Bias and Variance

- linear regression create best fit line. It is not suitable for non-linear points. when degree if polynomial =1. Then sumation of all the error will be high. But, in case of degree of polynomial =2. it will perform best then first scenario. Also, in degree of ploynomial=4 all point will fitted excatly yo the line. So, we have these three scenario.
- So, taking about first scenario, the error is very high beacuse points lying far from best fitted line. So, if we create model on training data set, and it is giving high error. this cenario known as under-fitting. So, whenever, we create any model and we get very high error or less accuracy then it is known as problem of underfitting. So, in underfitting case, accuracy for train data goes down, also for test data aslo accuracy goes down.
- In third scenario, each and every points are satisfied with the line and giving very low error, or bestest accuracy, then this case is known as overfitting, when are model gives very very good result on train data, but when new data comes it fails.
- but in second scenerio, its good as compare to both.

Second points

- for first scenario, we have high bias and high variance. Just kepp in mind that, BIAS IS ERROR OF TRAINING DATA. VARIANCE IS ERROR OF THE TEST DATA.
- In the third scenario, we have low bias and high variance, beacuse model is giving quite good result on train data but it fail with test data.
- In third scenarion, we get low error for both both train and test data, so its example of low bias, low variance.

With calssification problem statement

3 scenarion

- in first, trainning error is 1% and test error is 20% - this is scenarion of low bias and high variance (Over fitting)
- in second, trainnig error is 28% and test error is 26 % - this is case of high bias and high variance (underfitting)
- in third, training error is <10% and testing error is <10% - we can say that this is best case when we have low bias and low variance

Respect to decision tree and random forest

Respect ot Decision Tree

- when we have under fittind problem, our error rate of train and test both will be high. In overfitting condition train error will be very low but for testing data it would be high. (Krish naik explaination).
- In decision Tree, it creates trees to its complete depth. This is like an overfitting condition, when you get low bias and high variance, beacuse we are taking all train data and splitting it into its complete depth. So, for over come this problem we try pruning(pruning referes to - after a particular depth tree will not split further) and hyper parameter tunnig.

Respect ot random forest

- in random forest, we use multiple decison tree in parrelel, we basically have scenarion of low bias, high variance respect ot each and every decion tree but as we are combining those decision trees in parralel, (bootstrap aggregation- in which we give random data to different n number of decion tree and finally, we get output using voting classifier), we get low bias and low varinace

In [3]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:
```
dt = DecisionTreeClassifier()
```

In [4]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [5]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [6]:
```
rf = RandomForestClassifier()
rfr = RandomForestRegressor()
```

In [1]:
```
from xgboost import XGBClassifier
```

In [2]:
```
xgb = XGBClassifier()
```

XG_BOOST

- For understading XGBoost, we have to understand some terminology.

Ensemble

- Means combine multiple model
- we try to combine multiple models and try to train the data. eg. random forest and xgboost
- In ensemble technique, It has two technique 1) Bagging 2) Boosting.

Bagging

- Also called as bootstrap aggregrator.
- random forest is the example where bagging used.
- In, bagging the n number of model generates, in which model select sample of records and send to all models.
- On thing is that, it does resampling after sending sample of record to first model and so on. known as raw sampling with replaccement
- After completing this step, we enter test data to our models and every model gives output. then it apply voting classifier, which means that if 0 is repeated more than 1, then final output will be 0 (for calssification problem).
- Bootstrap agrregation - the step where we use raw sampling with replacement is called as bootstrap.

Boosting

- In first step it create base learner, and train data are passes thorugh it. and once it is trained then it leans that importance of the features, or we can say it will create next model by using previous errors of base learner.
- It will go on upto given critarea. This is known as boosting

ADABOOST

- if we have features f1, f2, f3, f4, and output o/p.
- And, if we have 7 records then all records will get sample weight. w=1/n. so, initially all the records assigned same the weight. 1/7
- now, it will create a base learner with a help of decision trees. But in adaboost decision trees are created WITH ONLY 1 DEPTH. this decsion tree called as stumps
- For each and every feature(f1, f2, f3, f4). it will create stumps. and the stump or decision tree with less entropy selects as base learning model.
- Then, if we have eg. binary classificaiton and base learner give result as 4 - zero and 1- one. then, it will find out the total error for the 1 incorrect output by summing up all the sample weight - in this case it will 1/7.
- then, it will find out the performance of the stumps by formula [1/2 log2((1-TE)/TE)]. which will be 0.89. in this case
- then, model will increase the weight of incorrectly classified records and reduce the weight of correctly classified weight.
- to upadte weight formula will be - weight(previous) * e^ (performance=0.89) >>> 0.395
- so, weight was 0.14, which has increased to 0.395
- and updating correctly clasified points- formula is weight(previous) * e^ (-(performance=0.89))>>>> output -0.05 which has reduced
- then in order to get normlized value it devide all the updated weights with sumation of all weights, so all value will be convert to 0.07 except weight of incorrectly classified record. and sumation of all the value will be 1 as we normalized the weight.
- Then, it creates new data set based on the previous weight. we will try to devide it into buckets. then, model will run 8 iteration to select different different records from this particular older dataset. Then weights will get updated and process will repeat.
- At the end, final outpur dicide by majority or voting classifier.

COME BACK TO XGBOOST

with respect to linear regression

- In gradient boosting, sequential decision tree concept is used. in which residual(error) has multiplied with learning rate(0-1) in order to overcome the problem of overfitting. ho(x) + alpha1 H1 (x) + alpha2 H2 (x) +.....+ alpha(n)HnX

1. Find out the y^(y hat) for the base model by formula of loss.(initialized the model with a constant value)
2. Itrate 1 to m where , m= number of trees, and compute psudo residuals(calculating a error by (y^-actual)). Once we get resisual
3. We create a decision Tree where in depedent feature will be residuals. now, decision tree regrssor will be trained, residualas as depedent features
4. then we fit base model where depedent feature will be residuals through iteration.
5. then we again find the gamma. and the equestion says that we have to find out the y^ such that it will minimize the loss.
6. follow from the first step and at last Update the model.Fm(x) = F m-1(x) + alpha (h(x)), where, F m-1(x) is constant value that we got before trained base model, (h(x)) is value of residual, and aplha is learning rate

- and this is a sequential process where n number of decision tree performed and trained by using previos model's error.

#

K-nn Algorithm

- K-nn algorithm is basically algorithm in which nearest neighbours are calculated by using uclaidian distance(for p1(X1,Y1) and p2(X2,Y2) points distance formula = [(X2-X1)^2 + (Y2-Y1)^2]^1/2.
- when new data enters. it will consider the output of nearest n number of neighbours and using voting classifier, it gives final outlput for new data point.
- Also, distance can be calculated by manhatian formula

In regression, everything is simmilar

- instead of voting classifier, it select n number of nearest neighbours and the final output will be the mean of all the results of nearest neighbours
- impacted by outliers and imbalanced dataset

To check the best value of k in k-nn

- We find the accuracy rate and (error rate = 1- accuracy rate) with value of k ranging from 1 to 40. and the value of k, after which graph shows constant value of error rate, that point can be taken as k value.

#

Logistic Regression

- Logistic regression is basically used for binary classification

Why it is called as regression !

> ans is in logistic regression, we use logit function, where we plot a straight line and try to adjust the line inorder to get maximum likelyhood, and then we use sigmoid function to get probabilty or value between 0 to 1. So, this is regression, where we get binary output OR In another words, logistic regression is a linear method but prediction are transformed using the sigmoid or logistic function.
>
> - eg. based on weight(X) > categorise into obese and not obese (y)
> - linear means y = mx + c (where m is slope and c is intercept)
> - in logistic, consider the condition > if y >= 0.5 (cutoff) then OBESE and if y <0.5 is NOT OBESE.

if I would able to classify this by stright line then what is the use of logistic regression ! (refer krish snap)

- if there is outlier eg. weight= 150, then best fit line will change, and sue to that it will affect result. and this will resulted in high error rate.

- And second major problem is, eg. if we have high value of weight then according to best fit line y value become more than 1, same for very less weight, it would be less than 0 (Negative).

Two reason

1. when we have outliers, best-fit line deviated.
2. We get output greater than 1 and less than zero, mostly ##### that is the reason that we shoul not use linear regression for this kind of classification problem.

For that we have to use sigmoid function. (1 / (1 + e^-z)) or e^z / (1+ e^z)

> Going Ahead

Assumptions:

- All the positive points denoted as +1 and negative points denoted as -1 , not zero.
- so, y = mx + c >>> y = w^t * x(where the w^T is the distance between the point and plan or line)
- distance above plan will be +ve and below will be -ve.
- cost function - (sumation)i=1 to n Yi Wi^T Xi should be maximum, beacuse (Yi * Wi^T Xi >0, it will classify all points correctly). >>>>Wi is coeeficient unless and untill i get maximum of sumation.)

- use sigmoid function (1 / (1 + e^-z)) or e^z / (1+ e^z) where z is Yi * Wi^T Xi
- sigmoid function transform any value to 0 to +1.
- It removes the effect of outliers.
- sigmoid curve tells the probability based on the x variables. ### curve of sigmoid function would not let the value of Y more than 1 or less than 0 if the value of x variables is too high or too less

- In linear regression, best fit line is selected as final, where in logistic curve with maximum likelihood is selected

Using log of odds or Logit function

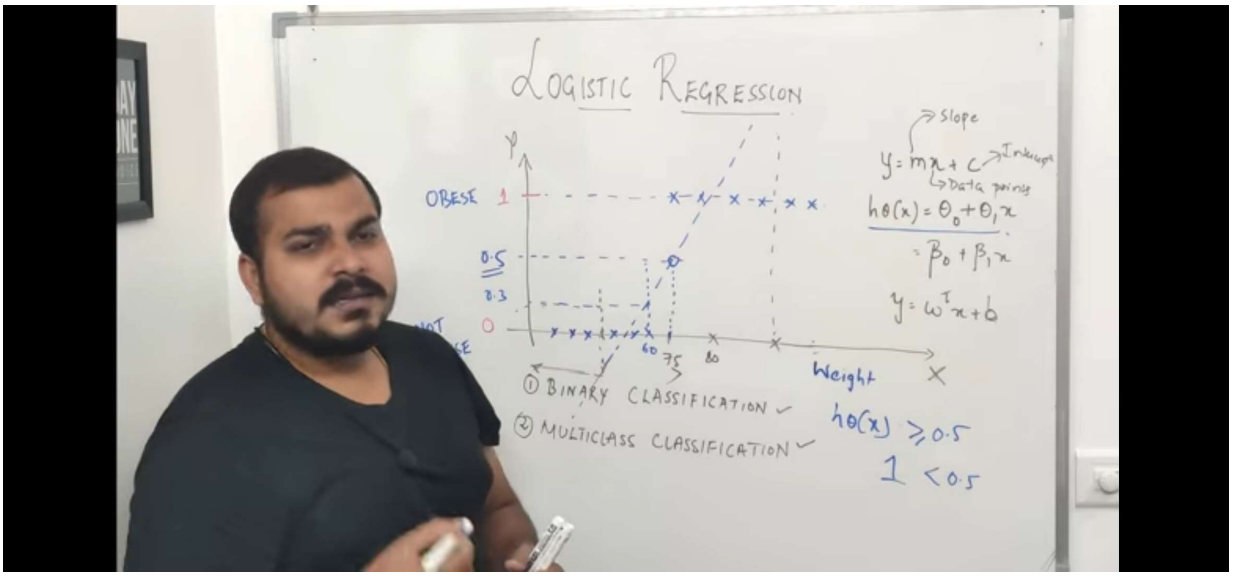
- Log(p/1-p) logit function
- eg. if probability is 1 then according to the formula log(1/ 1-1)>>> log(1/0) >>> + infinity and same for negative infinity.
- logit function will set data points at positive and negative infinity, then we choose best fit line
- after choosing best fit line we project the original data points to the candidate lines which are at positive and negative infinity
- this gives sample log odds value
- now, use the equestion of sigmoid, which is p = [e^(log(odds)) / 1 + e^(log(odds))] which is just reordering of the transformation from (Statquest video)

Learning:

- Learning logistic regression co-efficient is done by using maximum likelihood estimation, to predict value close to 1 for default class and close to zer for the other class.

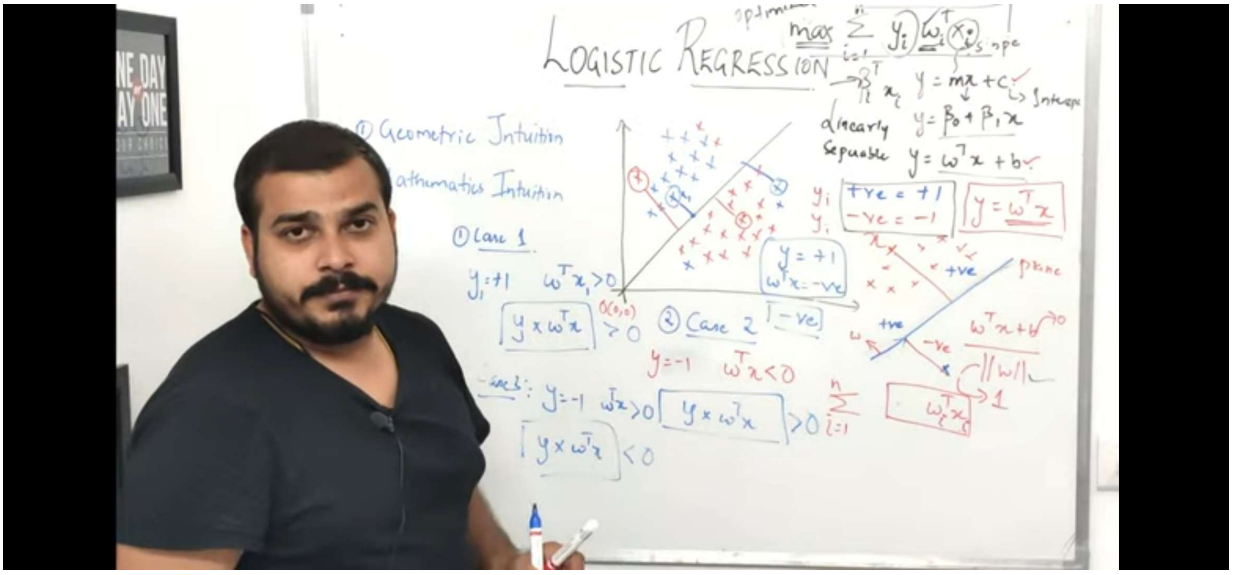
In [4]:
```
from IPython.display import Image
Image(filename='lg1.jpg')
```

Out[4]:



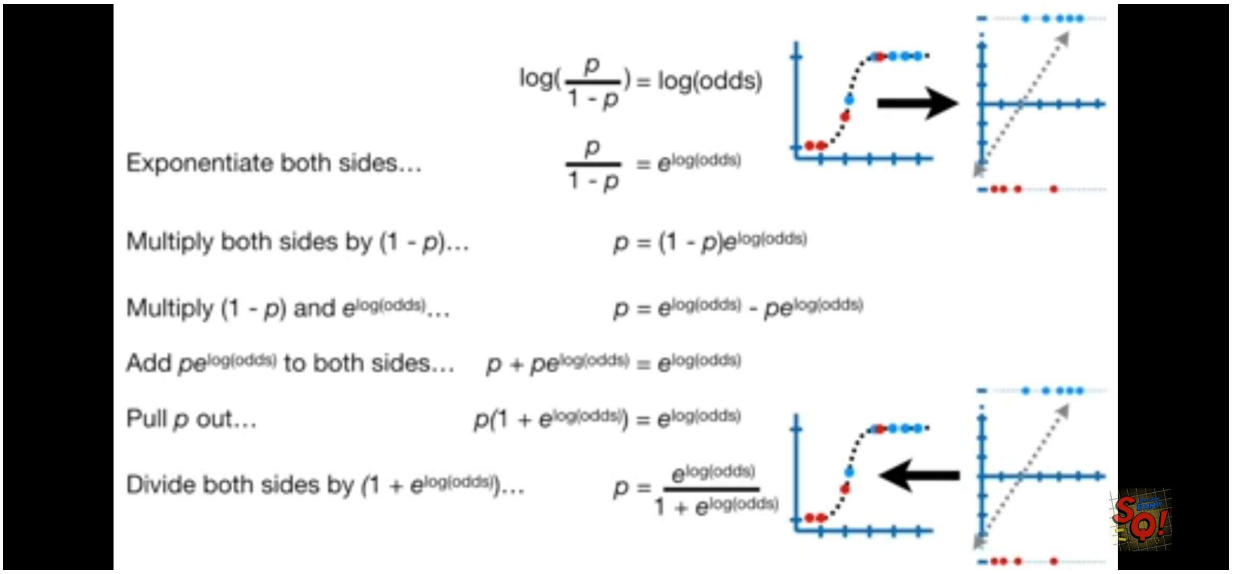
In [5]:
```
from IPython.display import Image
Image(filename='lg2.jpg')
```

Out[5]:



In [6]:
```
from IPython.display import Image
Image(filename='lg3.jpg')
```

Out[6]:



#

Linear Regression

- simple linear regression is Y = mx + c or y = a + bx,(m,b=slope, c,a=intercepts)and by using this equestion we find best fit line.
- Here, m indicates that if value of x variable increase by x then what will be the effect on Y-variable. eg. y = 3x + c means if x variable will increase by 1 unit , then Y will increase by 3 units.
- Another thing, that sumation of distance of all data points and regression line should be minimul. the regression line which give the minimul mse value, that is known as best-fit line.
- COST FUNCTION - in linear regrssion, (1/2m) sumation (i =1 to m) (y^(hat) - y)^2, where y^ = mx + c

Gradient Decent birth

- value of x is 1,2,3,4.
- when m(slope) = 1 and intercept =0 beacuse y = x >>> y and y^ value would be the same >>> which gives cost function 0.
- when m(slope) = 0.5 and intercept =0 beacuse y = 0.5 *x >>> which gives cost function 0.58
- when m(slope) = 0.3 and intercept =0 beacuse y = 0.3 *x >>> which gives cost functionwill be more than 1
- when m(slope) = 1.5 and intercept =0 beacuse y = 1.5 *x >>> which gives cost functionwill be 0.58 #### Now, if we plot a graph **cost value Vs slop(m)** it would form a ball shape curve, which is nothing but gradient decent. (krish naik ss)

When should you know that you should stop for selecting a m value, which looks good for best-fit line !

How do we arrive to the global minima ! a point in gradient decent curve where we get minimul value of cost function

- in order to move down in gradient decent, we use convergence theorem, which says that m= m - (Dm/dm) *learning rate, Here Dm/dm = derivative of m respect to m(slop). and one more point is , when we find a slop of particular point in gradient decent curve, if right hand*

of the slop is pointing downwards then that is negative slope. then >>>m= m- (-ve) learning rate (small) >>> m + smaller value.

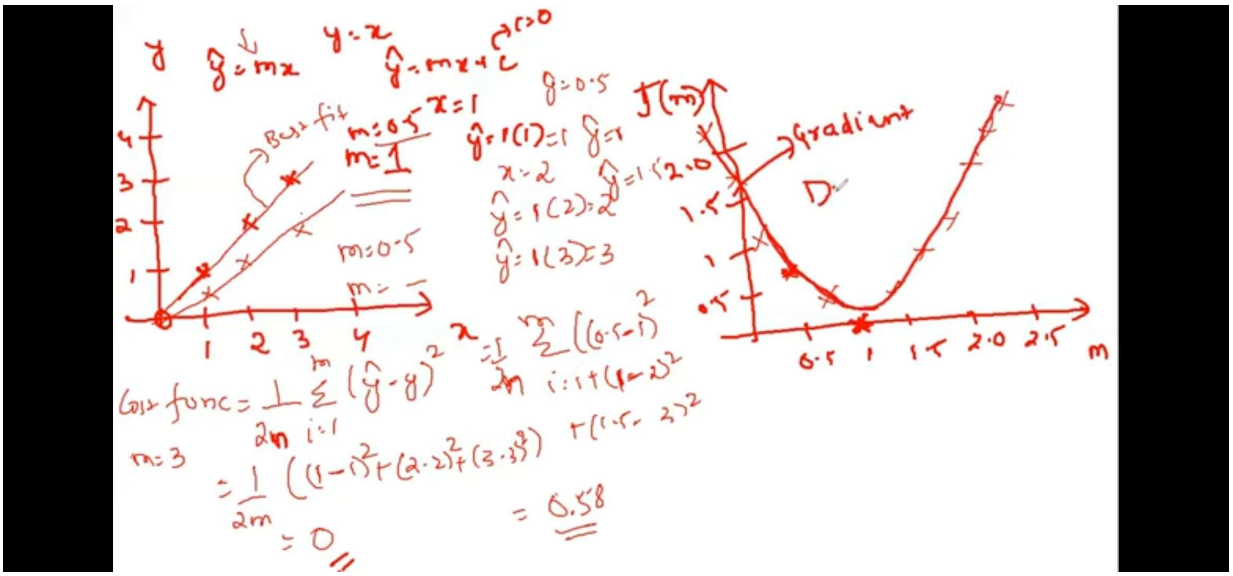
- This indicate that m value should increase and come nearer to 1, where we get minimul value of cost function.
- when we get a minimul value of cost function in gradient decent, and if it would increase then my m = m-(+ve) * learning rate>>>>> m = m - smaller value >>>> which indicate that my m value should decrease.
- convergence would not let the m value increase after it reaches at minimul value of cost funciton (MSE).

Why we do not take value of learning rate is high !

- answer is, if we take a learning rate value high then cost function would not reach globle minima in gradient decent, beacuse it will take bigger steps

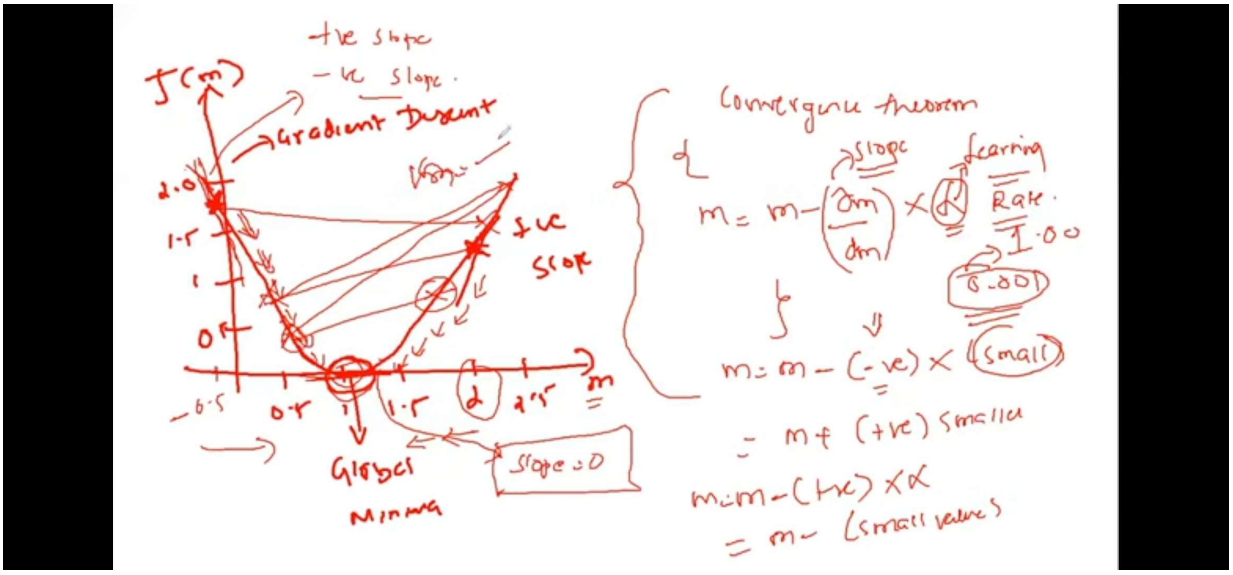
In [7]:
```
from IPython.display import Image
Image(filename='linear1.jpg')
```

Out[7]:



In [8]:
```
from IPython.display import Image
Image(filename='linear2.jpg')
```

Out[8]:

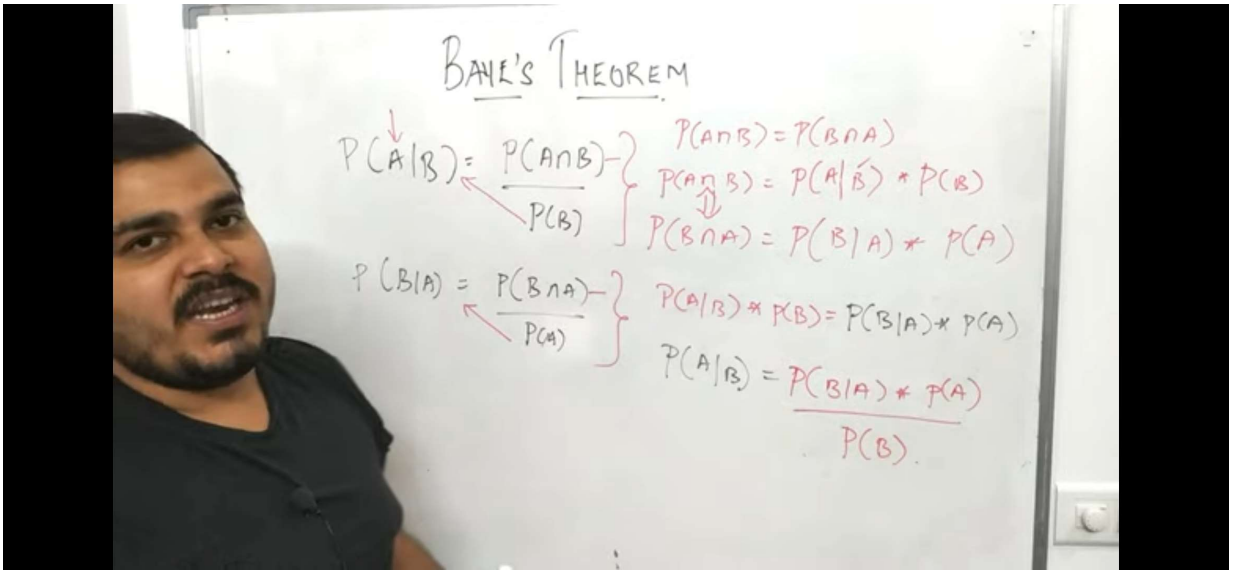


#

Naive Bayes Classifier

In [1]:
```
from IPython.display import Image
Image(filename='nb2.jpg')
```

Out[1]:

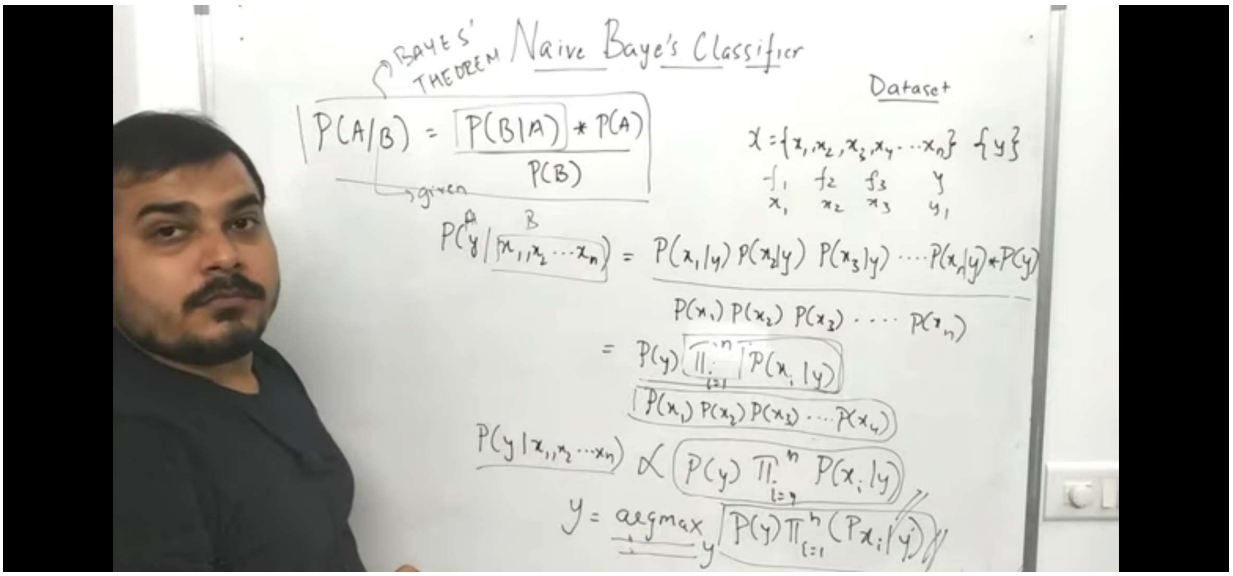


Conditional Probabilty >>> P(A/B) = P(A∩B) / P(B) or P(A|B)= P(B/A) * P(A) / P(B)

- tose two coin. which is example of independetly
- but,if there is three red and two black marble in the box, and queation is what is the probability of picking one black marble having one marble picked up earlier. this is depedent scenario
- the equation says that probability of a, given B.
- in formula - P(A|B) called as posterior probability >> P(B/A) is likelihood prob >> P(B) is marginal prob >> P(A) is prior prob

In [2]:
```
from IPython.display import Image
Image(filename='nb1.jpg')
```

Out[2]:

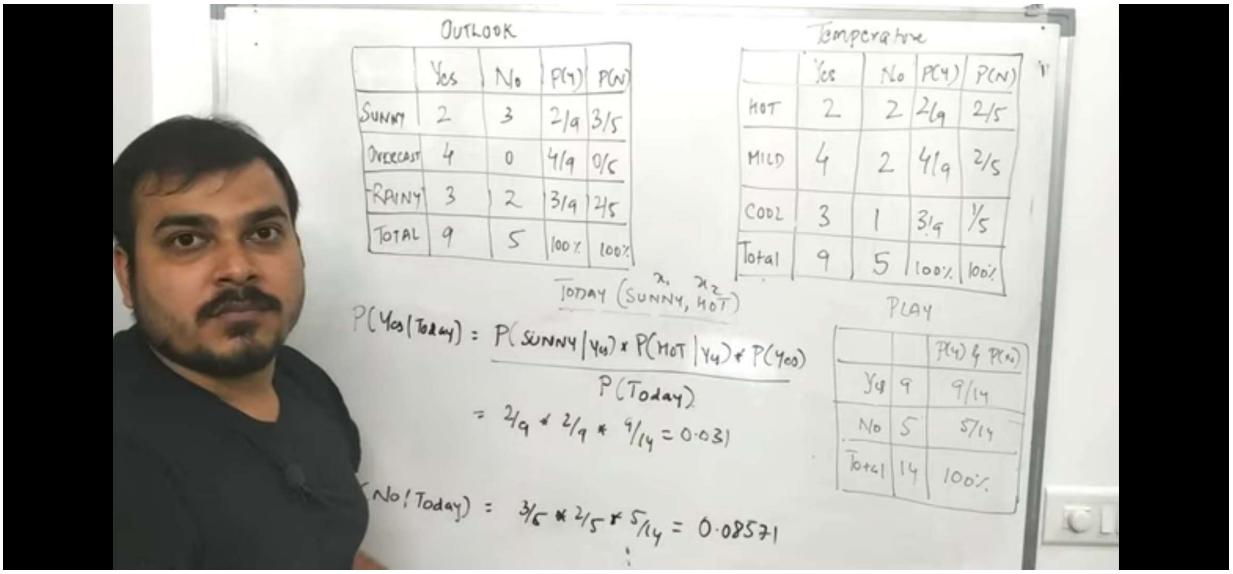


Dataset = x1,x2,x3,x4,x5,x6 and y

- so according to naive bayes theorm- i would saya that given x1,x2,x3,x4,x5,x6 probability of y.

In [3]:
```
from IPython.display import Image
Image(filename='nb3.jpg')
```

Out[3]:



- after getting values for Yes and no we have to normalise it by devide the value with the sumation of both yes and no. so prob of Yes = 0.27 and prob of No = 0.73. so, final output will be No. (argmax).

##3

Decision Tree

Decision Tree Entropy- calculate purity of subsets

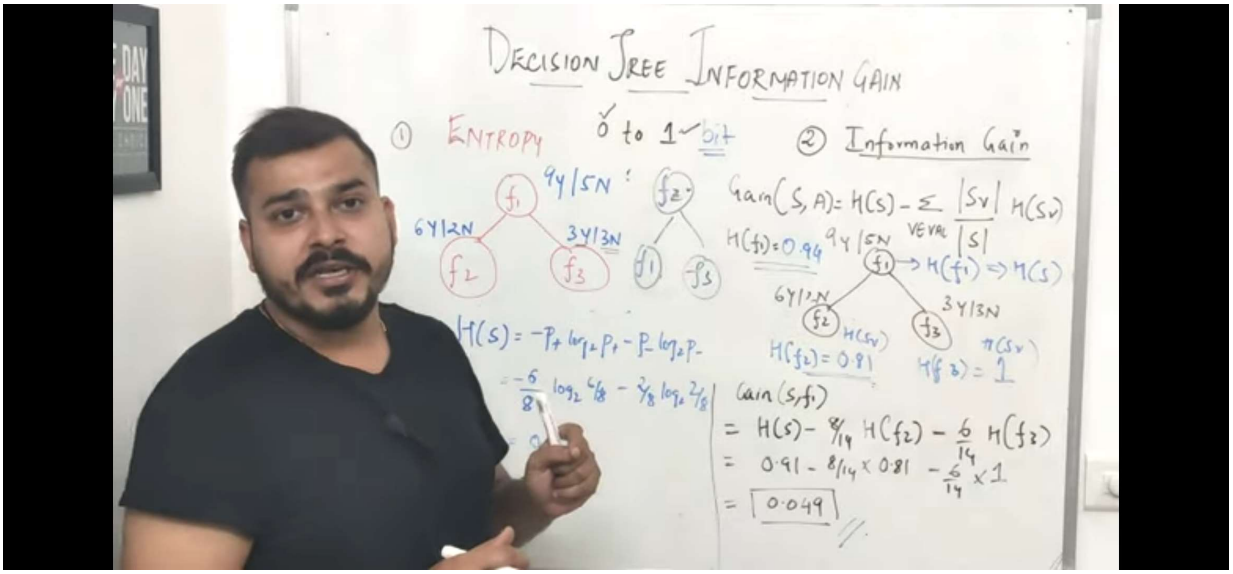
- eg. f1,f2,f3,f4 o/p, now in order to decide which feature can be used to split a tree, entropy technique is used
- Entropy measures the purity of the splits.
- Entropy is calculated by H(s) = - P(+) log(p+) - P(-) log(p-) , where % of positive class =P+ and % of negative class = P(-)
- eg. after using particular feature total record = 3 Yes and 2 No >>> so entropy = - 3/5 * log(3/5) - (2/5) log(2/5) = 0.78(between 0 to 1)
- less entropy is always good.
- tree stop splitting ahead when node has entropy = 0 eg, condition such as (5 Yes and 0 No)

Information Gain

- Gain(S,A) = H(S) - (sumation of) (|Sv| / |S|) * H(Sv) ; H(S) is entropy
- calculate imformation gain according to the image.
- the structure which gives highest information gain, that structure is used as final decsion tree.

In [6]:
```
from IPython.display import Image
Image(filename='ig.jpg')
```

Out[6]:



Gini

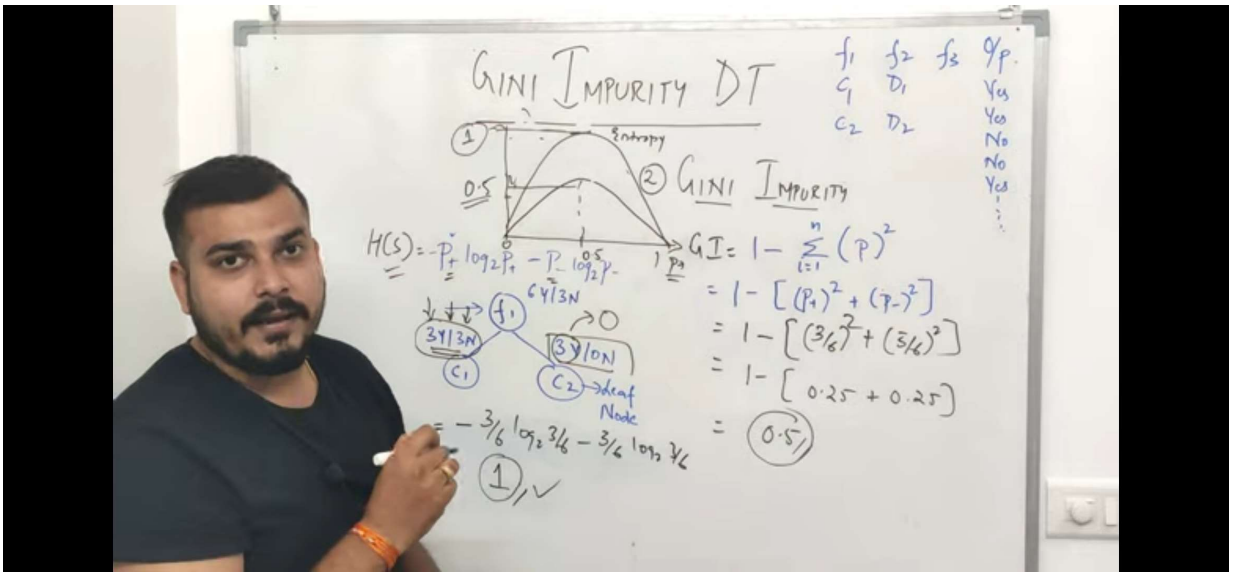
- most of the time gini is better then entropy

> formula: GI = 1- sum (i=1 to n) (p)^2 >>>> =1- [(P+)^2 + (P-)^2]

- Gini Impority is computationally efficient, it takes short period of time for excecution #### Reason
- in the formula of entropy, log is present, and it takes more time to calculate logrithmic calculaiton. As we do not have thpse kind of computation in gini formula.

In [3]:
```
#image
from IPython.display import Image
Image(filename='dtree2.jpg')
```

Out[3]:



In [ ]:

In [ ]:

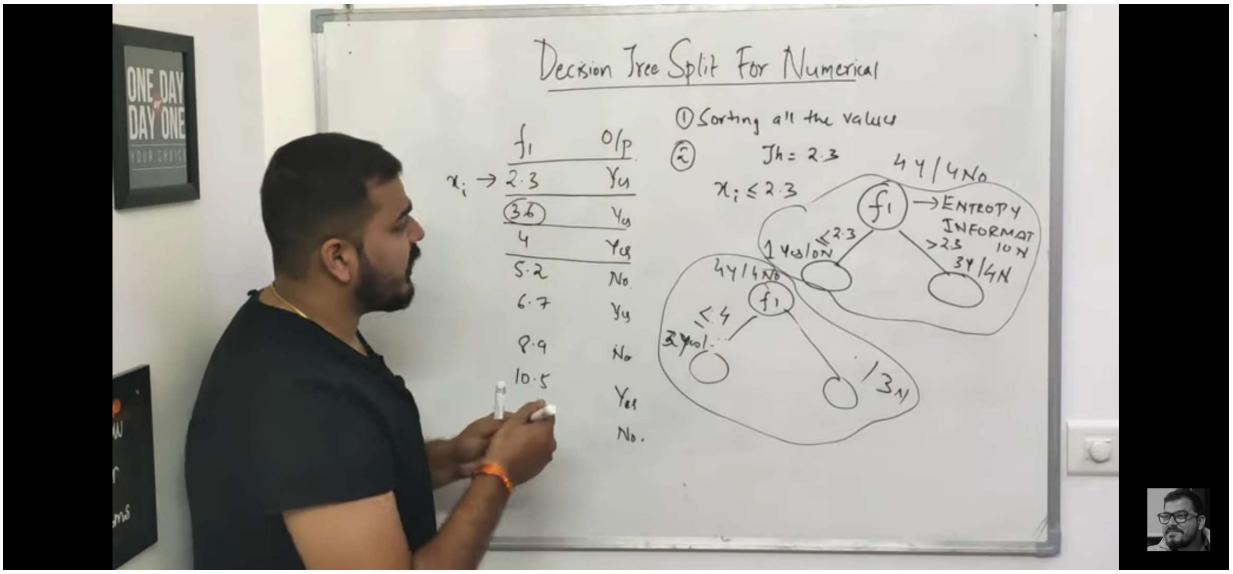
For numerical feature

- sort all the value in first step

- it wil check every record Xi >> if Xi <= thrshold value then it will create a branch.
- according to image it wil create n number of tree and calculate entropy and infomation gain for each.
- now, the tree which having best value of information gain that tree will be taken

In [2]:
```
#image
from IPython.display import Image
Image(filename='dtree3.jpg')
```

Out[2]:



In [ ]:

In [ ]:

In [ ]:

In [ ]:

#

Ridge and Lasso Regression or Regularization Tech

- In some case we get quite good result or very low error on traning data but it fails on test data, which is basically a problem of overfitting in regression.

> in another words, we get quite good best-fit line on training data but, when test data enters it gives high error.

- we use ridge and lasso regression to solve this problem.
- in linear regression we try to minimuze sumation of (y - y^)^2, simiilarly in ridge some terms add which are =

> sumation of (y - y^)^2 + lambda * (slop)^2

- According to graph, we can see the stip growth or high growth that the increasing value of exp by one unit will increase salary very highly. (high slope)
- so when another two term will be added(which is also kanown as penalty), we get some more value, therefore model will see for another line that reduces the value, so it would

create differetn bestfit line with comparatively small slope.

- In this case value of (y - y^)^2 and slope would be the smaller value compare to previous.

What we are doing here

- we are penelizing the features which having higher slops
- eg. in equestion y = mx + c if the value of m is higher then , it tries to penelize using ridge regreesion formula.
- it will try the all possible regression line, for which the value of slope is low, it would select those lines.

One Difference in LASSO (which is in slope)

- in lasso the formula will be

> (y - y^)^2 + lambda * |slop| >>>magnitude of slope or mode

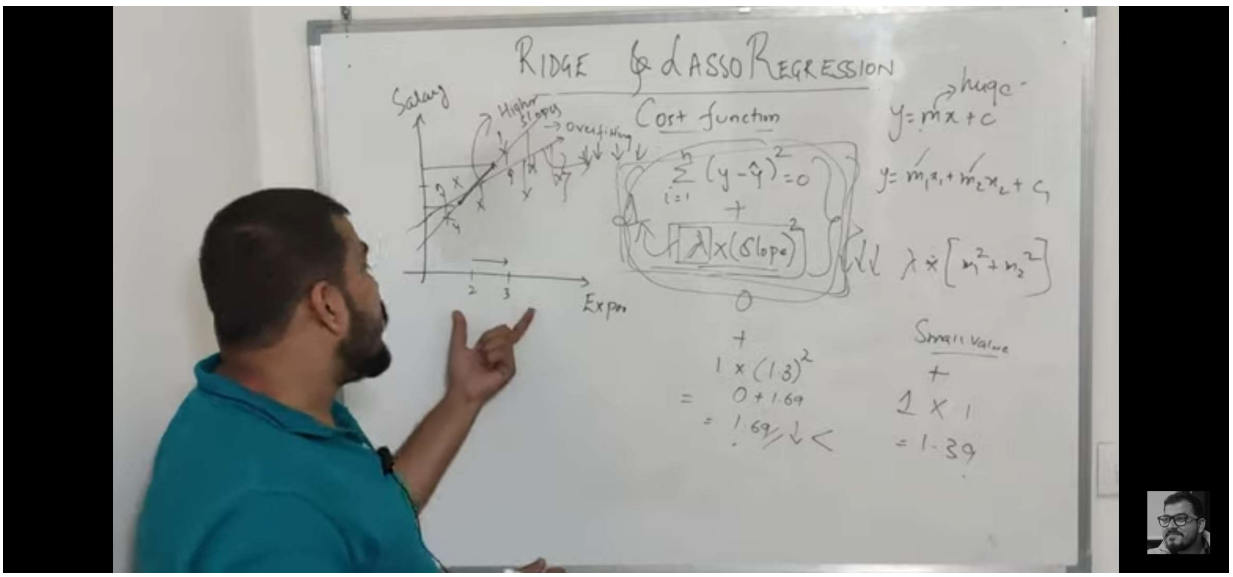
Lasso is significantly used to feature selection also.

Que- why megnituude

- in ridge regression whole slope was moving towards zero, not excatly, but in lasso regression, it move towards zero.
- It remove those feature, which has value of slope = zero. that means these are not imprtant for predicting best-fit line.
- In another words, in lasso regression the regressor line moving toward zero, and that is why some of the feature' slope beacomes zero those are insignificant.

In [1]:
```
#image
from IPython.display import Image
Image(filename='l1l2.jpg')
```

Out[1]:



#

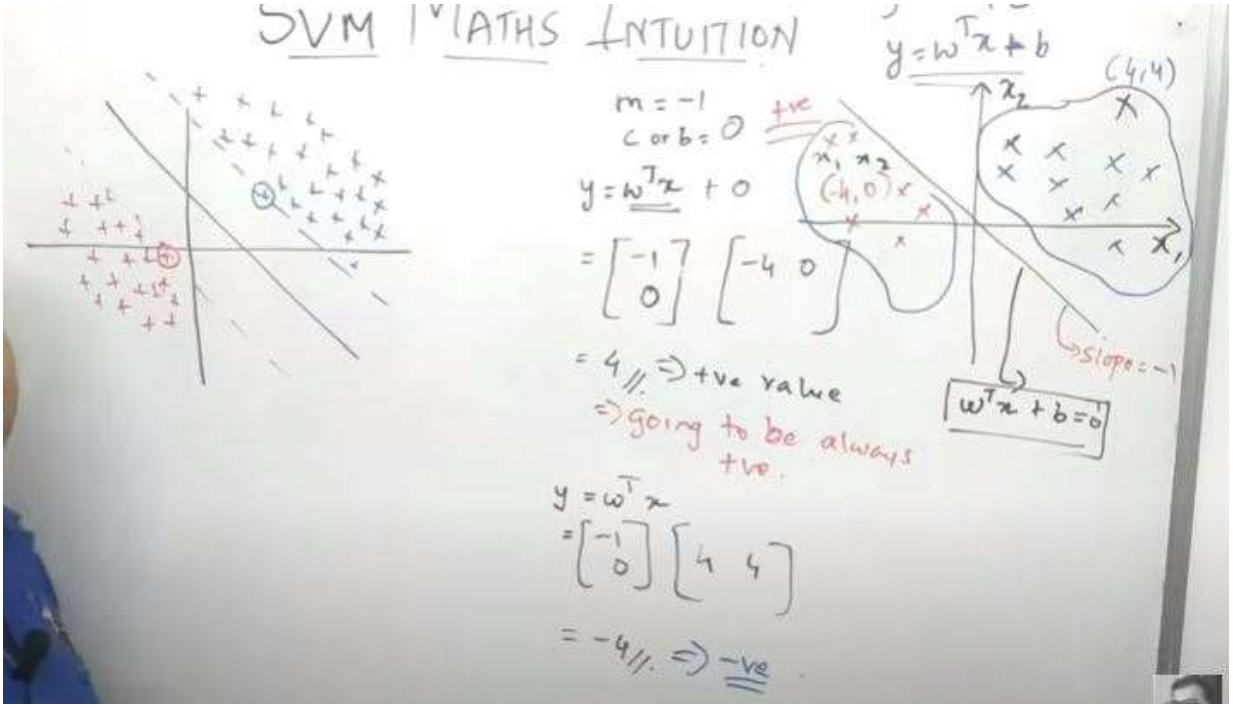
Support Vector Machine

- inner lines known as hyperplane and distance between hyperplane and outer-dotted line is known as marginal distance,

- we plot dotted hyper plane which passes through the nearest negative and positive points. (first image)
- according to image 2, we can create hyper plane like that. our aim is to maximize the marginal plane or distance.

In [6]:
```
#image
from IPython.display import Image
Image(filename='svm1.jpg')
```

Out[6]:



Support vectors

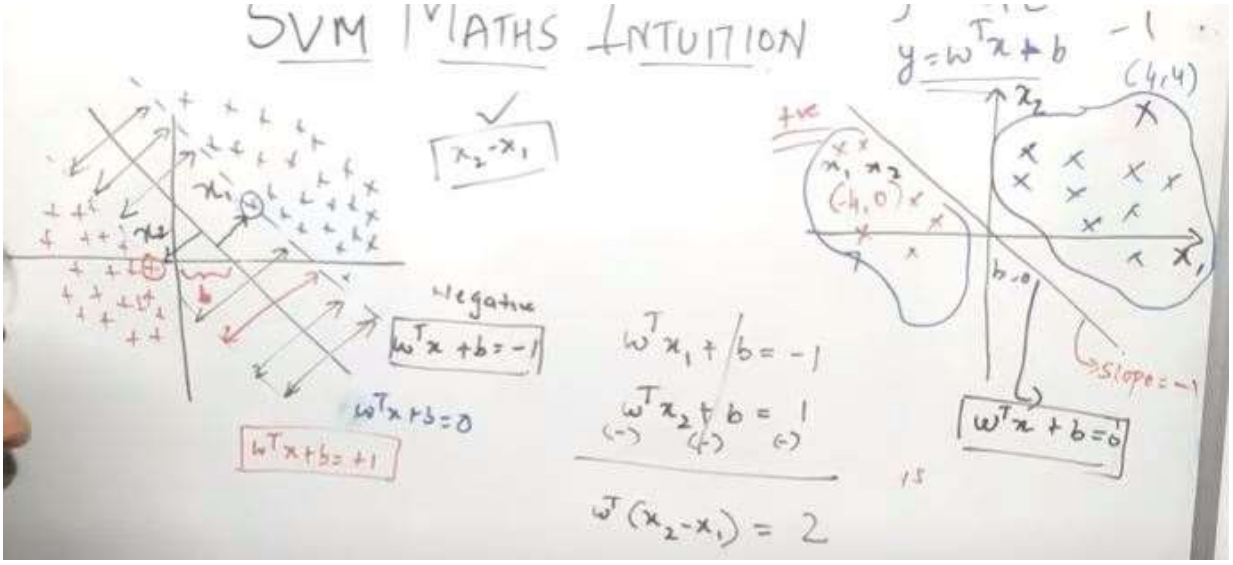
- nearest positive and negative points near to dotted hyper plane is known as support vectors.

What if we have non-linear points

- to solve it uses svm kernels, it converts 2-D to high dimesional
- Then, It creates hyper parameter

In [7]:
```
#image
from IPython.display import Image
Image(filename='svm2.jpg')
```

Out[7]:

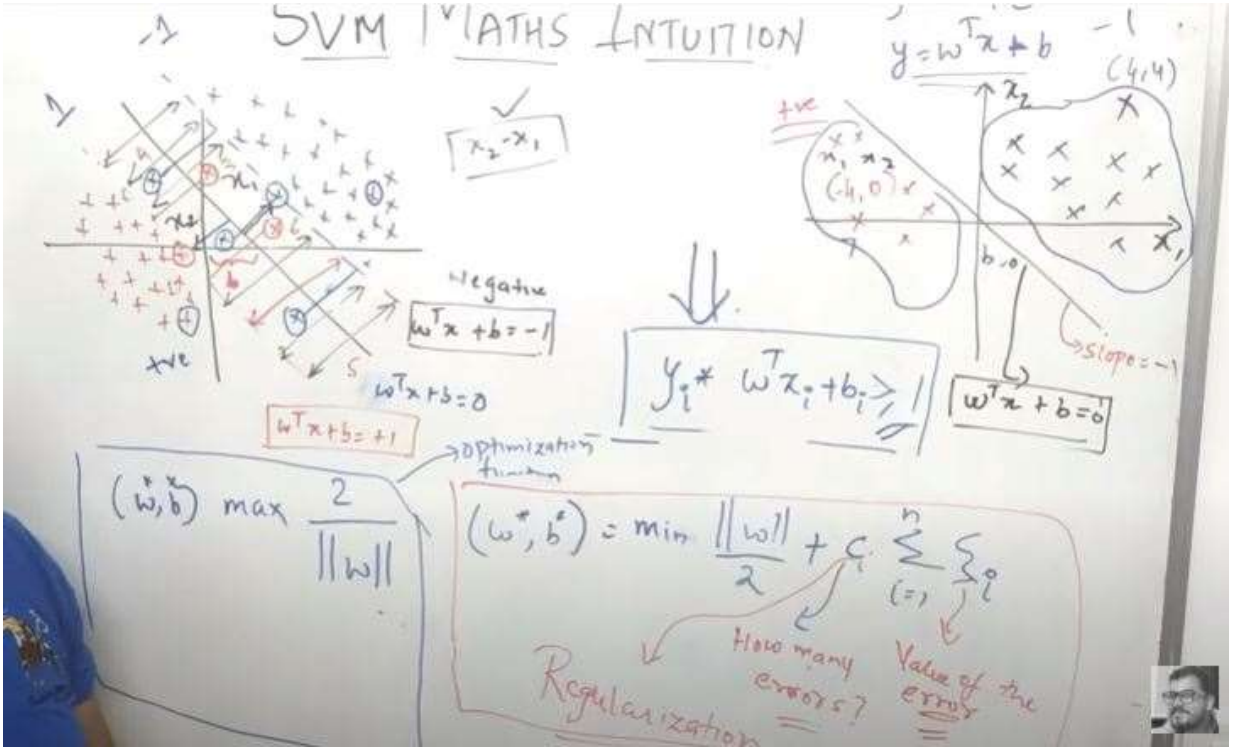


Above image is related to the concept of value of Y respect the slope.

In [ ]:
```
# Below image explains how to count the distance of hyperplane.
- we have to calculate (X2 - X1)
```

In [8]:
```
#image
from IPython.display import Image
Image(filename='svm3.jpg')
```

Out[8]:



we have to consider some errors in order to avoid overfitting.

#

PCA Component Analysis

- Principal Component Analysis is unsupervised machine learning technique, which helps to reduce number of dimentions or number of features to sime lower number.
- Increase number of dimensions, accuracy got impacted.

Steps

1. dimention reduction by standard scaler (Xi - mu) / sigma , where mu = mean, and sigma = std. deviation
2. In PCA, cells whcih are highly correlated, cluster together.

- in PCA, It basically reduces the dimention by finding the simmilar group of records.

#

LDA (linear Discriminant Analysis)

- LDA is like PCA, but it focuses on maximizing the seperatibility among known categories.
- LDA creates a new axis.

Points

1. Create an axis that Maximize the distance between means. (By equestion -) also, minimizing the scatter.
2. Minimize the variation withing each category.

#

In [1]:
```
a="9"
```

In [2]:
```
int(a)
```

Out[2]:
```
9
```

In [3]:
```
a
```

Out[3]:
```
'9'
```

In [8]:
```
mystr = ["python","data"]
```

In [9]:
```
mystr.append("Science")
```

In [12]:
```
len([2,[4,3]])
```

Out[12]:
```
2
```

In [53]:
```
a= [2,22,34,7,10]
```

In [55]:
```
a.sort()
```

In [56]:
```
a
```

Out[56]:
```
[2, 7, 10, 22, 34]
```

In [31]:
```
mylist = [' Python']
```

In [32]:
```
mylist.strip()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
```

```
<ipython-input-32-09b0c1125895> in <module>
----> 1 mylist.strip()

AttributeError: 'list' object has no attribute 'strip'
```

In [ ]: