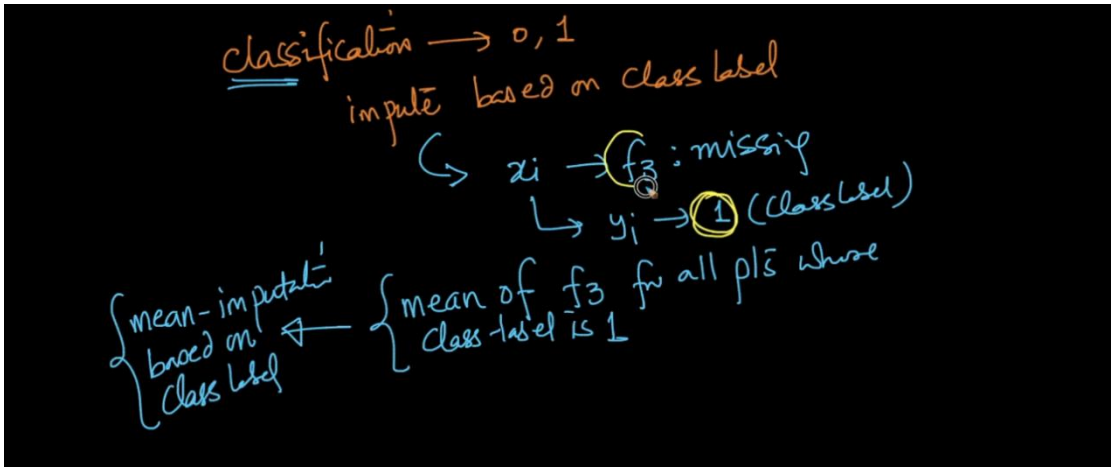
Always chose high k because out l with small K will disturb our model .

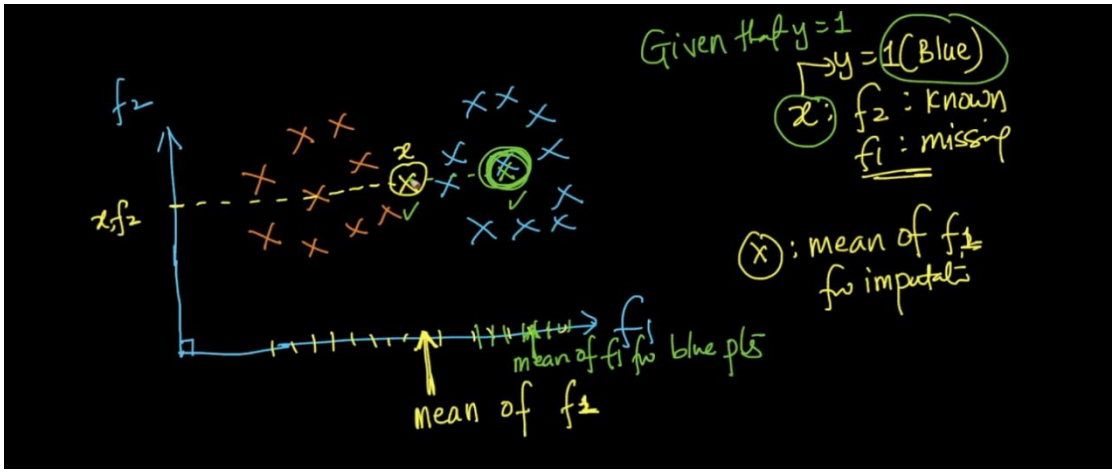
Missing Data Handle :



1st is we can simply take mean of that column and will replace empty value with that mean value .

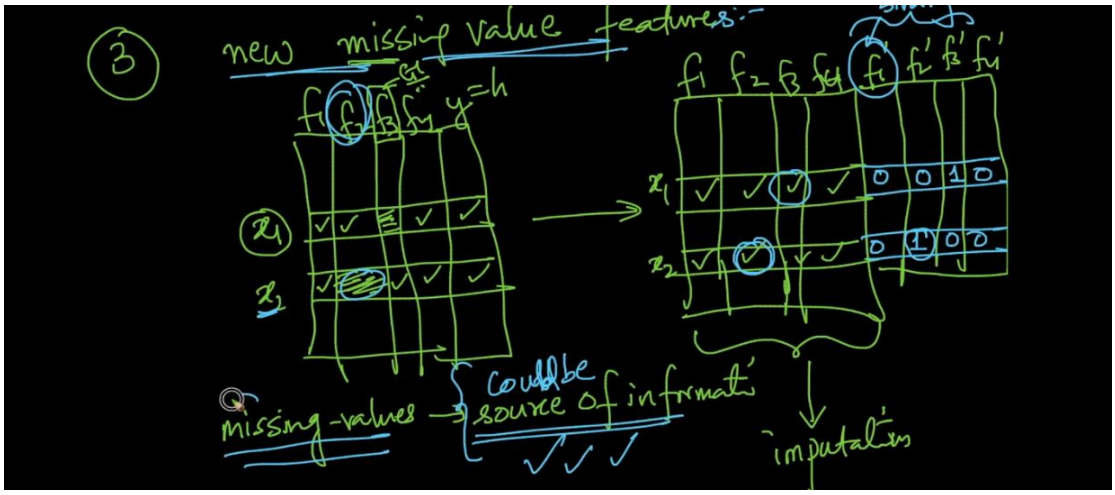
Mean Imputation using Class label .





Now instead of taking mean of whole column like f1 in above image we take mean of that class label area here class is blue so we take mean of blue area and that will very well than taking mean of whole column .

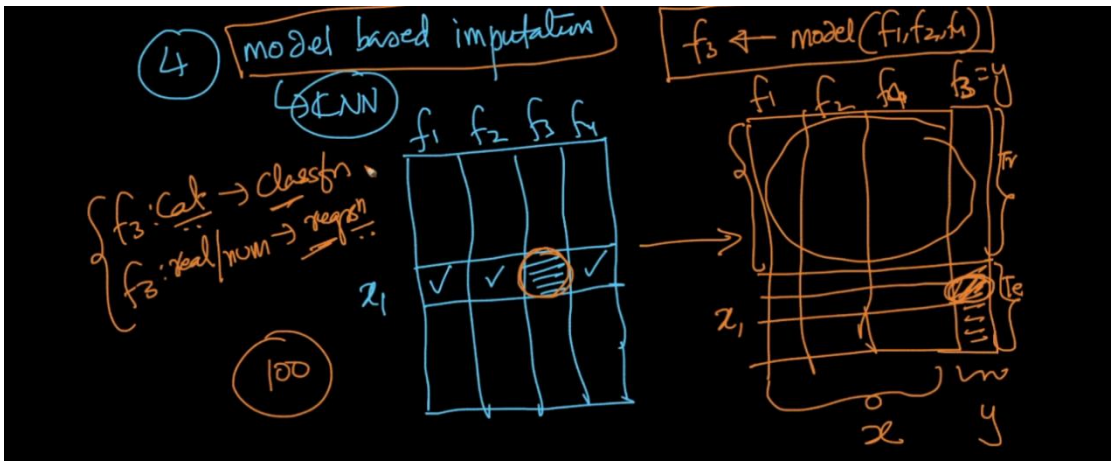
We took mean of blue because we already know that point belongs to that class.



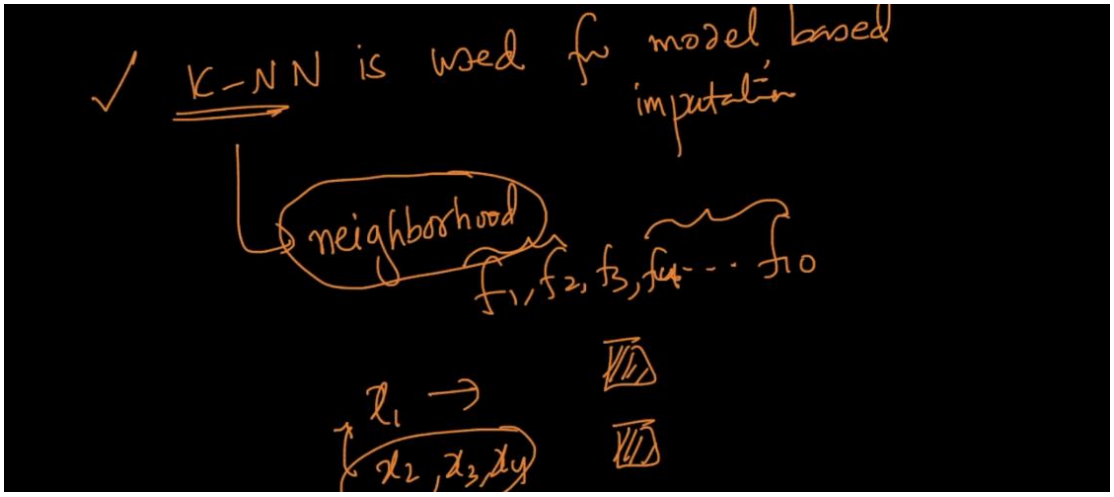
In above method we fill miss value by any standard method but after that we create another feature which keep details of missing value .

If u see above image for x1 f3 is missing so we store binary value like 0 0 1 0

Means f3 is missing this record some times helps a lot .



In this method whatever data has missing we make it as test data and other data as a train we implement any algorithm and then that model will predict missing value .



We can use KNN for the same .

Means lets say we 10 features and f3 Is missing then we will find all nearest point to x3 and if all points x2,x1,x4 is similar to x3 then it must similar in feature f3 also so we put value of that features in place of f3 .

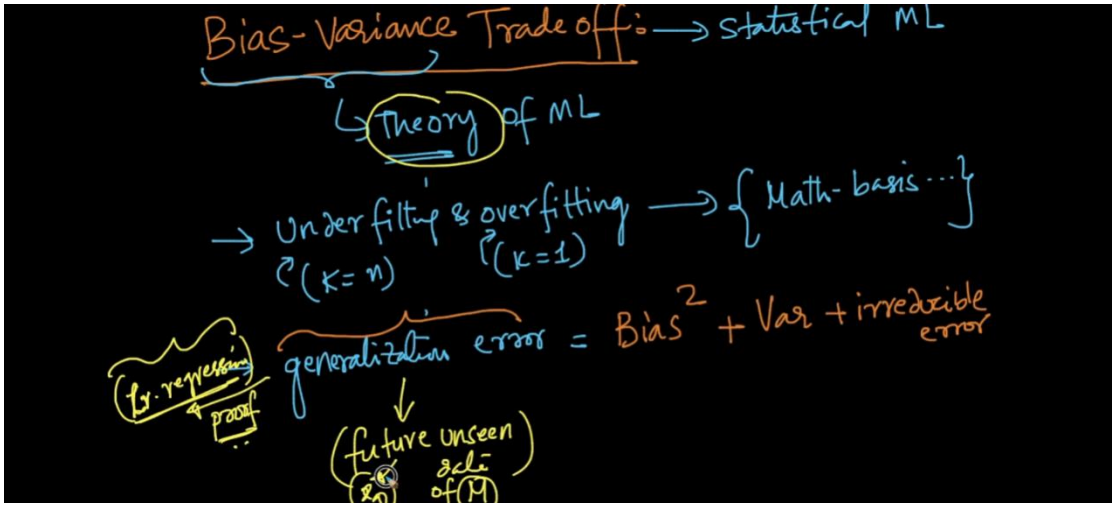
Curse of Dimension :

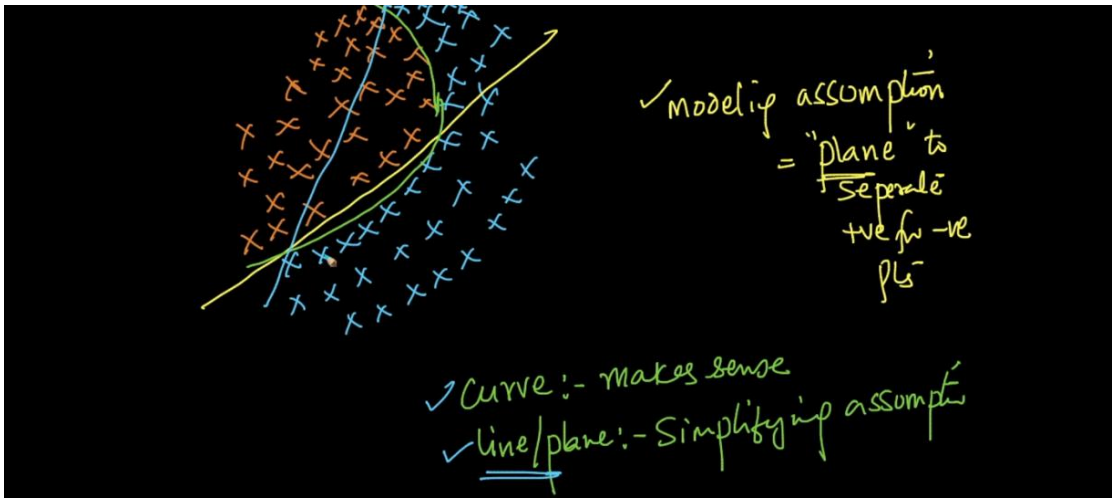
As dimension (features increase) no of data points also increase very fast to make a good model .

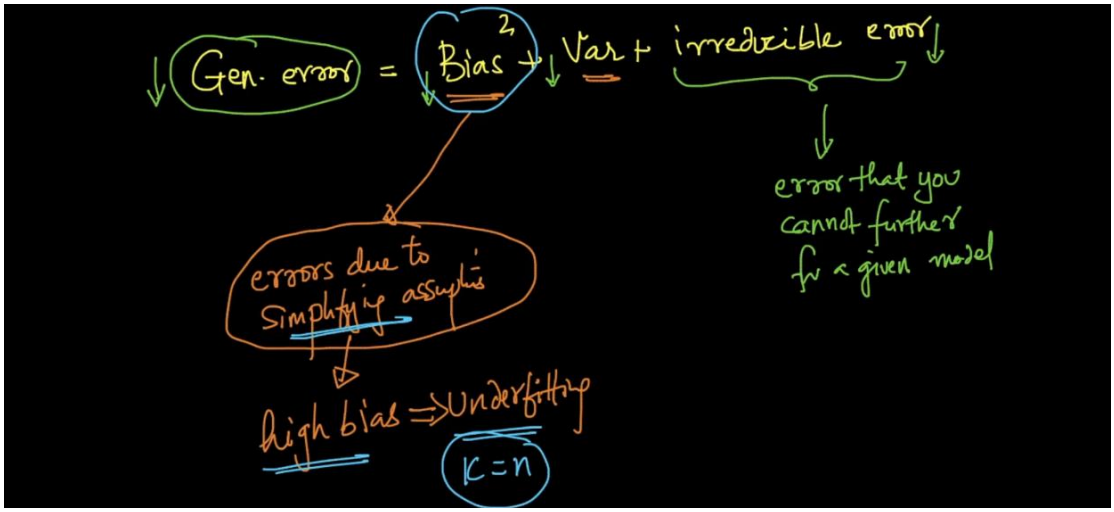
If no of data point are fixed but dimension increase then our model will not perform very well .

Need to watch video again .

Bias Variance Trade Off :





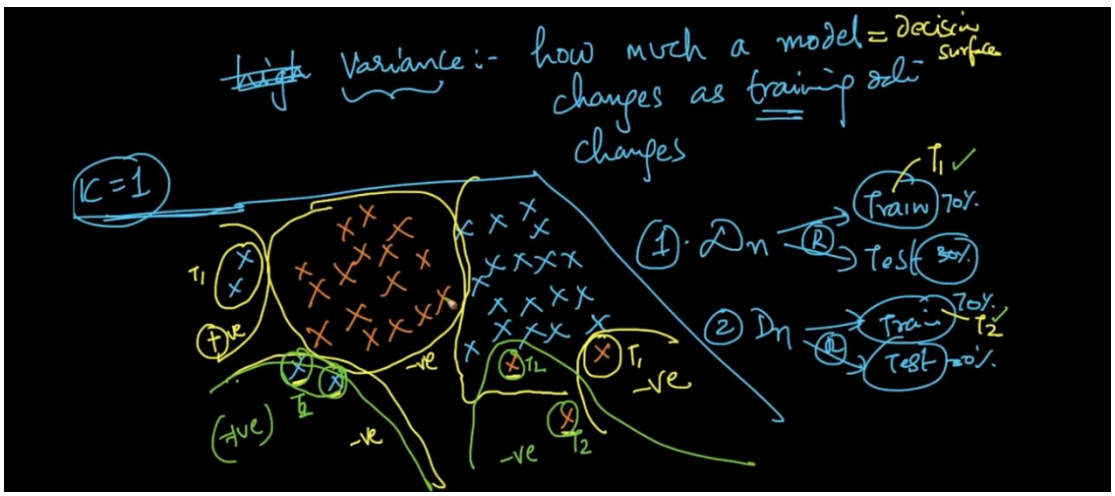


High bias is nothing but under fit model .
It occurs just taking simplifying assumption .

Lets say we have a data set + and - points now in that 80 % points are - and only 20% points are positive so we blindly predict any new point as - because of its high probability in data . We are not considering our 20% data and this is very worst called under fit .

It will occur when k = n means we we take k value which = no of data points .

High Variance :

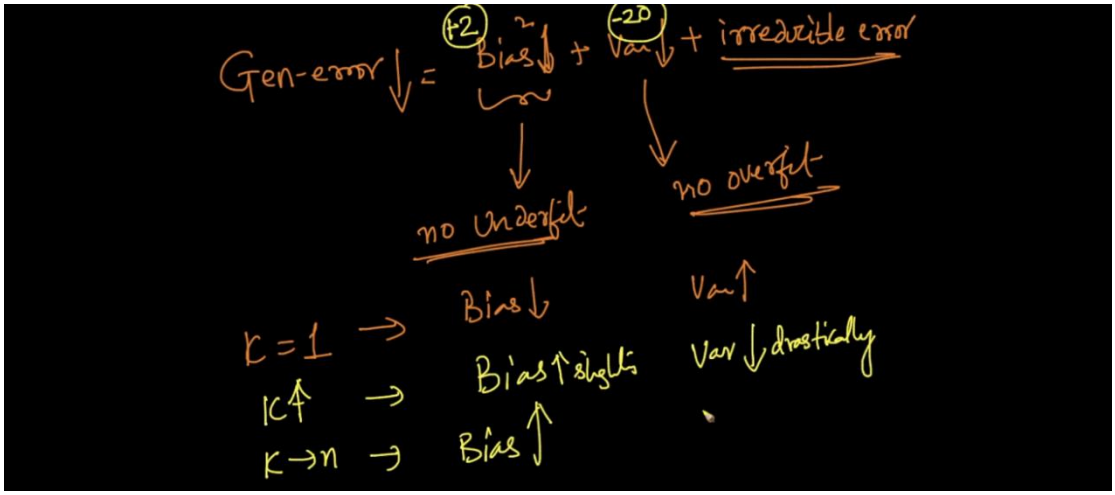


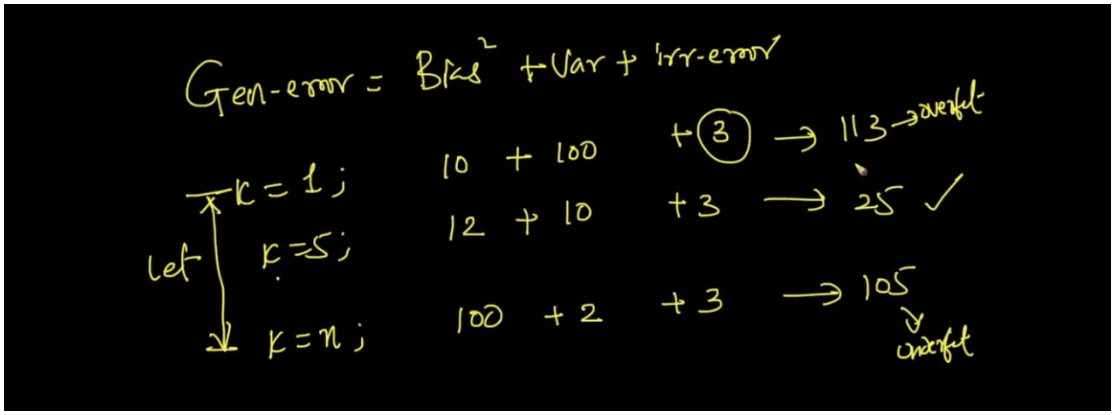
This is nothing but over fit problem ..

Also this occurs because of change in training data .

Lets say we have data D we split into train and test randomly .

Now we trained model on that data with k =1 if you see some out l are - but second time I trained model because of random data split we get different train set so in that same point becomes + this is the effect of High variance .



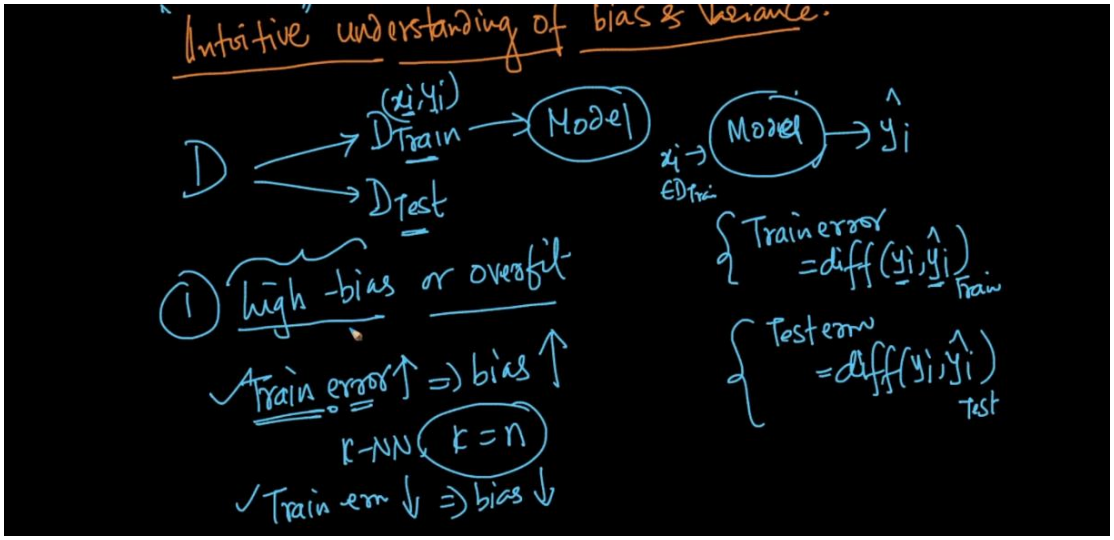


In this way we need to select proper K to avoid high bias high variance effect .

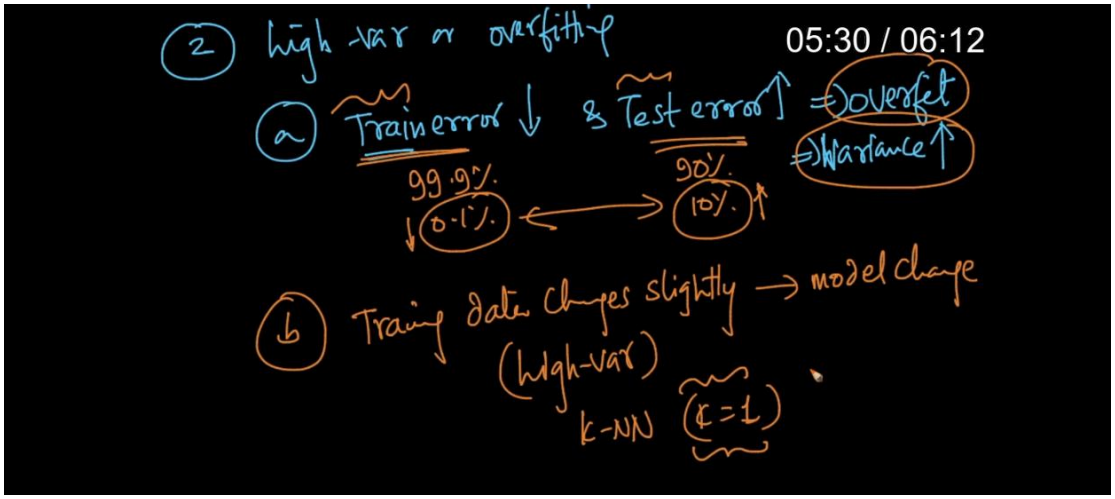
How can we say our model is high bias ?

Very simple if model on train data gives high error means it is high bias .

Means we are not considering 20% of + points just we are predicting each point as - because of its high probability k = n ..

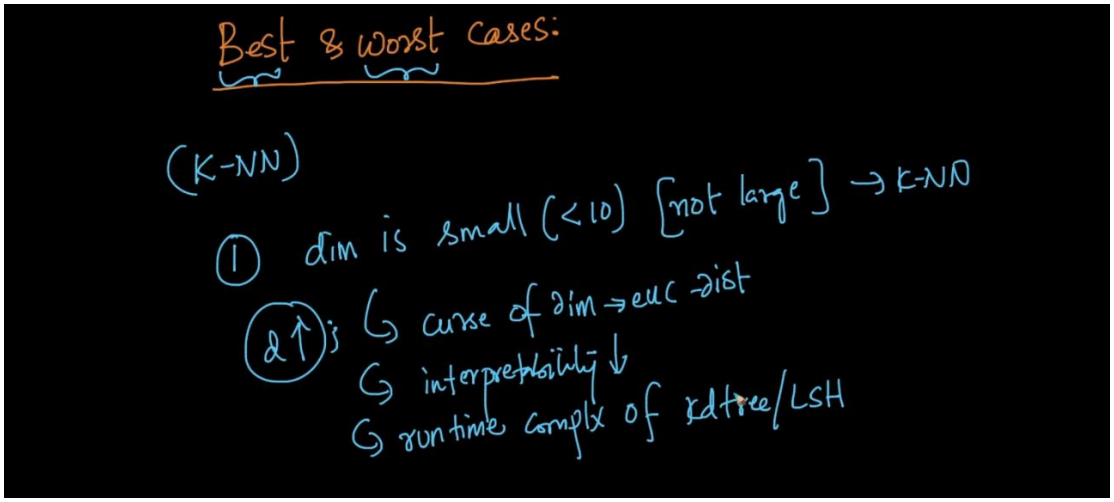


We can say model is high variance when our model train error is very less and test error is very high also called over fit .



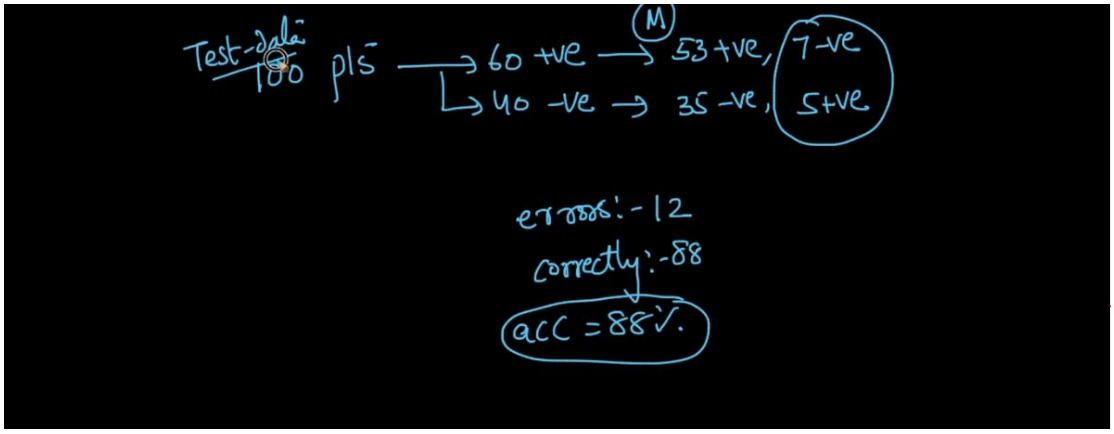
Best and Worst Case KNN :

When dimension are less then KNN is good algorithm .



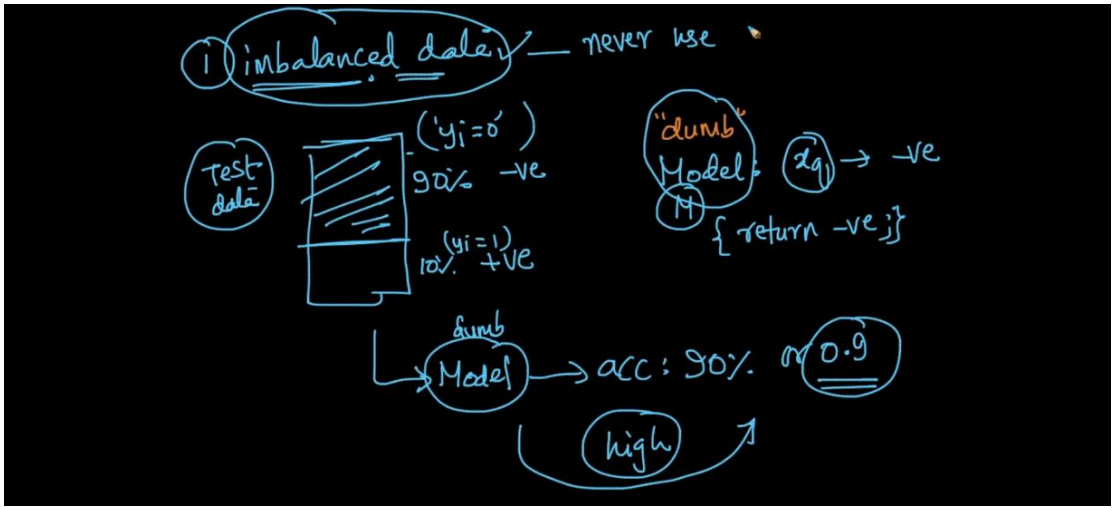
In low latency application KNN not used its time complexity is high .

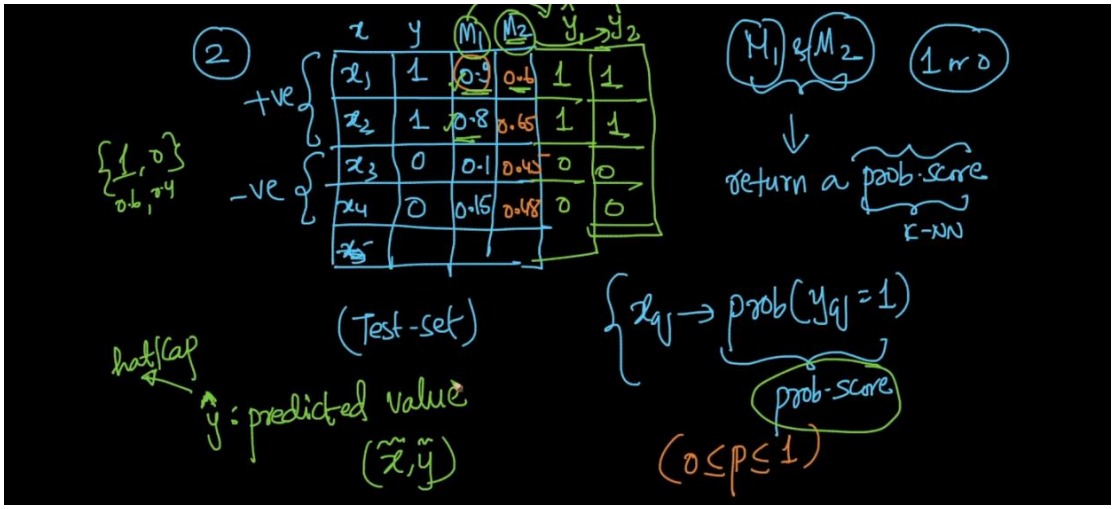
Accuracy :



As our model predicted 88 correct points out off 100 accuracy is 88 %.

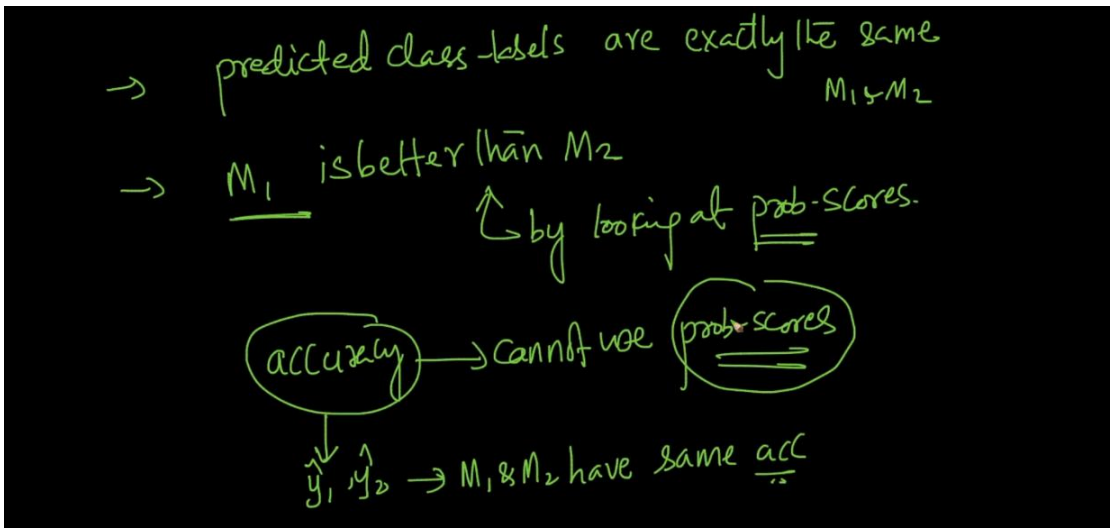
This is not a good measure when data is not balanced .



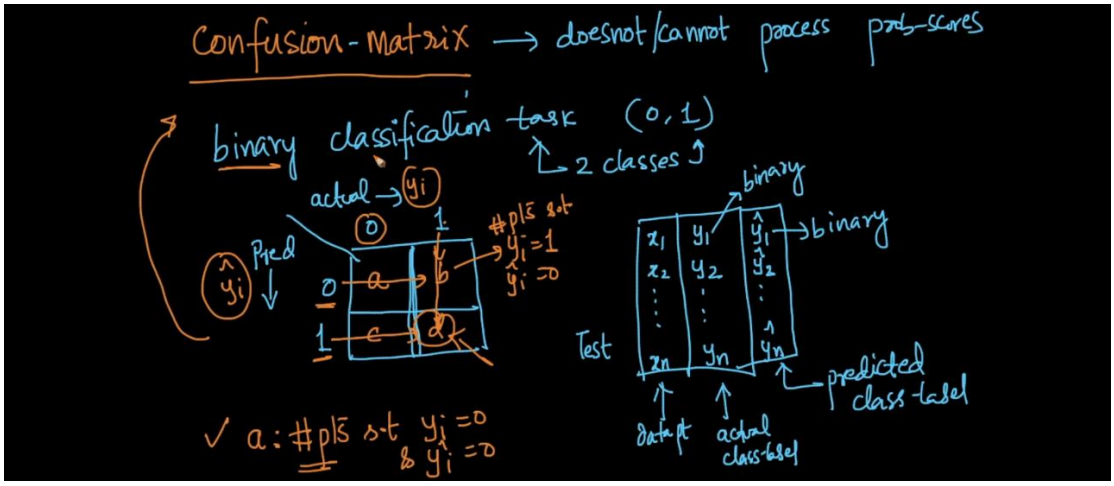


See m1 predict low prob for x3 and x4 means prob of being 1 is very less and its correct .

So compare both model as per prob score we can easily say m1 is too better than m2 but if we use accuracy matric then it will predict accuracy of both model is same hence both models are same but this wrong .



Confusion Matrix :



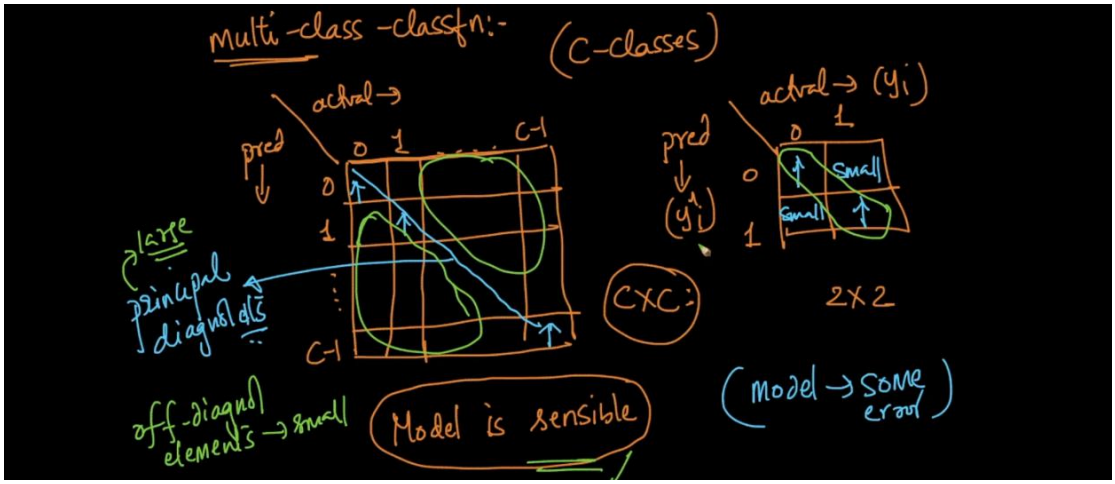
See actual value is 0 and predicted is also 0 means our model did great job . That is True Negative TN .

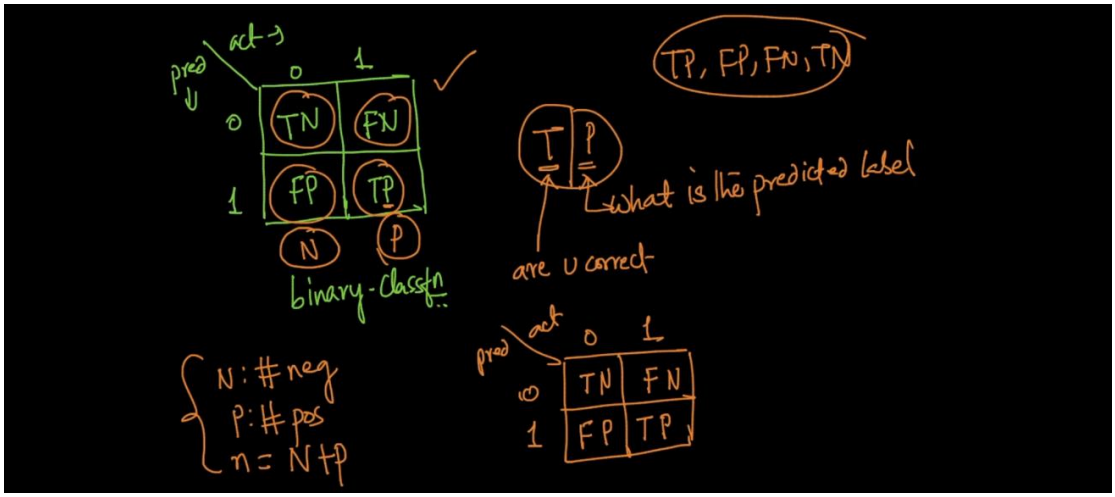
Actual value is 0 and model predict as 1 means model did a mistake called False Positive FP .

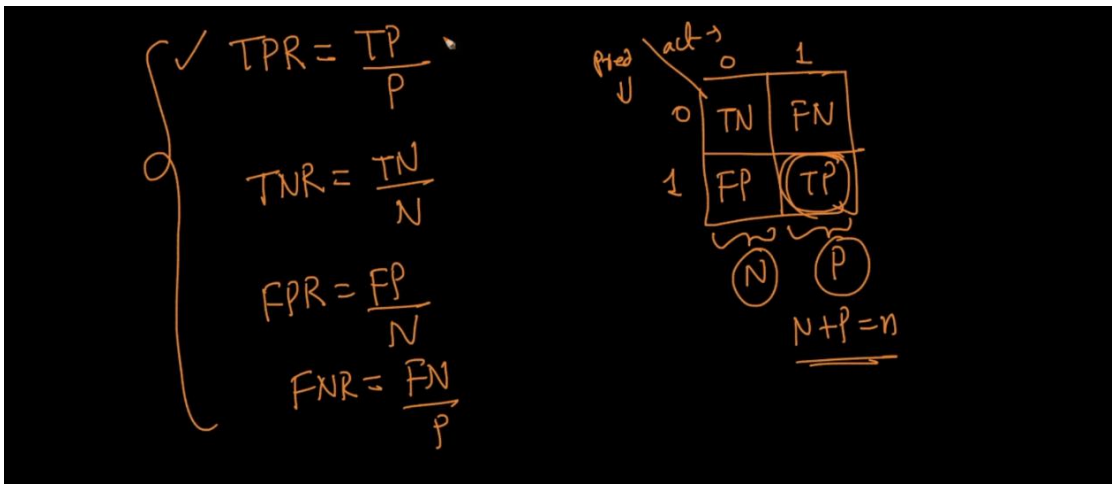
Actual value is 1 and model predict as 0 means model did a mistake called False Negative FN .

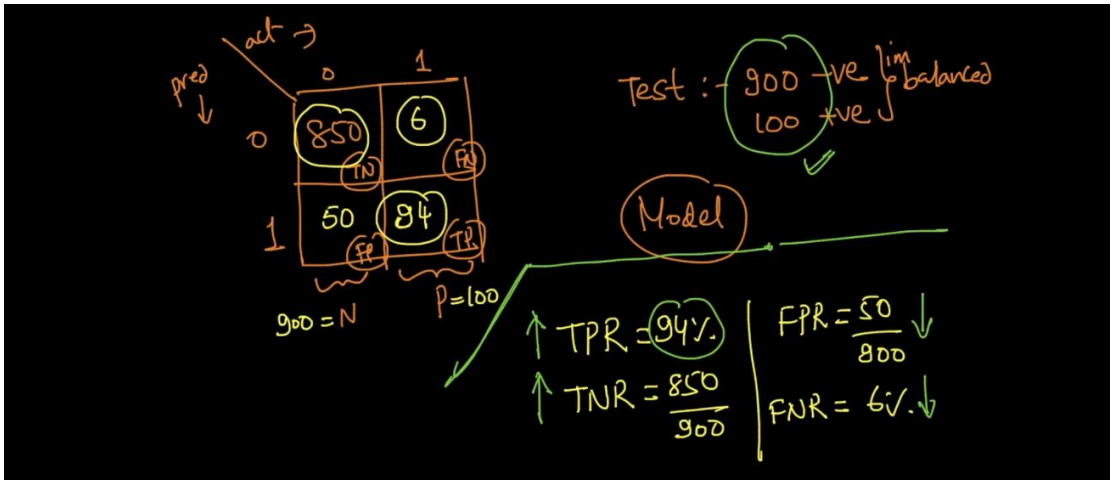
Actual value is 1 and model predict as 1 means model did a mistake called True Positive TP ..

So we need TP and TN values always high and FP and FN always low if so then only we can say our model working good .

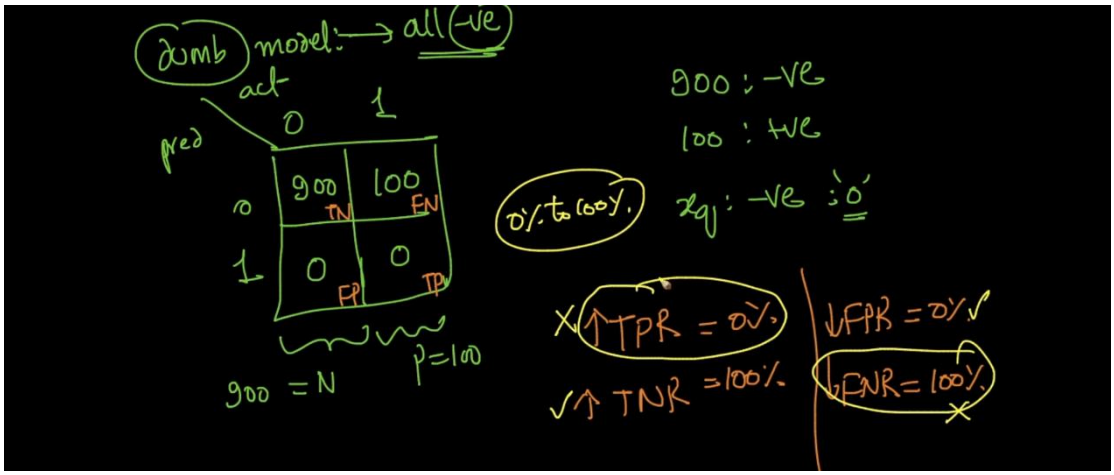

Multi-class-classfn:-
(C-classes)
actual →
pred ↓
0 1 C-1
large principal diagnol elements
off-diagnol elements → small
C×C:
Model is sensible
actual → (yi)
pred ↓ (yi)
0 1
Small
Small
2×2
(Model → some error)

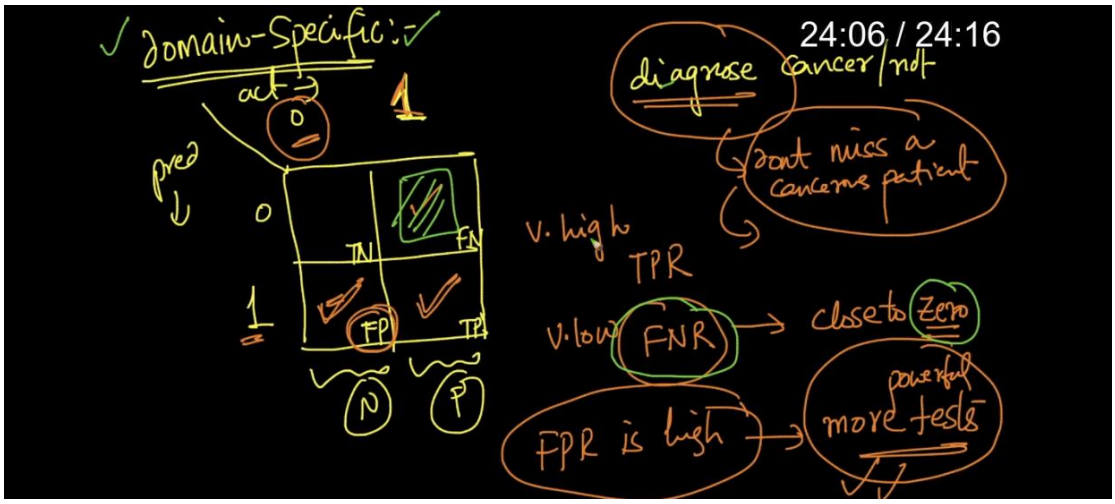

pred ↓
act →
0 1
TN FN
FP TP
N P
binary-classfn
TP, FP, FN, TN
T P
what is the predicted label
are u correct
N: # neg
P: # pos
n = N+P


TPR = TP/P
TNR = TN/N
FPR = FP/N
FNR = FN/P
pred ↓
act →
0 1
TN FN
FP TP
N P
N+P = n



See below image by just seeing that rates we can say our model working something bad .



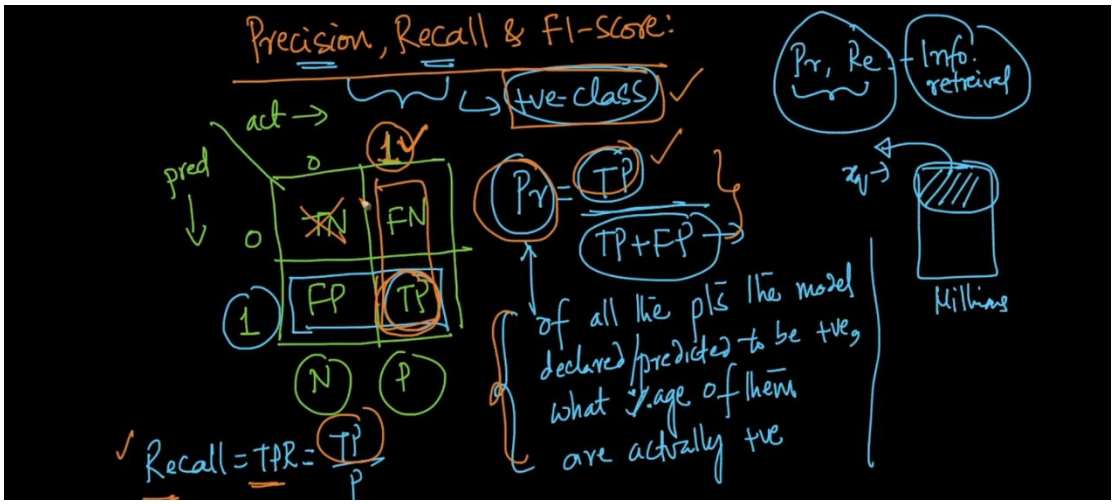


Above example of cancer or not . in this case we need TP rate very high means if patient has cancer then our model will predict yes he has cancer very sure ..

If our model predict wrong in this case then it would be very risky .

In this way confusion matrix helps a lot ..

Precision recall :



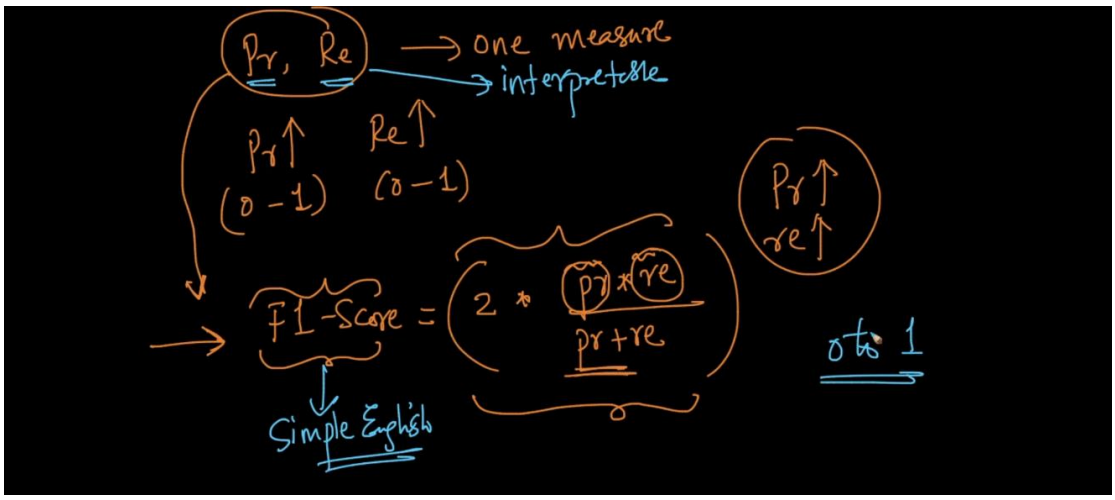
Precision or recall only cares about positive points .

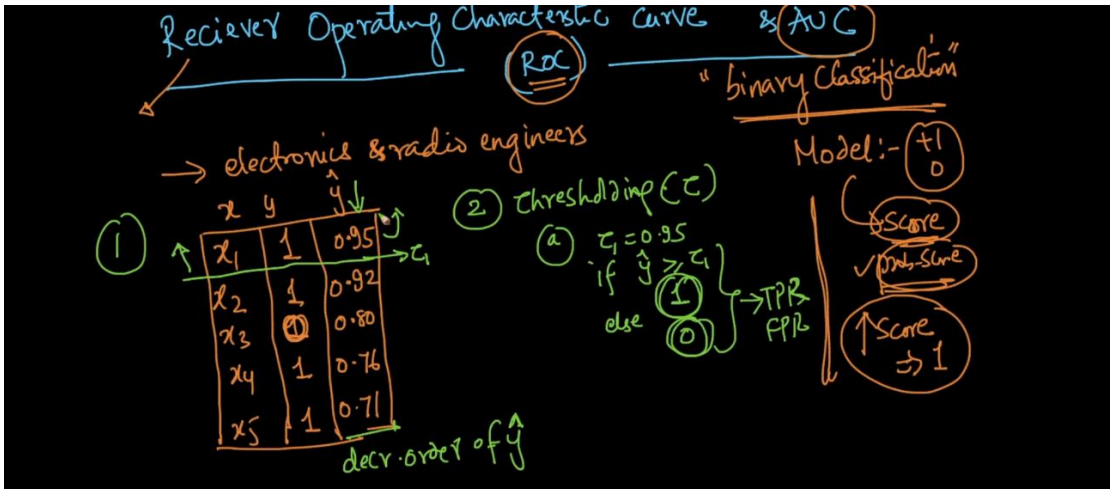
Precision means of all points which model predicted as positive ,out of that how much are exactly positive .

Recall means of all actual positive points how much much predicted as positive ..

F1 score cal by combing above 2 measures but its very easily understandable but some times it is useful in Kaggle competition .

We want P and R are always high and f1 also high .



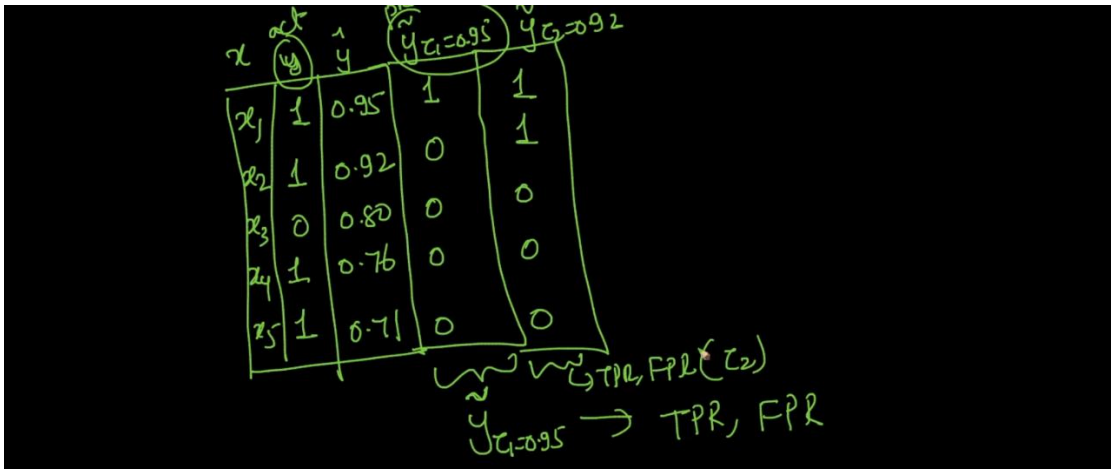


Lets say we have data x see above image and output y . so now our model predicted probability score of y =1 so we get value y1 … t5 .

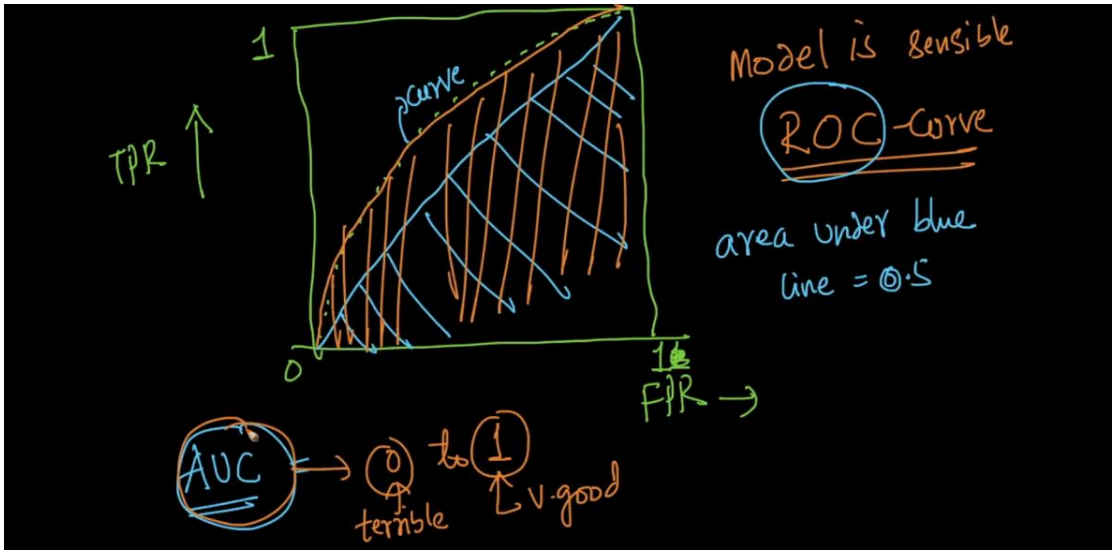
First step is we sort them in ascending order and pick first value as thresh hold called as tau 1 .

Then we we compare y^ with thresh hold if y>y^ then it belong to class 1 .

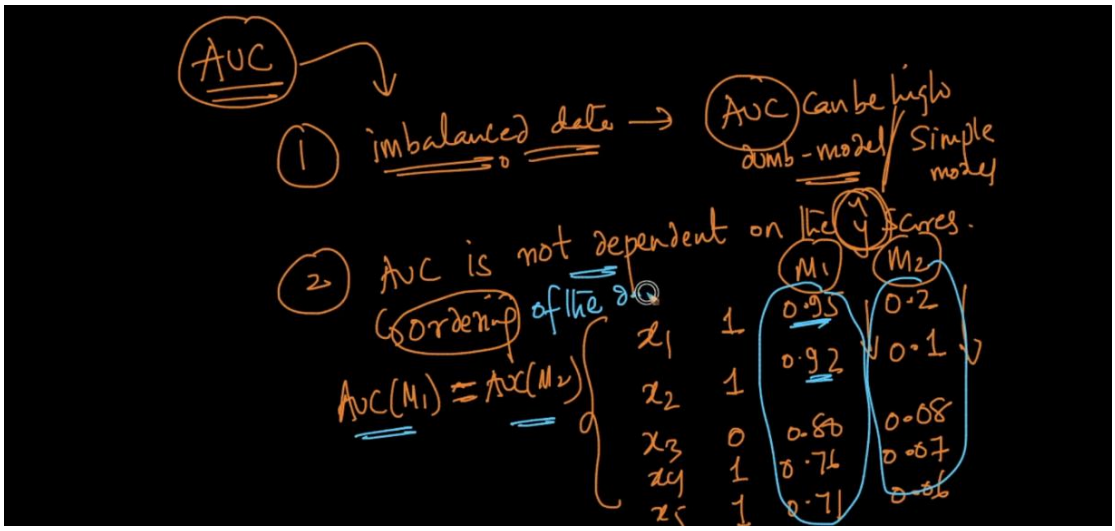
We repeat same process until n points .



For each tau1 …. tau n we find TPR and FPR and plot a graph its called ROC .



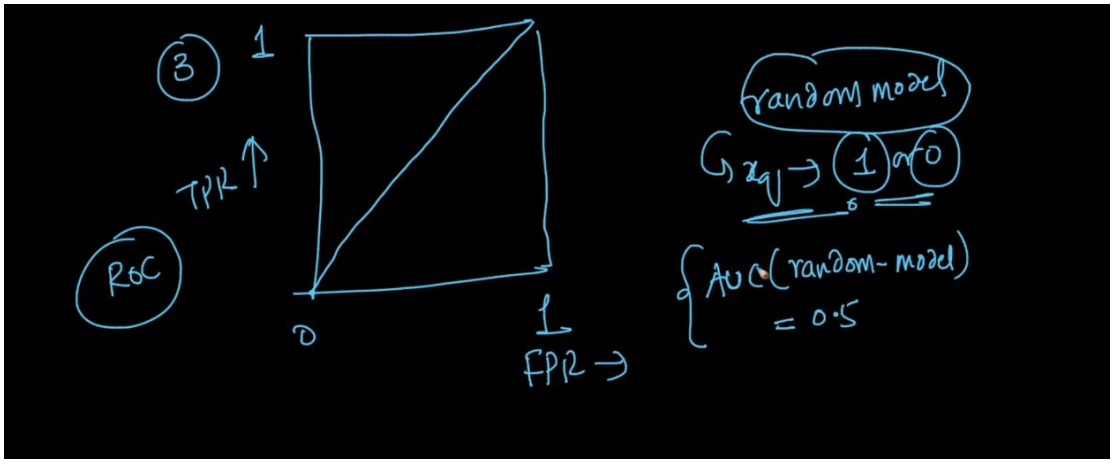
Properties of AUC :



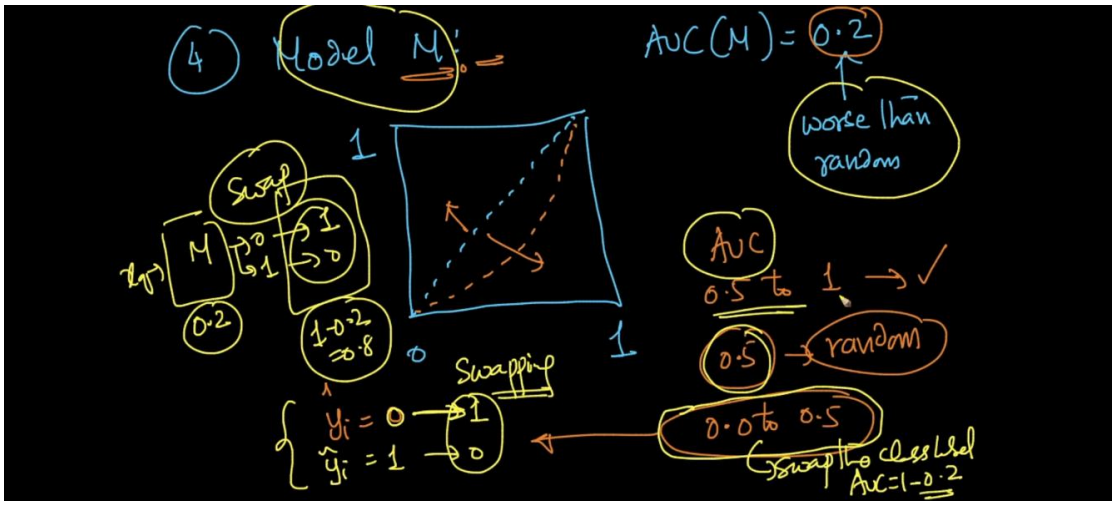
For not balanced data set AUC is always high .

AUC is not depend on probability score its just depend on ordering of score see above both model have same descending order then AUC for both will be same .

For random model AUC is 0.5

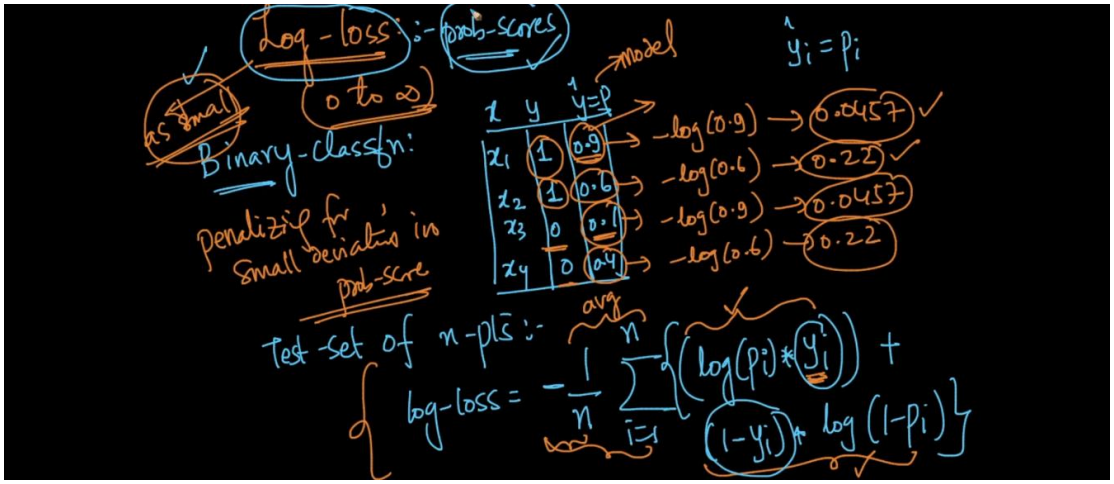


Means given any new point I am randomly telling its 0 or 1 for that AUC will be 0.5



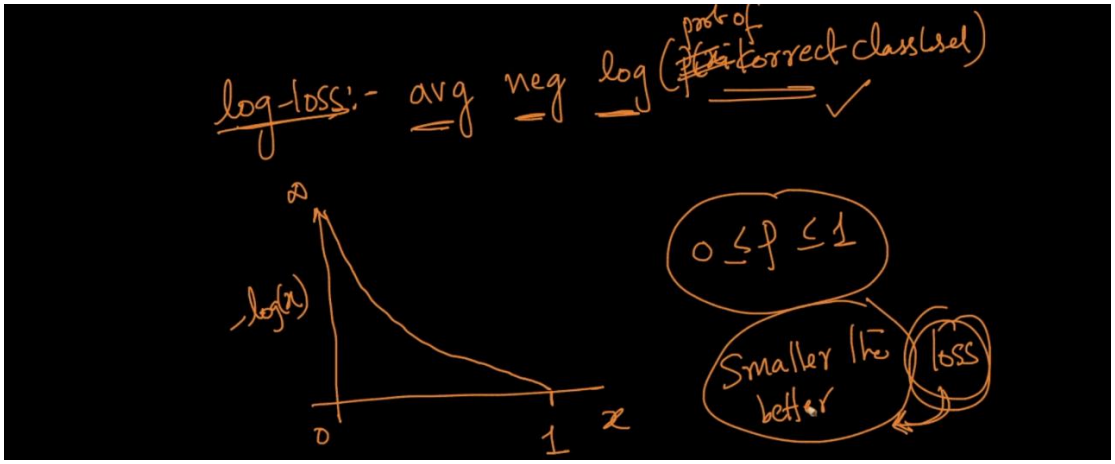
If any model has AUC below 0.5 then just swipe output . means if it predict 0 then change it to 1 vice v .

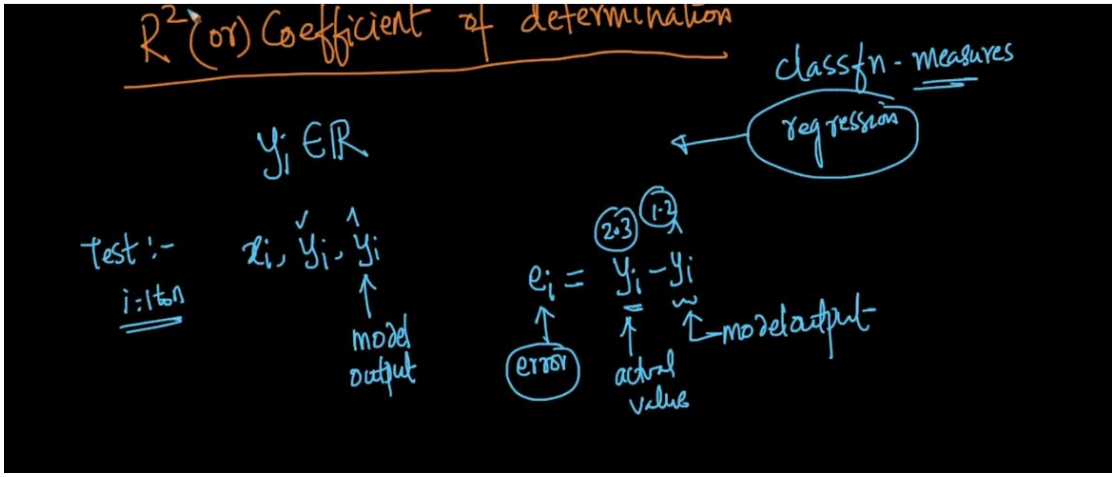
Log Loss :



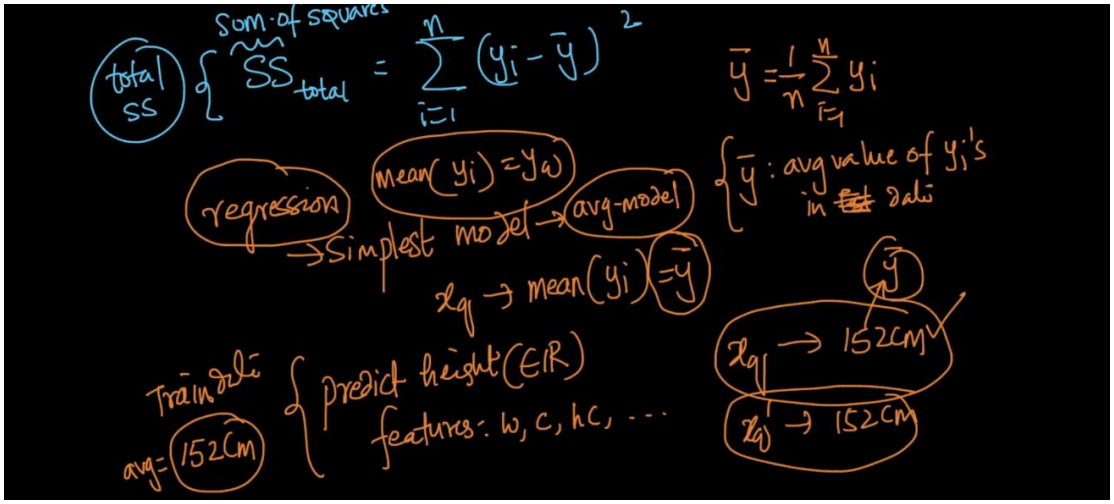
See above formula log loss use exactly all p values for y^ .
Now see above table as error increases log loss value also start to increase .
Log loss can lie between 0 to inf .
Log loss close to 0 is always best means in that case there is very less loss or error in our model .

Log loss is not to easy interpretable that's the only problem now lets say some one say log loss of my model is 1 or 10 what can we say its very difficult because it lies between 0 to inf .. and best case is 0 .



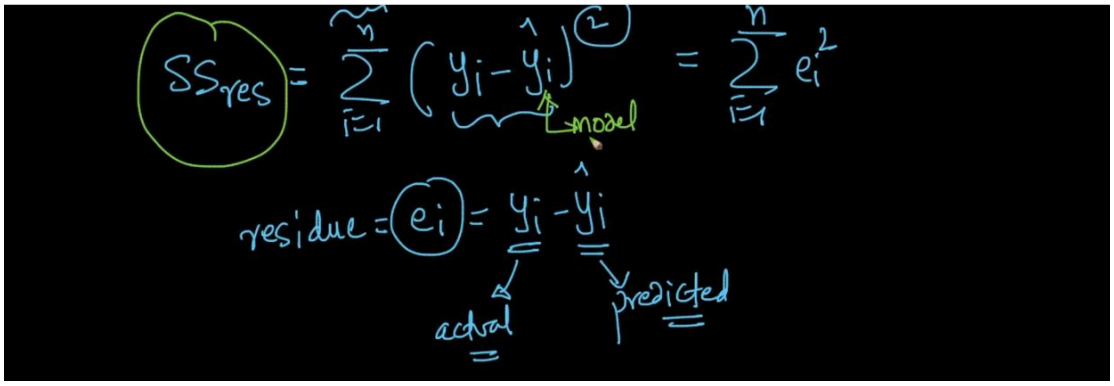


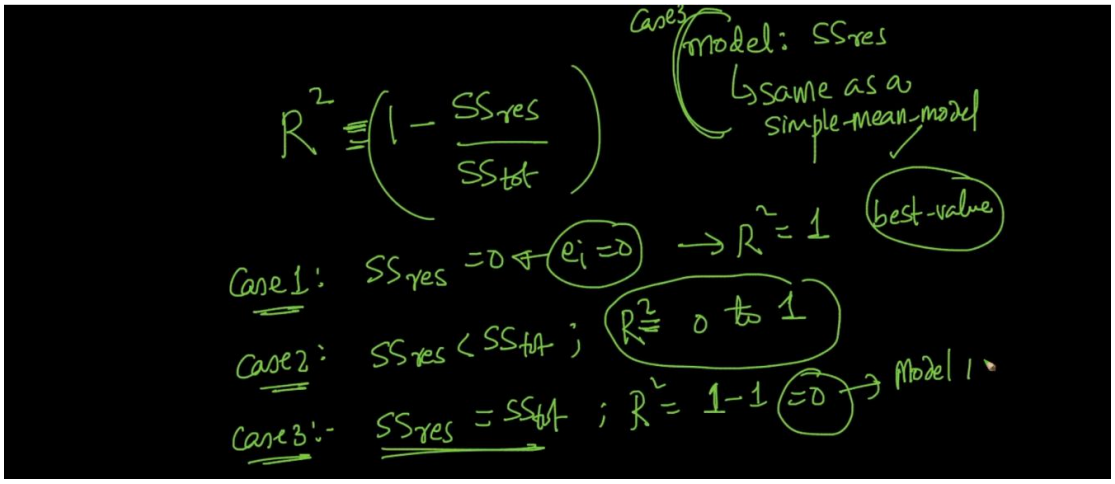
Very simple model we can design using regression is mean model means for each new point we predict mean value of y col .



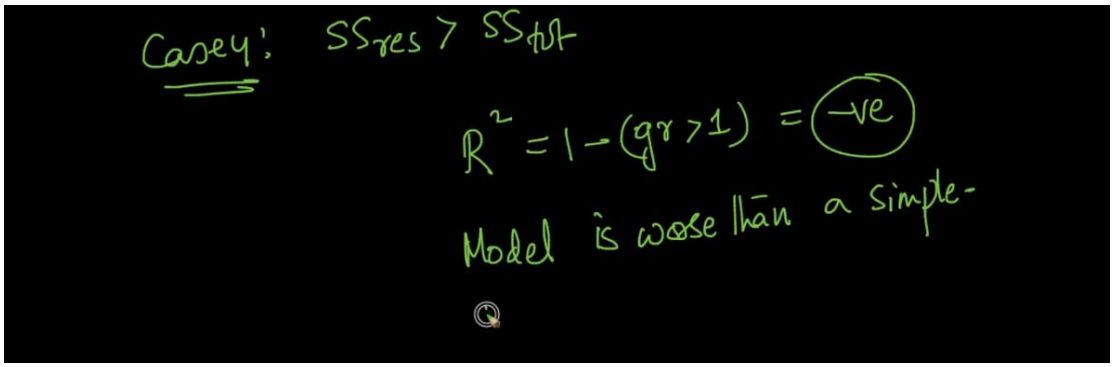
Y bar is mean of y col .

Sum if residual this is another term here we take difference between each y with predicted y^ .



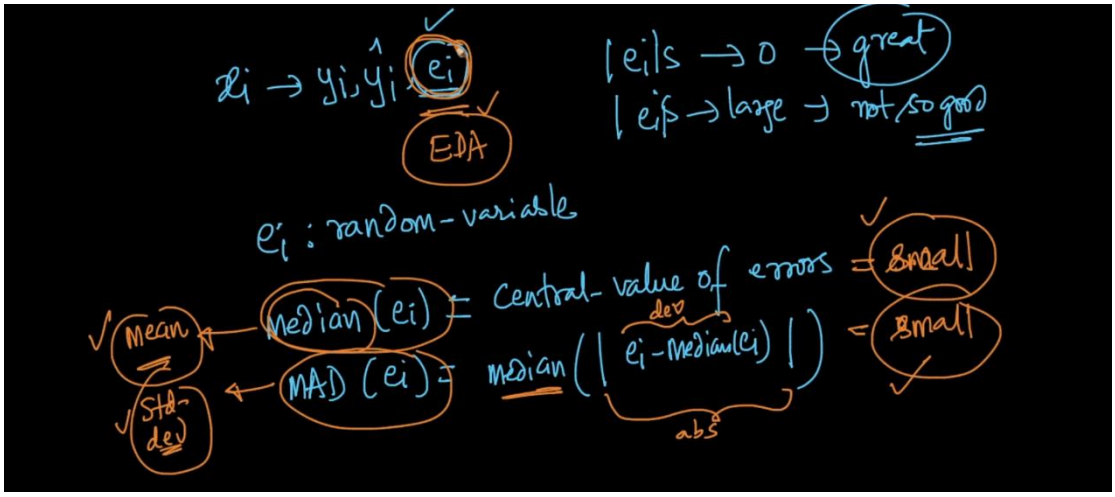


R2 close to 1 means model is best R2 close to 0 means model is worst .



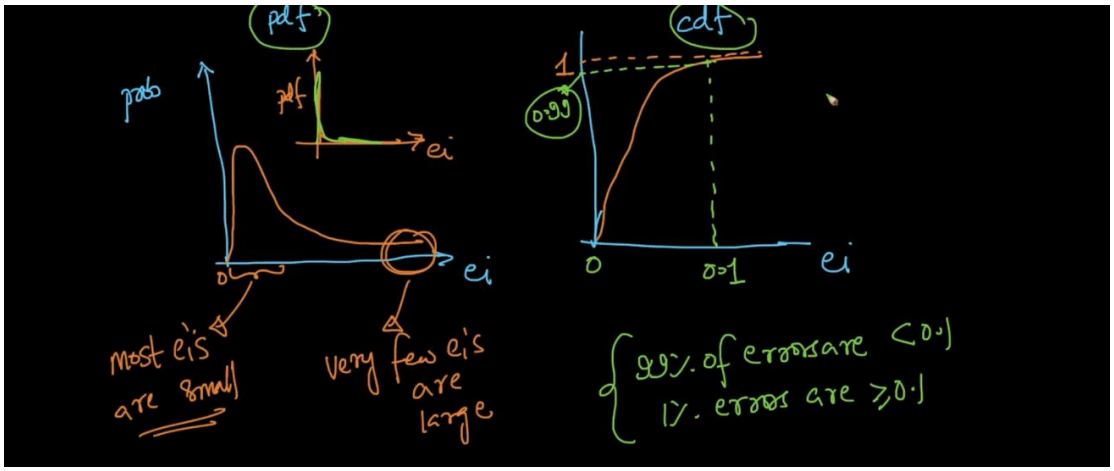
Now r2 is not robust to out l. if lets say there is 1 out l. in data so only for that point we get very high error value and just because of that whole r2 vary .

So we use Median absolute deviation (MAD) for each error .



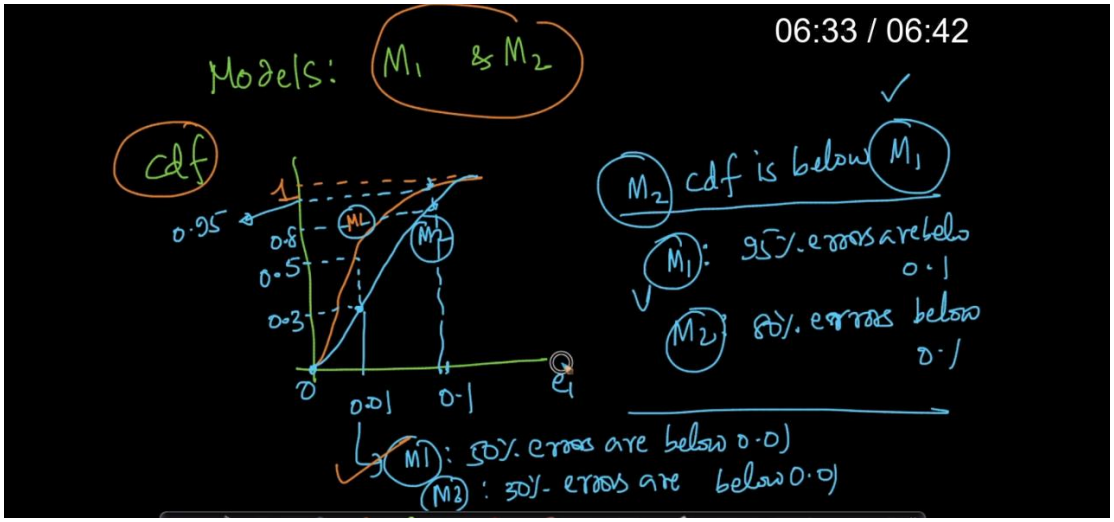
As we know that median and MAD not affected by out l.

By using PDF and CDF we can very easily understand model performance on error .



From PDF we say that max points has error is low and very few points have error Is high .

From CDF we can say 99% of value has error < 0.1 and only 1 % have > 0.1 error .


06:33 / 06:42
Models: M1 & M2
cdf
1
0.95
0.8
0.5
0.3
M1
M2
0
0.01
0.1
e
M2 cdf is below M1
M1: 95% errors are below 0.1
M2: 80% errors below 0.1
M1: 50% errors are below 0.01
M2: 30% errors are below 0.01