

Linear Regression (Statistical)

(1) Simple linear Regression

→ The number of independent variables is one.

lets say we have the following data

X	Y
5	0.2
10	0.1
2	0.4
7	0.7

Here we want to predict Y based on values of X.

Thus Y is dependent variable and X is independent variable

linear regression always uses a linear equation

$$Y = \alpha_0 + \alpha_1 X \quad \left. \begin{array}{l} \text{following eq"} \text{ is used for} \\ \text{simple linear regression} \end{array} \right\}$$

↓ ↗
regression coefficients

Target is to find these coefficients such that we get best possible value of Y given X.

* Statistical Method

$$\mathbf{A} = \begin{bmatrix} \sum y^2 & \sum xy \\ \sum yx & \sum x^2 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

As Seen,

$$y = \alpha_0 + \alpha_1 x \quad \text{--- (1)}$$

we can also say

$$\alpha_0 = \bar{y} - \alpha_1 \bar{x} \quad \text{--- (2)}$$

using elimination in (1) & (2)

$$(y - \bar{y}) = \alpha_1 (x - \bar{x})$$

for simplicity

$$y = \alpha_1 n$$

$$n^T y = \alpha_1 n^T n$$

$$(n^T n)^{-1} n^T y = \alpha_1$$

$$[\sum (x - \bar{x})^2]^{-1} \cdot [\sum (x - \bar{x})(y - \bar{y})] = \alpha_1$$

$$\frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2} = \alpha_1$$

equivalent

Now,

$$\alpha_0 = \bar{y} - \alpha_1 \bar{x} \quad \text{from (2)}$$

Note

$$\alpha_1 = \frac{\text{cov}(x, y)}{\text{var}(x) \times \text{var}(y)} \times \begin{cases} \text{Standard deviation}(y) \\ \text{Standard deviation}(x) \end{cases}$$

correlation coefficient

The above formula is equivalent to derived formula

(2) Multiple Linear Regression

→ The number of independent variable is more than 1

Let say we have following data

x_1	x_2	y
3	5	0.1
7	6	3
1	2	0.5
4	9	0.2

The formula for a linear equation

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2$$

* Statistical Method

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 - \textcircled{1}$$

We can also write

$$\alpha_0 = y - \alpha_1 \bar{x}_1 - \alpha_2 \bar{x}_2 - \textcircled{2}$$

using elimination on $\textcircled{1}$ & $\textcircled{2}$

$$(y - \bar{y}) = \alpha_1 (x_1 - \bar{x}_1) + \alpha_2 (x_2 - \bar{x}_2)$$

for simplicity

$$y = \alpha_1 n_1 + \alpha_2 n_2 - \textcircled{3}$$

$$\begin{matrix} & 4 \times 1 & 4 \times 1 \\ y = & \left[\begin{matrix} \uparrow & \uparrow \\ n_1 & n_2 \end{matrix} \right] & - \alpha \end{matrix}$$

$\underbrace{\quad}_{4 \times 1} \quad \underbrace{\quad}_{4 \times 2} \quad \underbrace{\quad}_{2 \times 1}$

It can also be written as

$$\alpha = \left([n_1, n_2]^T [n_1, n_2] \right)^{-1} [n_1, n_2]^T y$$

$$= \begin{bmatrix} \sum n_1^2 & \sum n_1 n_2 \\ \sum n_1 n_2 & \sum n_2^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum n_1 y \\ \sum n_2 y \end{bmatrix}$$

$$\alpha = \frac{1}{\sum n_1^2 \sum n_2^2 - (\sum n_1 n_2)^2} \begin{bmatrix} \sum n_2^2 & -\sum n_1 n_2 \\ -\sum n_1 n_2 & \sum n_1^2 \end{bmatrix} \begin{bmatrix} \sum n_1 y \\ \sum n_2 y \end{bmatrix}$$

$$\therefore \alpha_1 = \frac{\sum n_2^2 \sum n_1 y - \sum n_1 n_2 \sum n_2 y}{\sum n_1^2 \sum n_2^2 - (\sum n_1 n_2)^2}$$

$$\alpha_2 = \frac{\sum n_1^2 \sum n_2 y - \sum n_1 n_2 \sum n_1 y}{\sum n_1^2 \sum n_2^2 - (\sum n_1 n_2)^2}$$

Now,

$$\alpha_0 = \bar{y} - \alpha_1 \bar{x}_1 - \alpha_2 \bar{x}_2 \quad \text{from } ②$$

Note -

$$\bullet \quad \sum n_i^2 = \sum (x_i - \bar{x})^2 = \sum x_i^2 - N \bar{x}^2$$

$$\text{from } ③ \quad \quad \quad = \sum x_i^2 - N \left(\frac{\sum x_i}{N} \right)^2$$

$$= \sum x_i^2 - \frac{(\sum x_i)^2}{N}$$

$$\bullet \sum xy = \sum (x_i - \bar{x}_i)(y_i - \bar{y})$$

from ③

$$\sum xy = \sum x_i y_i - N \bar{x}_i \bar{y}$$

$$= \sum x_i y_i - N \bar{x}_i \frac{\sum y_i}{N} - N \bar{y} \sum x_i + N \bar{x}_i \bar{y}$$

$$= \sum x_i y_i - N \bar{x}_i \bar{y} - N \bar{x}_i \bar{y} + N \bar{x}_i \bar{y}$$

$$= \sum x_i y_i - N \bar{x}_i \bar{y}$$

$$= \sum x_i y_i - N \frac{\sum x_i}{N} \frac{\sum y_i}{N}$$

$$= \sum x_i y_i - \frac{\sum x_i \sum y_i}{N}$$

Linear Regression (Model Based)

$$y = w_1 n + w_0 \quad \text{Linear}$$

$$y = w_2 n^2 + w_1 n + w_0 \quad \text{Quadratic}$$

$$y = w_3 n^3 + w_2 n^2 + w_1 n + w_0 \quad \text{Cubic}$$

Here w 's are equivalent to x 's seen previously.

All of the above equations can be solved using deep learning model based approach.

→ Cost Function

$$L = \frac{1}{2N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Here N is the number of training samples.

→ Gradient

$$\frac{\partial L}{\partial w_0} = \frac{1}{2N} \sum_{i=1}^N \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_0}$$

$$= \frac{1}{2N} \sum_{i=1}^N -2(y_i - \hat{y}_i) \times 1$$

$$= -\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)$$

$$\frac{\partial L}{\partial w_1} = \frac{1}{2N} \sum_{i=1}^N \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_1}$$

$$= \frac{1}{2N} \sum_{i=1}^N -2(y_i - \hat{y}_i) n_i$$

$$= -\frac{1}{N} \sum (y_i - \hat{y}_i) n_i$$

generalizing

$$w_0 \rightarrow -\frac{1}{N} \sum (y_i - \hat{y}_i)$$

$$w_1 \rightarrow -\frac{1}{N} \sum (y_i - \hat{y}_i) n_i$$

$$w_2 \rightarrow -\frac{1}{N} \sum (y_i - \hat{y}_i) n_i^2$$

→ Steps for Linear Regression

- (1) Decide on α , ϵ & stopping criteria
- (2) Make initial guess for w_0 & w_1
- (3) Calculate $w^{(k+1)} = w^{(k)} - \frac{\alpha}{N} \sum (y_i - \hat{y}_i) n_i$

(4) Calculate stopping criterion

if condition satisfied stop

if not go to step 3

Logistic Regression - (Model Based)

The equations represent non-linear function

Quadratic, Cubic, Linear that we saw earlier are all linear functions

To introduce non-linearity we use a sigmoid function

Let's say,

$$z = w_1 x + w_0$$

$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$

→ Cost Function

Binary cross entropy for binary classification

$$L = -\sum_{i=1}^N \left(y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i) \right)$$

N is the number of training samples

→ Gradient

$$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z} \frac{\partial z}{\partial w_0}$$

Now,

$$\frac{\partial L}{\partial \hat{y}_i} = - \left(\frac{y_i}{\hat{y}_i} + \frac{-(1-y_i)}{(1-\hat{y}_i)} \right)$$

$$\frac{\partial L}{\partial w_0} = \left(\frac{y_i - \hat{y}_i}{\hat{y}_i (1 - \hat{y}_i)} - \frac{\hat{y}_i + y_i \hat{y}_i}{\hat{y}_i (1 - \hat{y}_i)} \right)$$

$$= - \left(\frac{y_i - \hat{y}_i}{\hat{y}_i (1 - \hat{y}_i)} \right)$$

$$\bullet \frac{\partial \hat{y}_i}{\partial w_0} = \frac{1}{1 + e^{-2}} = \frac{1}{1 + e^{-2}} \cdot \hat{y}_i (1 - \hat{y}_i)$$

$$\bullet \frac{\partial L}{\partial w_0} = 1$$

$$\text{similarly } \frac{\partial L}{\partial w_1} = n$$

$$\bullet \frac{\partial L}{\partial w_0} = \frac{(y_i - \hat{y}_i)}{\hat{y}_i (1 - \hat{y}_i)}$$

$$\frac{\partial L}{\partial w_0} = - \sum_{i=1}^N (y_i - \hat{y}_i) \cdot 1 \quad \text{for } w_0$$

$$\text{similarly } \frac{\partial L}{\partial w_1} = - \sum_{i=1}^N (y_i - \hat{y}_i) \cdot n_i \quad \text{for } w_1$$

Validation

→ Normal Validation

Divide the dataset into 3 parts

Train , Validation , Test

$\underbrace{\hspace{1cm}}$ $\underbrace{\hspace{1cm}}$

70% 30%

Use the validation set to evaluate the results obtained during training. Validation set can be 5-10 % of the 70% train split.

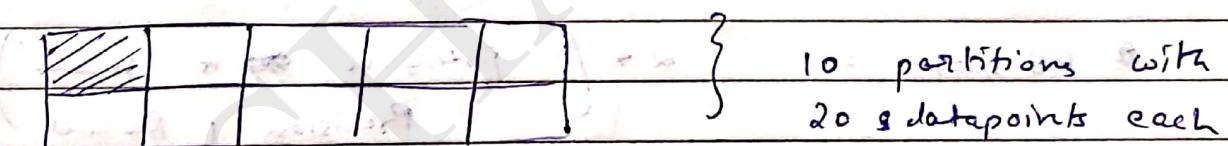
→ K - Fold Cross Validation

Let's say 70% of the train split contains 200 samples

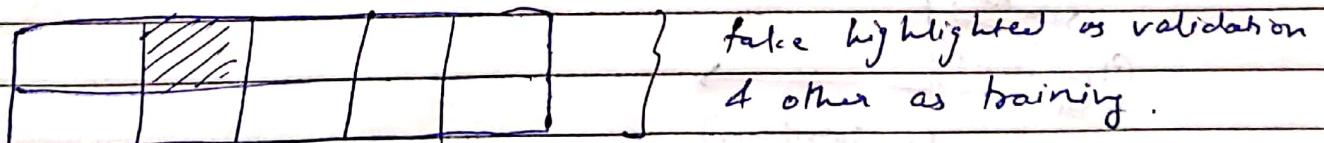
In K-Fold cross validation the data is divided in bins/partitions

e.g. $k = 10$

thus, 10 partitions & 200 data points



Take highlighted as validation & other as training data and obtain accuracy



Similarly for all partitions & average all accuracy to get results.

Evaluating Model Results

→ Confusion Matrix

		Actual		
		True positive	False Positive	predicted has heart disease
predicted	False Negative	True Negative		
	Actually has heart disease	Actually doesn't have heart disease		

The True positive & True Negative are correct predictions & other are incorrect.

		Actual			
		Dog	Cat	Deer	
		Dog	Cat	Deer	
predicted	Dog				
	Cat				
	Deer				

Another example

Here the highlighted region is correct prediction others are incorrect.

→ Some Formula's

Let's say,

$$\text{True Positive} = TP$$

$$\text{True Negative} = TN$$

$$\text{False Positive} = FP$$

$$\text{False Negative} = FN$$

- Sensitivity (Recall)

$$= \frac{TP}{TP + FN}$$

- Specificity = $\frac{TN}{TN + FP}$

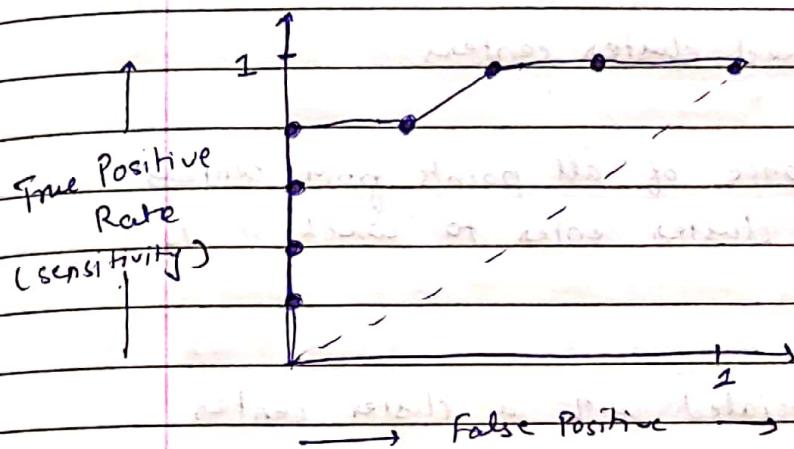
- Precision = $\frac{TP}{TP + FP}$

- F1 Score = $2 * \left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \right)$

- Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$

→ ROC & AUC

• A ROC (Received Operator Characteristic)



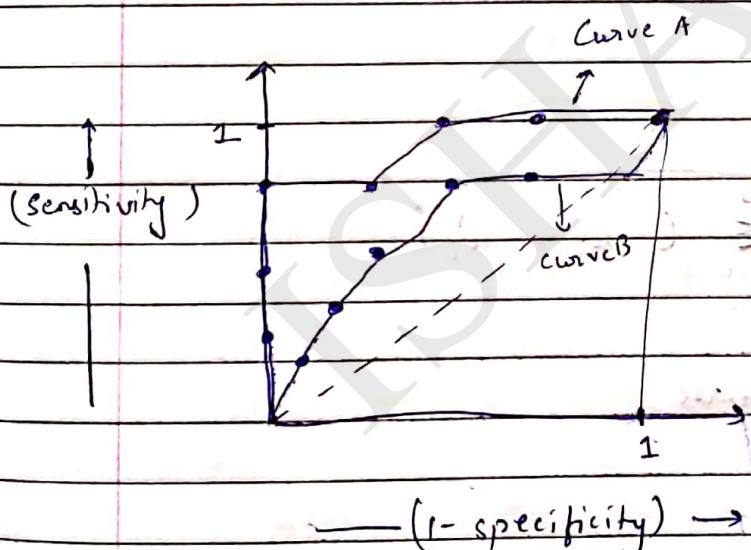
Here we want the false positive rate

to be minimum &

true positive rate to be maximum

All the dots were plotted in the figure are results generated from different models

• AUC (Area Under Curve)



Here in the figure the area under the curve A is greater than the area under curve B and thus the

area under curve A is better

Note → The models of each curve are made using a specific ML technique. eg - (Random Forest, SVM, Deep learning model, regression etc.)

WEEK 9

Machine Learning - K-Nearest Neighbour

* KNN (K-Nearest Neighbours Classification)

Supervised learning is used for

multi-classification problem

Split the Data

(1) ~~to split test & validation~~
 train, test, validation } Now, validation is of two types
 ↓
 70% 10% 20%

(1) Normal validation

(2) k-fold cross validation

lets say

	A	B	C	D	Target	training dataset
1	1	1	2	0	0	row 1 and 2
2	2	2	2	0	0	row 3

3 rows 3 features 1 target

features

A	B	C	target	test dataset
1	2	3	1	0

Euclidean distance (with training in front with

from training data
target class

$$1 \Rightarrow \sqrt{(2-1)^2 + (3-1)^2 + (1-2)^2} \Rightarrow \sqrt{6}, 0$$

$$2 \Rightarrow \sqrt{(2-2)^2 + (3-1)^2 + (1-2)^2} \Rightarrow \sqrt{5}, 0$$

$$3 \Rightarrow \sqrt{(2-3)^2 + (3-3)^2 + (1-1)^2} \Rightarrow \sqrt{1}, 1$$

Arrange distances in sorted order

$$\sqrt{1}, \sqrt{5}, \sqrt{6}$$

→ Now, if $k = 1$, then answer

it means 1 Nearest-neighbour

Predicted

∴ Target class of test data is (1)

→ if $k = 2$

it means 2 N-N

∴ There is a tie one neighbour has class 0
& other has class one

To deal with this increase or decrease value of k
till answer is obtained

→ if $k = 3$

it means 3 N-N

∴ Predicted target class is (0)

since majority is 0 (two NN have 0 & one NN has 1 as

one NN has 1 as
target class = 1

→ instead of testing directly, validation is used for better accuracy

* K-Means Clustering (unsupervised learning)

$K=2 \rightarrow$ no. of clusters you want in the given data

- Choose two randomly assigned cluster centers

- Now calculate euclidean distance of all points from centers and assign point to center cluster center to which it is closest

- Every datapoint is now associated with a cluster center so now calculate mean of all points in a cluster and place centers at the mean

Loop until

cluster centers don't change significantly

→ Formula's

$$\text{distance of point } x_i \text{ from cluster center } n^* = \sqrt{\sum_{i=1}^N (x_i - n_i^*)^2}$$

N = no. of features

x_i = datapoint

n^* = center

$$\text{center of cluster } n^* = \frac{1}{N} \sum_{i=1}^N x_i$$

$N =$ no. of datapoints in the cluster

$x_i =$ individual datapoint of the cluster

- Threshold } = if all (distance of old center < thresh)
from new center

* DBSCAN (Unsupervised learning)

- Core point → If for any datapoint if that datapoint is surrounded by X or more datapoints with the distance being less than d then that point is called a core point. (X & d can be decided based on performance on dataset)
- Non-Core point → If not a core point but is at a distance less than d from a core point (Boundary point)
- Noise point → Neither a core point nor boundary point

Steps

- (1) Calculate all core points, boundary points and noise points
- (2) Pick a random core point and name it as cluster 1 now merge all the core points at a distance less than d from the current point. continue until there are no more core points to merge to cluster 1

(2) Out of remaining core points do the same and form clusters (not segregated land, due to step 1)

(2)

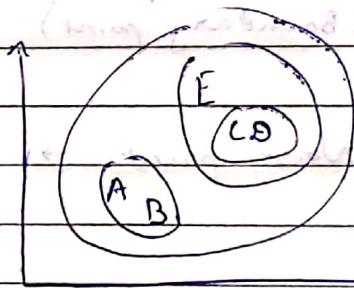
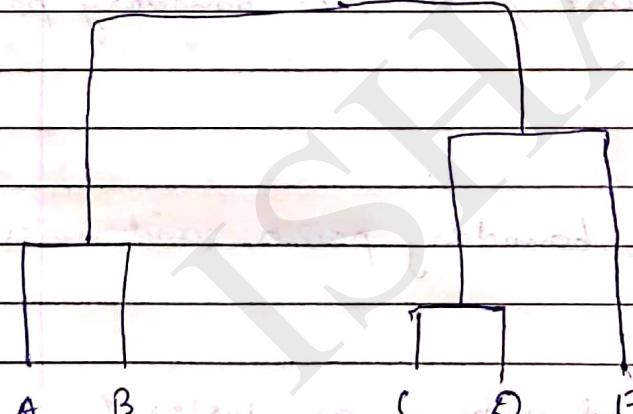
(3)

(3) Now add all the boundary points that can be added to cluster 1. Note that boundary points can't extend the cluster like core points

(4) Now no other point can be added to cluster 1. Out of the remaining core points choose 1 point and name it as cluster 2 and carry out steps (2) & (3) again

* Hierarchical Clustering (unsupervised learning)

→ Agglomerative



The above figure is called Dendrogram

Here,

- out of all the points we find two points among all such that the distance separating them is minimum
in this case those points are C & D

Now we make $[C, D]$ as one cluster

- Now considering $[C, D]$ as a point we find the two points such that distance separating them is minimum

But how do we calculate distance of a point from the point $[C, D]$?

→ There are 3 ways :

(distance below means Euclidean distance)

(1) Single Linkage

$$\text{distance of point } A \text{ from point } [C, D] = \min \left(\begin{array}{l} \text{distance of point } A \text{ from point } C \\ \text{distance of point } A \text{ from point } D \end{array} \right)$$

(2) Complete linkage

$$\text{distance of point } A \text{ from point } [C, D] = \max \left(\begin{array}{l} \text{distance of point } A \text{ from point } C \\ \text{distance of point } A \text{ from point } D \end{array} \right)$$

(3) Average linkage

$$\text{distance of point } A \text{ from point } [C, D] = \arg \left(\begin{array}{l} \text{distance of point } A \text{ from point } C \\ \text{distance of point } A \text{ from point } D \end{array} \right)$$

Naive Bayes Classification

- Single Event Probability - $P(X), P(Y)$
- Joint Event Probability - $P(X, Y)$
- Conditional Probability - $P(X|Y), P(Y|X)$

Relation between Joint & Conditional probabilities

$$P(X, Y) = P(Y|X) * P(X) = P(X|Y) * P(Y)$$

from above equation

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)} \quad \left. \begin{array}{l} \text{Bayes} \\ \text{Theorem} \end{array} \right\}$$

$$\text{posterior} = \frac{\text{likelihood} * \text{prior}}{\text{Evidence}}$$

* Now, let's say Target class has 2 values C_1 & C_2

Expressing in terms of probability

$$P(C_1|X) = \frac{P(X|C_1) * P(C_1)}{P(X)}$$

↓
 target class
 values attributes

We need to compare $P(C_1|X)$ & $P(C_2|X)$ to get the best match & as per above eqⁿ denominator for C_1 & C_2 is same and thus it can be ignored

$$\therefore p(c|x) = p(x|c) * p(c)$$

Now, ideally $p(x|c) = \frac{p(x_1, x_2, \dots, x_n | c)}{p(x_1, x_2, \dots, x_n)}$

$$p(c|x_1, x_2, x_3, \dots, x_n) = \left(\frac{p(x_1|x, \dots, x_n, c)}{p(x_2|x, \dots, x_n, c)} * \right.$$

$$(x)_q + (x_1)_q = (x)_q + (x_1)_q * p(x_3|x, \dots, x_n, c) *$$

$$\text{repeat } \left(\frac{(x)_q + (x_1)_q}{(x)_q} = (x)_q * p(x_n|x, \dots, x_{n-1}, c) \right)$$

now \rightarrow By chain rule

But we assume that x_1, x_2, \dots, x_n are independent
(thus method is called "naive bayes")

So,

$$p(x_1|x_2, \dots, x_n, c) = p(x_1|c)$$

and thus the above equation becomes as follows

$$p(c|x_1, x_2, \dots, x_n) = (p(x_1|c) * p(x_2|c) * p(x_3|c) * \dots * p(x_n|c))$$

Calculate $P(C_1 | X)$, $P(C_2 | X)$... $P(C_n | X)$

Then calculate maximum out of them and assign the data entry to respected class

$$\underset{k \in \{1, 2, \dots, k\}}{\operatorname{argmax}} P(C_k) = \prod_{i=1}^n P(x_i | C_k) = ①$$

→ Log Trick (apply log in ①)

$$\log(P(C_k)) \stackrel{?}{=} \sum_{i=1}^n \log(P(x_i | C_k))$$

* Laplace Smoothing

Here $P(X_1 | C)$, $P(X_2 | C)$... $P(X_n | C)$ any of them can be equal to zero. In this case overall probability will be zero and inappropriate answer is obtained.

$$\text{Let's say, } P(X_1 | C) = \frac{0}{9}, \quad P(X_2 | C) = \frac{5}{9}, \quad P(X_3 | C) = \frac{4}{9}$$

After Laplace smoothing

$$P(X_1 | C) = \frac{0+1}{9+3}, \quad P(X_2 | C) = \frac{5+1}{9+3}, \quad P(X_3 | C) = \frac{4+1}{9+3}$$

$$(because 3 is used)$$

$$(x_1, x_2, x_3 \text{ are three})$$

$$P(X_1 | C) = \frac{1}{12}, \quad P(X_2 | C) = \frac{6}{12}, \quad P(X_3 | C) = \frac{5}{12}$$

zero probability is removed

Types of Naive Bayes Classifiers

(1) Bernoulli Naive Bayes

- Used when all the features are distributed by bernoulli distribution (ie - all features contain only 0's & 1's as values)

$$P(\text{Success}) = p$$

$$P(\text{Failure}) = q = 1-p$$

$X = 1$ for success and $X = 0$ for failure

Then X has a bernoulli's distribution

Now,

$$P(X=n) = (p^n (1-p)^{1-n})$$

(2) Multinomial Naive Bayes

- used when all the feature are discrete count of some entity. Also count is finite

$$P(X_1 = n_1, X_2 = n_2, \dots, X_k = n_k) = \frac{\left(\sum_{i=1}^k n_i \right)!}{\prod_{i=1}^k n_i!}$$

eg = blood group	A	O	B
probability	0.44	0.55	0.01

lets say there are 6 people

2 → A

3 → O

1 → B

$$\left(\prod_{i=1}^k p_i^{n_i} \right)$$

$$P(X_1 = 2, X_2 = 3, X_3 = 1) = \frac{6!}{2! 3! 1!} p_1^2 p_2^3 p_3$$

$$(0.44)^2 (0.55)^3 (0.01)^1$$

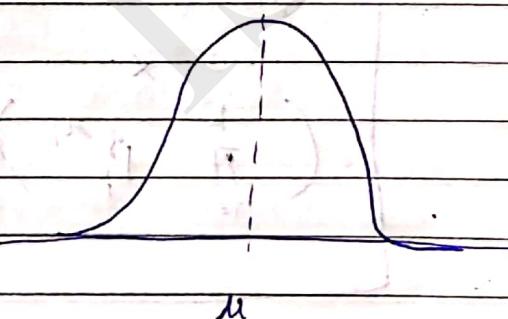
(3) Gaussian Naive Bayes

- used when all the features are continuous in their distribution and have a normal distribution. Also the count can range from $-\infty$ to ∞ .

$$P(X_1 = n_1, X_2 = n_2, \dots, X_m = n_m) = \prod_{i=1}^m g(n_i; \mu_i, \sigma_i)$$

Here, $g(n_i; \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2\sigma_i^2}(n_i - \mu_i)^2}$

$$-\infty < n_i < \infty$$



Decision Trees

→ Root Node

The topmost node of the tree is called the root node.

It is also the first node in (the) tree when we go from top to bottom. Trees have only one root node.

→ Leaf Node

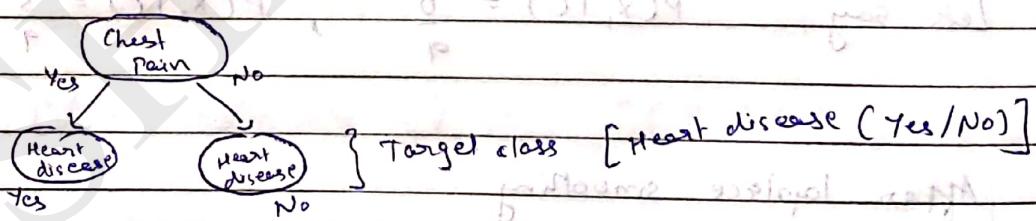
The bottom most nodes of a tree are leaf nodes.

These nodes don't have child nodes. Tree can have multiple leaf nodes.

* Decision Trees can be divided into two types

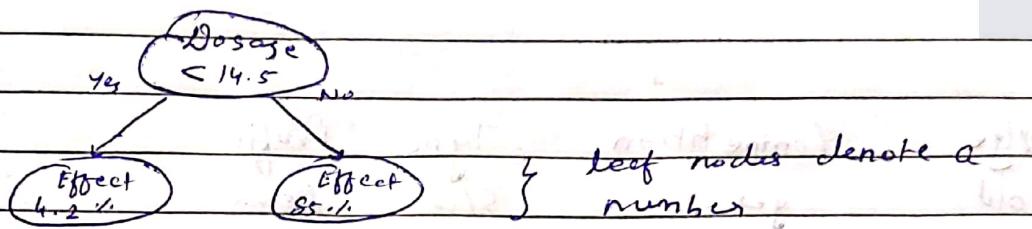
→ Classification Trees (used to solve classification tasks)

leaf nodes of these trees are associated with target class



→ Regression Trees (used to solve regression tasks)

leaf nodes of these trees are associated with a number



* Classification Trees

There are two main ways by which we can construct classification trees from given data which would have multiple attributes and a target class.

(1) Gini Impurity

(2) Entropy

$$\begin{aligned}
 & \left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 = \left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 = \frac{1}{9} + \frac{4}{9} = \frac{5}{9} \\
 & \left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 = \left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 = \frac{1}{9} + \frac{4}{9} = \frac{5}{9}
 \end{aligned}$$

$$(n_1^2 + n_2^2) / n^2$$

Entropy

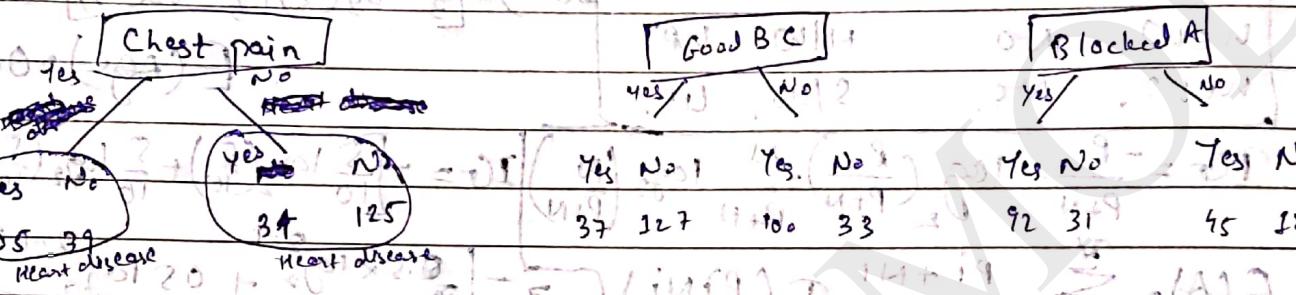
Information

Information gain

DECISION TREES

GINI IMPURITY

Chest Pain	Good Blood	Blocked Arteries	Heart Disease	208	610
No	No	No	No	304	610
Yes	Yes	Yes	Yes	208	304
Yes	Yes	No	No	208	304
etc	etc	etc	etc	304	304



Impurity

Gini Impurity Formulas:

$$\rightarrow 1 - \sum_{j=1}^{\text{No. of nodes}} p_j^2 \Rightarrow \text{Gini Impurity}$$

$$\rightarrow \sum_{j=1}^{\text{No. of nodes}} \left(\frac{p_j + n_j}{p_j + n} \right) \times \text{Gini Index Impurity} \quad \text{total impurity count}$$

Now, for chest pain case

$$\text{gini index for left node} = 1 - \left(\frac{105}{105+39} \right)^2 - \left(\frac{39}{105+39} \right)^2 = 0.395$$

$$\text{gini index for right node} = 1 - \left(\frac{34}{34+125} \right)^2 - \left(\frac{125}{34+125} \right)^2 = 0.336$$

Total impurity count

$$= \left(\frac{144}{144 + 159} \right) 0.395 + \left(\frac{159}{144 + 159} \right) 0.336$$

$$= 0.364$$

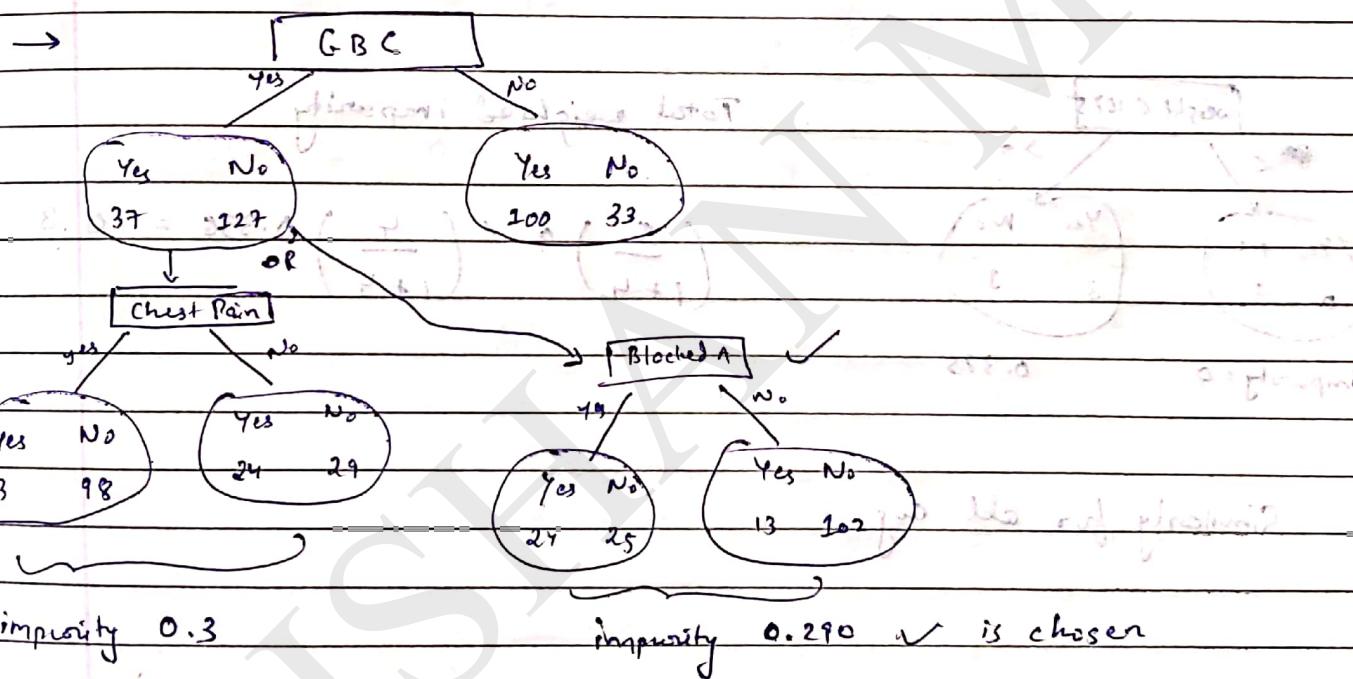
Similarly for others

$$GBC = 0.360$$

$$BA = 0.381$$

minimum out of 3 is GBC

GBC is root node



Note - (1) Calculate all of gini impurity scores

(2) if node itself has lowest score and child node have high value then don't separate and make it parent

(3) If separate results in improvement pick lowest valued node

→ Dealing with numeric values

↓ max. disease label

Weight & Heart Disease

155

No + Yes

Avg 167.5

0.3

180

Yes

P22.5

Avg 185

0.47

190

No

minimum weight of division 2

Avg 205

0.27

→ split using < 205

220

Yes

0.280 - 0.283

Avg 222.5

0.428

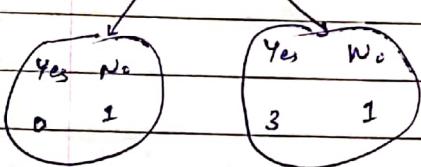
0.280 - 0.283

225

Yes

weight < 167.5

Total weighted impurity



$$\left(\frac{1}{1+4}\right)0 + \left(\frac{4}{1+4}\right)0.33 = 0.3$$

impurity = 0

0.375

Similarly for all avg

E.0 disease

lowest weighted impurity for no splits (0) = only

when weight > 205 then 0.280 < 0.283 then 0.283

lowest weighted impurity for first split

and lower weight then 0.280 < 0.283 then 0.283

Decision Trees

ENTROPY

Page No.

Date



Age	Competition	Type	Profit
old	yes	S/W	Down
old	No	S/W	Down
old	No	H/W	Down
mid	Yes	S/W	Down
mid	Yes	H/W	Down
mid	No	H/W	Up
mid	No	S/W	Up
new	Yes	S/W	Up
new	No	H/W	Up
new	No	S/W	Up

→ Entropy Formula's

$$\cdot \text{IG} = - \left(\left(\frac{P}{P+N} \right) \log_2 \left(\frac{P}{P+N} \right) + \left(\frac{N}{P+N} \right) \log_2 \left(\frac{N}{P+N} \right) \right)$$

$$\cdot E(A) = \sum_{i=1}^{\text{no. of Nodes}} \left(\frac{P_i + N_i}{P+N} \right) I(P_i, N_i)$$

$$\cdot \text{Gain} = \text{IG} - E(A)$$

Note $\log_2 n = \frac{\log_{10} n}{\log_{10} 2}$

→ Now, solving the above example

Age information has three classes (old, mid, new) and two states (Down, Up)

	Down	Up
old	3	0
Mid	2	2
New	0	3

$$I(\text{old}) = - \left(\frac{3}{3} \log_2 \left(\frac{3}{3} \right) + \frac{0}{3} \log_2 \left(\frac{0}{3} \right) \right) = 0$$

$$I(\text{mid}) = - \left(\frac{2}{4} \log_2 \left(\frac{2}{4} \right) + \frac{2}{4} \log_2 \left(\frac{2}{2} \right) \right) = 1$$

$$I(\text{new}) = - \left(\frac{0}{3} \log_2 \left(\frac{0}{3} \right) + \frac{3}{3} \log_2 \left(\frac{3}{3} \right) \right) = 0$$

$$E(\text{Age}) = \left[0 \times \frac{3}{10} + 1 \times \frac{4}{10} + 0 \times \frac{3}{10} \right] = 0.4$$

$$IG = - \left(\frac{5}{10} \log_2 \left(\frac{5}{10} \right) + \frac{5}{10} \log_2 \left(\frac{5}{10} \right) \right)$$

$$= - (0.5 \log_2 2^{-1} + 0.5 \log_2 2^{-1})$$

$$= - (-0.5 - 0.5)$$

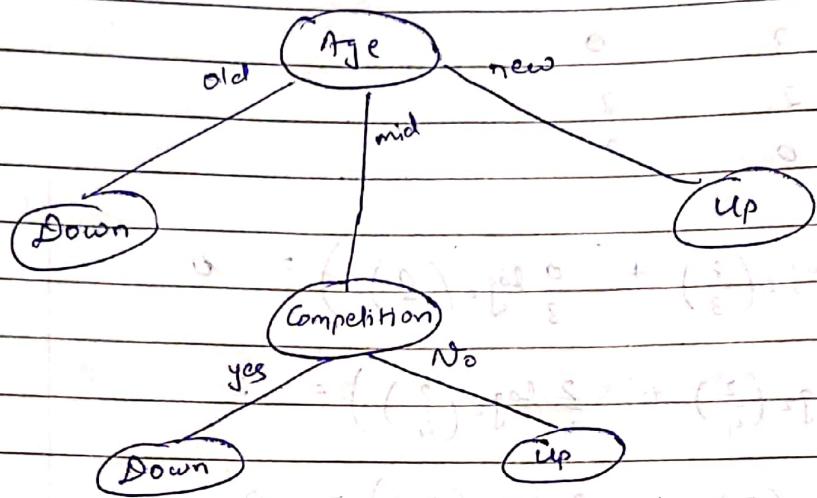
$$= - (-1) = 1$$

$$\text{Gain(Age)} = 1 - 0.4 = 0.6$$

$$\text{Similarly } \text{Gain(Competition)} = 0.124$$

$$\text{Gain(Type)} = 0$$

We split value/attribute with a maximum gain



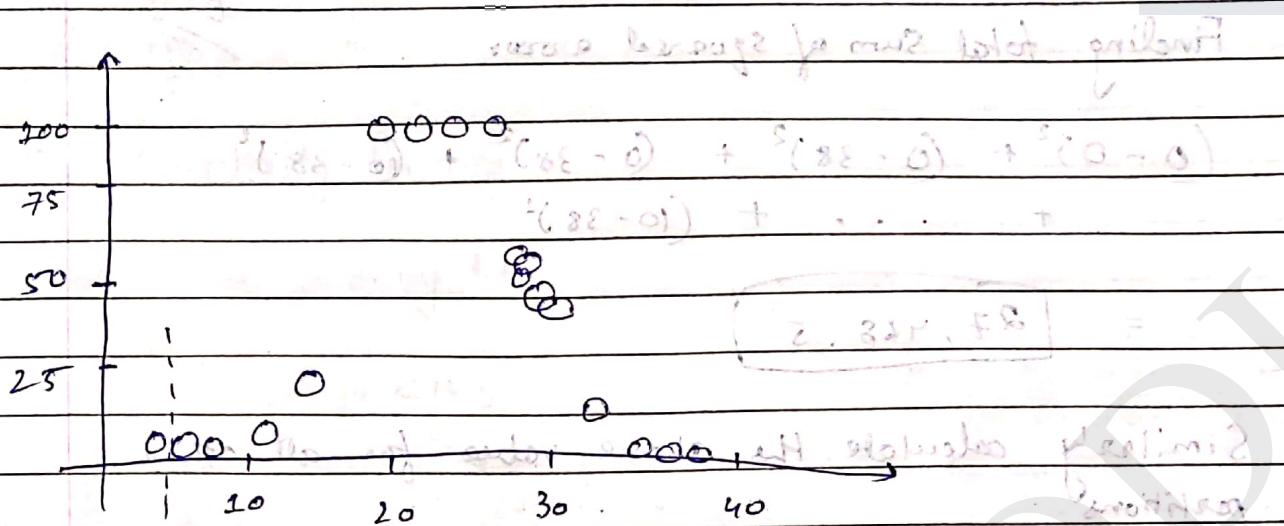
* Regression Trees

In the following example we have used drug dosage and its corresponding effect. We want to predict effect of a drug given its dosage.

$$D.O = P.O + D \cdot \text{Effect}(D)$$

Effect = (weightage) and adjustment

$D \cdot \text{Effect} = D \cdot \frac{\text{Effect}}{D}$



Drug dosage (mg)

This is below minimum threshold at 20 mg

(lowest drug to pos. \rightarrow zero) corresponds to first

Dosage	Effect	Now,	Serial no. of the
1 — 5	0	from partition	points
2 — 6	0	1, 2 & 3 ... n	numbered if
3 — 7	0	:	:
11	10	10, 11, 12 ... 20	separation
15	22	1 ... n-1 & n	
:	repeating in two or three of students. Like		
35	50		
36	100	100 of two-thirds missing (the two points left)	
etc	etc	100 of two-thirds missing of which are partitioned	
etc	etc	Here separation is dosage $< \frac{s+6}{2}$	

For 1st partition dosage minimum limit for two

1 and 2 students. In students w/

$$\text{avg of effect} = \frac{0+0}{2} = 0 \quad \text{and} \quad \text{avg of effect} = \frac{10+0+10+22+\dots+10}{100} = 6.88 \text{ mg}$$

$$\text{lets say} = 38.8$$

Now, we find sum of squared error

$$(0-0)^2 + (0-38.8)^2 + (0-38.8)^2 + (0-38.8)^2 + \dots + (0-38.8)^2$$

1 2 ... n

Similarly we calculate above value for all partitions

Then choose the partition with minimum error as root of the tree, and split the tree ($\text{dosage} < \text{avg of points perted}$)

eg dosage

1	-	5	if the partition is as
2	-	6	follows,
3	-	7	$(1 \ 4 \ 2 \dots n)$
:	:		Then the split of tree will
n	-	etc	be $(\text{dosage} < 6)$

Till now, we have considered only one column
let's say we have following columns

Dosage Age Sex Effect

Find the minimum squared error for each attribute
(Note that for sex there will be one partition only with male & female as classes)

Out of these minimum square error of each attribute, the attribute with lowest (minimum squared error) is selected as root.

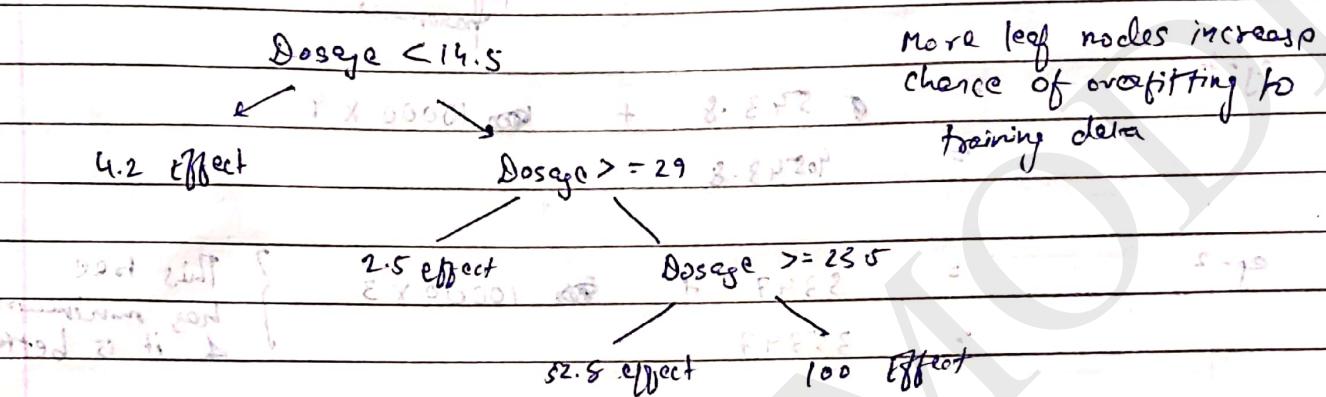
→ If we splitting the tree into sub nodes, we can end up overfitting on the training data

To avoid overfitting we use a technique called Pruning

Pruning FPT : avoids overfitting minimizes total generalization

→ Cost Complexity Pruning

Consider the following tree



$$\text{Sum of squared error for } X < 14.5 = (0 - 4.2)^2 + (0 - 4.2)^2 = 32.0$$

view various branches on minimal splitting + 1.00 + with (all points with dosage < 14.5 [effect] - avg effect)

$$\text{" " " " " " " } >= 29 = (n - 2.5)^2 + \dots$$

expanded to 75 with 100 to

$$\text{" " " " " " " } >= 23.5 = 148.8$$

$$\text{" " " " " " " } < 23.5 = 0$$

$$\rightarrow \text{Thus total} = 320 + 75 + 148.8 + 0 = 543.8$$

Now, $\text{Dosage} < 14.5$

4.2 effect Dosage >= 29

2.5 effect 74% effect

is more generalized

calculating total minimum squared error = 5347

$$\text{Formula} = \text{Sum of squared error} + \alpha T$$

$T = \log_2 (\text{no. of leaf nodes})$

Hyper parameter

$$543.8 + 10000 \times 4$$

$$40543.8$$

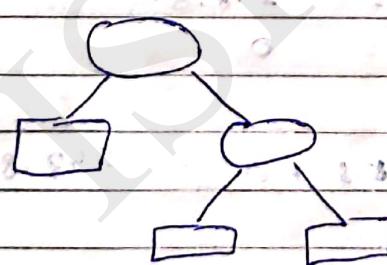
$$\begin{aligned} \text{eg-1} \\ \text{eg-2} &= 5347 + 10000 \times 3 \\ &= 35347 \end{aligned}$$

This tree has minimum & it is better

→ α can be determined for by k-fold cross validation on multiple training set & finding minimum squared error using testing set

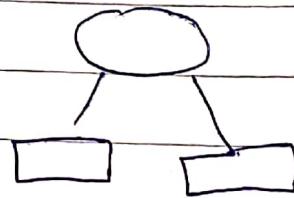
- ① Take the entire data, make a tree. Now find α values at which the structure of tree changes

eg-



at $\alpha = 0$, the above tree had 525 (sum of squared error) which uses minimum when compared to all other combinations

at $\alpha = 10000$



had sum of squared error of 20560 which was minimum among all the combinations of trees.

at $\alpha = 15000$



had sum of squared error which was minimum among all combinations

- ② We have α values now; take a train-test split & create trees from train such that squared error is minimum for a given value of α .

Calculate sum of squared error of the test set on all the tree. The tree with minimum error is the desired tree.

Do this 10 times, since 10-fold cross-validation. The tree with most occurrences as the desired tree is the required tree.

* Random Forest

Steps

- ① Create a bootstrapped dataset
- ② Create a decision tree for bootstrapped dataset
- ③ Evaluate all the trees on test / out of bag data and
- ④ Change parameters & go to step 1

→ Creating Bootstrapped Dataset

- Randomly select rows from the original dataset and create a new bootstrapped dataset.

Note

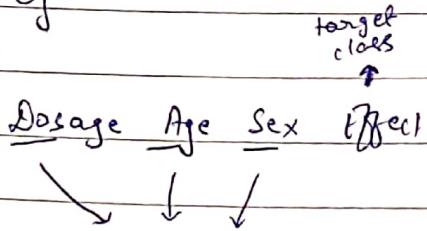
Repetition is allowed (same row can be added to the dataset)
length of the bootstrapped dataset should be equal to original dataset

- Generally all the bootstrapped datasets aggregated span over $\approx 2/3$ of the original data, remaining $1/3$ data is called out of bag data.

→ Creating Decision Tree from ~~Bad~~ Bootstrapped Dataset

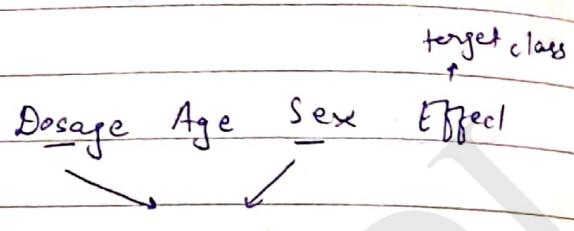
- Unlike normal decision tree where you consider all the available attributes to determine the split of a node, here we randomly pick a set number of attributes

eg -



use all these attributes to split the node

Continue same process excluding the attribute used for split



randomly select a fixed number of attributes (2) in this case and split node. Continue same process excluding attribute used for split.

→ Evaluate Trees

lets say we created 100 bootstrapped dataset & thus 100 trees.

Now, pass each entity from the out of bag data into all the 100 trees and the class predicted maximum no. of times is desired output.

→ Change Parameters & go to Step 1

parameters like bootstrapped data coverage & number of random attributes to be selected to create tree can be changed.

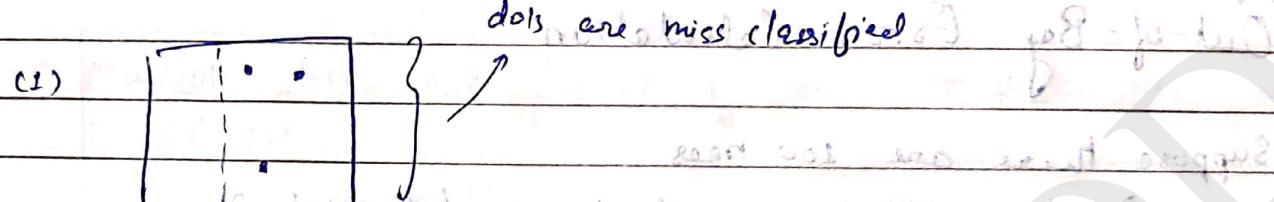
Note

- The process of aggregating 100 tree & thus 100 bootstrapped dataset & getting better result is called bagging

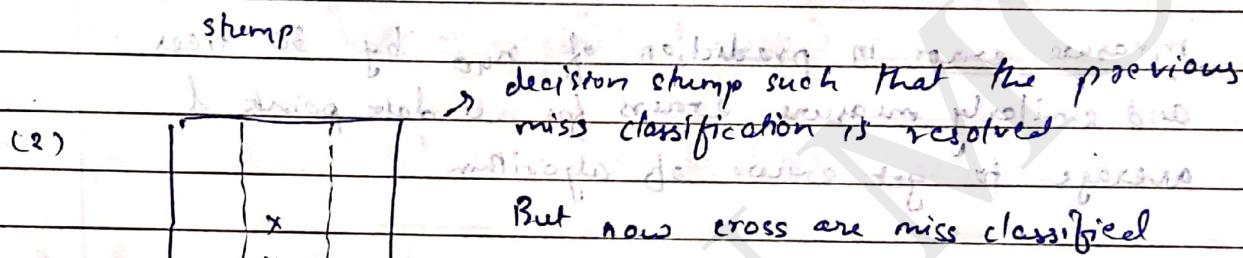
* Boosting

→ Decision Stump

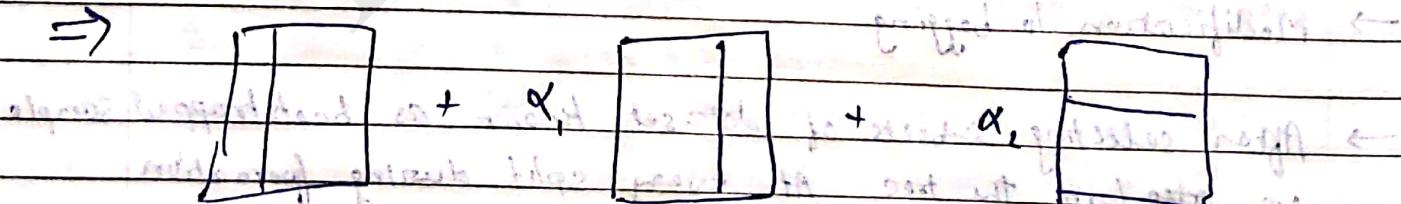
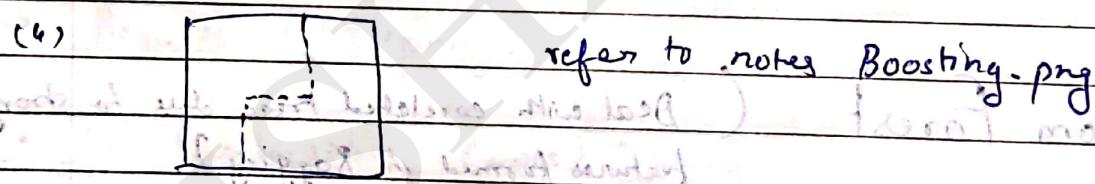
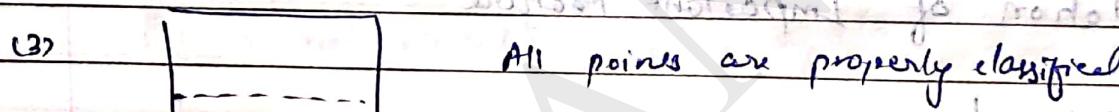
binary tree with one ^{root} node & two child nodes



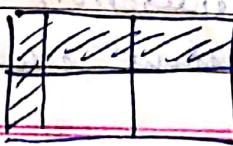
initial decision



But now cross are miss classified



Result is weighted sum of all classifiers =
successive classifiers are weighted



AdaBoost Algorithm

→ Imp points

(1) AdaBoost combines "weak learners" to make classification

These weak learners are almost always stumps

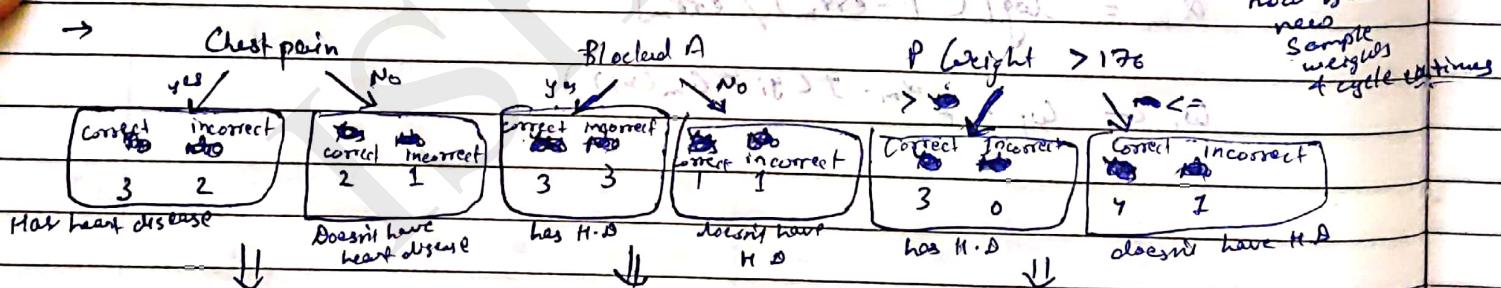
(2) Some stumps get more (say) in classification than others

(3) Each stump is made by taking previous stumps' mistake into consideration

Example

total no. of samples = 8
so that all sample weights add up to 1

	Chest Pain	Blocked Arteries	Patient	Heart Disease	Sample M	New Sample	Normalized Weights
1	Yes	Yes	265	Yes	1/8	0.05	$0.05/0.68 = 0.07$
2	No	Yes	180	Yes	1/8	0.05	0.07
3	Yes	No	210	Yes	1/8	$0.05 \times 2 = 0.10$	0.07
4	Yes	Yes	167	Yes	1/8	0.33	$0.33/0.68 = 0.49$
5	No	Yes	156	No	1/8	0.05	0.07
6	No	Yes	125	No	1/8	0.05	0.07
7	Yes	No	168	No	1/8	0.05	0.07
8	Yes	Yes	172	No	1/8	0.05	0.07
<u>Total 8 samples</u>							
							1



$$G_m = 0.47$$

$$\text{Gini} = 0.5$$

$$\text{Gini} = 0.2$$

Minimum

thus first stump in classifications

Now, error for 1st stump is $\frac{1}{8}$ because 1 element

$\frac{1}{8}$ since only one value is missclassified

i.e [it is incorrect] it says that person is having heart disease but the node in which it is placed says person shouldn't have heart disease thus missclassified

∴ Amount of information conveyed by the stump in final output

OR

Amount of say

∴ amount of say = 0.97 { weights of various

amount of say = $\frac{1}{2} \log \left(\frac{1 - \frac{3}{8}}{\frac{3}{8}} \right) = 0.42$ } chest pain

→ Now let's see how to modify weights so next stump will take into consideration errors in current

→ In P weight errors was due to sample 4 so we will increase sample 4's weight to decrease all other weights

• Formula (increase weight)

new sample weight = sample weight $\times e^{\text{amount of say}}$

$$= \frac{1}{8} \times e^{0.97} = 0.33$$

* Formula (Decrease weight)

$$\text{New sample weight} = \text{sample weight} \times e^{-\text{amount of say}}$$

weight

$$\text{Second sample weight} = \frac{1}{8} \times e^{-0.97} = 0.005$$

→ Now second stump can be calculated by

(1) Weighted Gini index

$$\text{eg chest pain} = \left(1 - \left(\frac{3}{5} \right)^2 - \left(\frac{2}{5} \right)^2 \right) \left(\frac{0.07 + 0.07 + 0.49}{0.07 + 0.07} \right)$$

$$\left(1 - \left(\frac{2}{3} \right)^2 - \left(\frac{1}{3} \right)^2 \right) (0.07 \times 3)$$

similar for Blocked Arteries } 4 P weight } 4 values with

(2) sample Range

weights

1	0.07	< 0.07
2	0.07	$0.07 < \dots < 0.14$
3	0.07	$0.14 < \dots < 0.21$
4	0.49	$0.21 < \dots < 0.30$
5	0.07	.
6	0.07	.
7	0.07	.
8	0.07	.

choose random eg $\Rightarrow 0.4$ → make a new table & put the sample in that table

select the entry corresponding to the range in which it falls

reset sample weights as $1/N$ and make

→ lets say we made 5 stumps
 now if we feed a given set of 3 stumps say
 has heart disease & 2 stumps say doesn't have

$$\text{Yes} = \sum_{i=1}^3 \text{Amount of say (Has heart disease)}$$

$$\text{No} = \sum_{i=1}^2 \text{Amount of say (doesn't have H.D.)}$$

if Yes > No → Ans is Yes
 elif No > Yes → Ans is No

* Gradient Boosting

Algorithm (Regression)

→ Input - Data : $\{(x_i, y_i)\}_{i=1}^n$

Loss : $L(y_i, f(x_i))$ → prediction

Step 1

Height	Favourite color	Gender	Weight	$x_{i,1}$
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

differentiating the loss function & equating to 0 to get the initial prediction.

$$\frac{d}{d \text{prediction}} \sum_{i=1}^n \frac{1}{2} (y_i - \text{prediction})^2 = 0$$

$$- \sum_{i=1}^n (y_i - \text{prediction}) = 0$$

$$- \left[(88 - \text{prediction}) + (76 - \text{prediction}) + (58 - \text{prediction}) \right] = 0$$

$$\text{prediction} = \frac{88 + 76 + 56}{3} \quad (\text{Avg of all samples})$$

Step 2

(A) Residuals are calculated from latest predictions

$$\text{formula} = - \left[\frac{d \text{ [Loss]}}{d \text{ prediction}} \right]$$

$$r_{11} = (y_1 - \text{prediction}) = 88 - 73.3 = 14.7$$

residuals

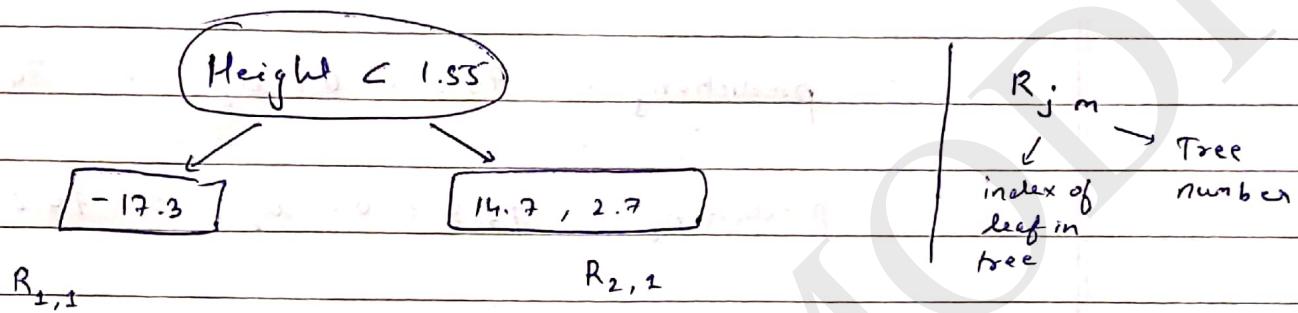
$$r_{21} = (y_2 - \text{prediction}) = 76 - 73.3 = 2.7$$

$y_{i,1}$
sample number
 \downarrow
tree number

$$r_{31} = (y_3 - \text{prediction}) = 58 - 73.3 = -15.3$$

So we make first tree with 3 features other than weights & considering $r_{i,1}$ as target class

(B) Fit a regression tree with terminal region



(C) Get a terminal value at leaf Nodes

Now we need to find a value u which helps us move in the direction where the loss reduces.

$$\text{for } R_{1,1} \quad \frac{d}{du} \sum (y_i - (\text{prediction} + u))^2 = 0$$

$$-(y_i - (\text{prediction} + u)) = 0$$

$$-y_i + \text{prediction} + u = 0$$

$$u = y_i - \text{prediction}$$

$$u = 56 - 73.3$$

$$u_{1,1} = -17.3$$

$$\text{for } R_{2,2} \quad 2u = (88 - 73.3) + (76 - 73.3)$$

$$u_{2,2} = 8.7$$

(D) Update

$$\text{prediction}_{\text{new}} = \text{prediction}_{\text{prev}} + \frac{1}{M} \times (\mu \text{ of the leaf node that classifies the sample})$$

$$\therefore \text{prediction}_1 = 73.3 + 0.1 \mu_{2,1} = 74.2$$

$$\text{prediction}_2 = 73.3 + 0.1 \mu_{2,1} = 74.2$$

$$\text{prediction}_3 = 73.3 + 0.1 \mu_{1,1} = 71.6$$

Now, repeat step 2, M no. of times (generally M=100)

Output

lets say M = 2

test \Rightarrow 1.4 Green - Male

value = prediction_{initial} + ~~something~~

$$+ 0.1 \times \text{height} < 1.55$$

$$\mu_{1,1} = -17.3 \quad \mu_{2,1} = 8.7$$

$$+ 0.1 \times \text{height} < 1.55$$

$$\mu_{1,2} = -15.6 \quad \mu_{2,2} = 7.8$$

$$\therefore \text{value} = 73.3 + (0.1 \times -17.3) + (0.1 \times -15.6) = 70$$

Algorithm (Classification)

→ Input - Data = $\{(x_i, y_i)\}_{i=1}^n$

$$\text{Loss} = L(y_i, f(x)) \rightarrow \text{prediction}$$

~~Step 1~~

Likes Popcorn	Age	Favourite Color	Loves Troll 2
Yes	12	Blue	Yes
No	87	Green	Yes
No	44	Blue	No

Here,

$$\text{No. of samples} \\ - \text{Loss function} = -\sum_{i=1}^n y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)$$

- Let's say, to start with \hat{y}_i for (loves troll 2) is $\left[\frac{2}{3} = 0.67 \right]$

$$P = \hat{y}_i = \frac{2}{3}$$

- $\log(\text{Odds}) = \log\left(\frac{2}{1}\right)$ or $\left(\begin{array}{l} \log\left(\frac{2}{1}\right) \\ \text{odds of} \\ \text{loves troll 2} \end{array} \right)$ Not considered for this example
- $\log(\text{Odds}) = \log\left(\frac{1}{2}\right)$ or $\left(\begin{array}{l} \log\left(\frac{1}{2}\right) \\ \text{odds of not} \\ \text{loving troll 2} \end{array} \right)$

$$P = \frac{e^{\log(\text{Odds})}}{1 + e^{\log(\text{Odds})}}$$

&

$$\log(\text{Odds}) = \log\left(\frac{P}{1-P}\right)$$

These were all the formula's, Now lets comeback to minimizing the loss function

$$-\left[y_i \log(p) + (1-y_i) \log(1-p)\right]$$

ignoring summation for simplicity

$$-\left[y_i (\log(p) - \log(1-p)) + \log(1-p)\right]$$

$$-\left[y_i \log\left(\frac{p}{1-p}\right) + \log\left(1 - \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}\right)\right]$$

$$-\left[y_i \log(\text{odds}) + \log\frac{1}{1 + e^{\log(\text{odds})}}\right]$$

$$\left[\log\left(1 + e^{\log(\text{odds})}\right) - y_i \log(\text{odds})\right]$$

Now differentiating 4 equations to 0

$$\frac{d}{d \log(\text{odds})} \left[\log\left(1 + e^{\log(\text{odds})}\right) - y_i \log(\text{odds}) \right]$$

$$\frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} - y_i$$

$$p = y_i$$

Using the derived formula along with summation

$$\sum_{i=1}^N (p - y_i) = 0$$

$$= (p - 1) + (p - 1) + (p - 0) = 0$$

$$3p - 2 = 0$$

$$p = \frac{2}{3}$$

Thus,

$$\log(\text{odds}) = \log\left(\frac{p}{1-p}\right) = \log(2)$$

So, step 1 helps us find the initial probability value at which loss is minimum.

Step 2

~~latest~~ formula = $\left[\frac{d \text{ [Loss]}}{d \log(\text{odds})} \right]$

(A) Residuals are calculated from latest prediction

$$y_{11} = (y_1 - p) = 1 - 0.67 = 0.33$$

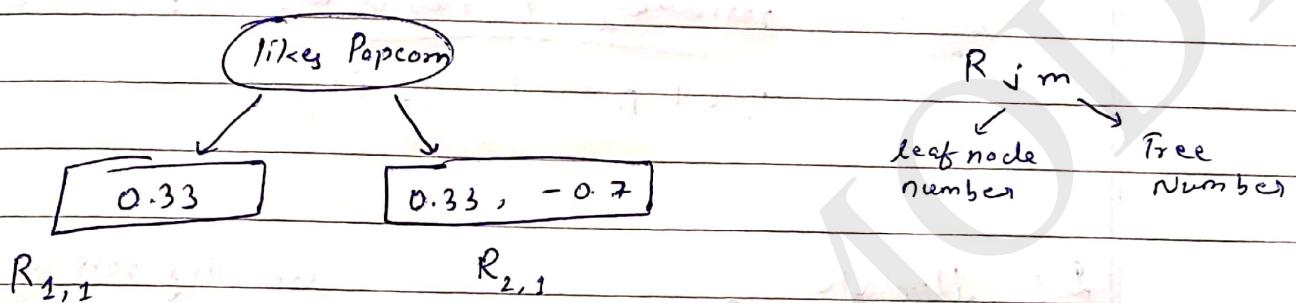
Sample number Tree number

$$y_{21} = (y_2 - p) = 1 - 0.67 = 0.33$$

$$y_{31} = (y_3 - p) = 0 - 0.67 = -0.67$$

So we make first tree with 3 features other than target class & we take y_{im} (residuals) as the target ~~class~~ class.

(B) Fit a classification Tree



(C) Get Terminal value of leaf Node

Now we need to find a value m which helps us move in direction such that loss decreases.

$$\text{For } R_{1,1} \quad d \left[\log \left(1 + e^{\log(\text{odds}) + m} \right) - y_i (\log(\text{odds}) + m) \right]$$

$$d_m$$

differentiating using Taylor series & equating to 0

$$\frac{d}{dm} \left(\log \left(1 + e^{\log(\text{odds})} \right) - y_i \log(\text{odds}) \right)$$

$$m =$$

$$\frac{d^2 \left(\log \left(1 + e^{\log(\text{odds})} \right) - y_i \log(\text{odds}) \right)}{dm^2}$$

$$\mu_{1,1} = \frac{\text{residual}}{p(1-p)} = 1.5$$

For $R_{2,1}$

similarly,

$$\mu_{2,1} = \frac{\text{residual}_2 + \text{residual}_3}{p_2(1-p_2) + p_3(1-p_3)} = -0.77$$

(D) Update

learning rate

$$\log(\text{odds})_{\text{new}} = \log(\text{odds})_{\text{prev}} + \eta \times (\mu \text{ of the leaf node that classifies the sample})$$

$$\begin{aligned} \log(2/1) &+ 0.8 \times 1.5 \\ \log(2/1) &+ 0.8 \times -0.77 \\ \log(1/2) &+ 0.8 \times -0.77 \end{aligned} \quad \left. \begin{array}{l} \log(\text{odds}) \\ \text{for new predictions} \\ \text{for all samples} \end{array} \right\}$$

Now,

repeat step 2, M no. of times (generally M = 100)

* Application of KNN Collaborative filtering (Recommender Systems)

There are two types of Recommender Systems

(1) Item Recommendation

(2) Rating Prediction

Let's say

U: set of users

p: $U \times S \rightarrow \text{Rating}$

S: set of items

next to you (31, x39) ... (31, x39), (31, x39) and

This follows learning p from data in base of large numbers.

- use p to predict utility value of each item to each user



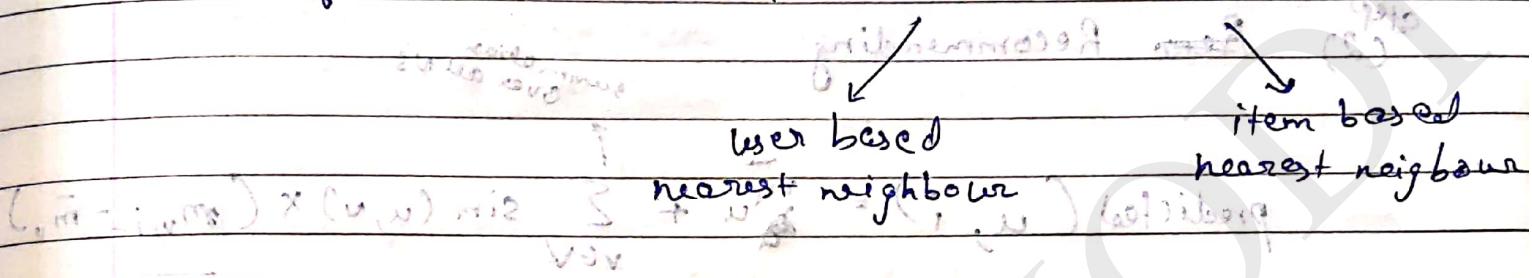
Subcategories of recommendation system based on user

(1) Content Based

Based on previous likes & dislikes of user

(2) Collaborative Filtering

Based on rating of other users which are similar to current



→ User based NNT

step

(1) Use Pearson's correlation coefficient

lets say dataset of user & ratings is as follows

Example

	movie 1	m2	m3	m4	m5
user 1	4	3	-	.	5
user 2	1	3	-	3	2
user 3	-	-	3	5	1

$$\text{Sim}(u, v) = \frac{\sum_{i=1}^m (m_{u,i} - \bar{m}_u)(m_{v,i} - \bar{m}_v)}{\sqrt{\sum_{i=1}^m (m_{u,i} - \bar{m}_u)^2} \sqrt{\sum_{i=1}^m (m_{v,i} - \bar{m}_v)^2}}$$

user with which we want to compare

user for which we want to predict

thus we find k most similar v 's for given u

lets say $k = 3$, thus we get

some sets $\{v_1, v_2, v_3\}$ which have highest similarity with u
in terms of rating and signs

Step (2) Recommending

$$\text{predicted } (m_{u,i}) = \bar{m}_u + \sum_{v \in V} \frac{\text{sim}(u, v) \times (m_{v,i} - \bar{m}_v)}{\sum_{v \in V} |\text{sim}(u, v)|}$$

→ Item based NN

for above given Example

$$\text{Step (1)} \quad \text{sim}(i, j) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}$$

$$\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \quad \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}$$

Step (2) Select k most nearest neighbours [k nearest j 's to i]

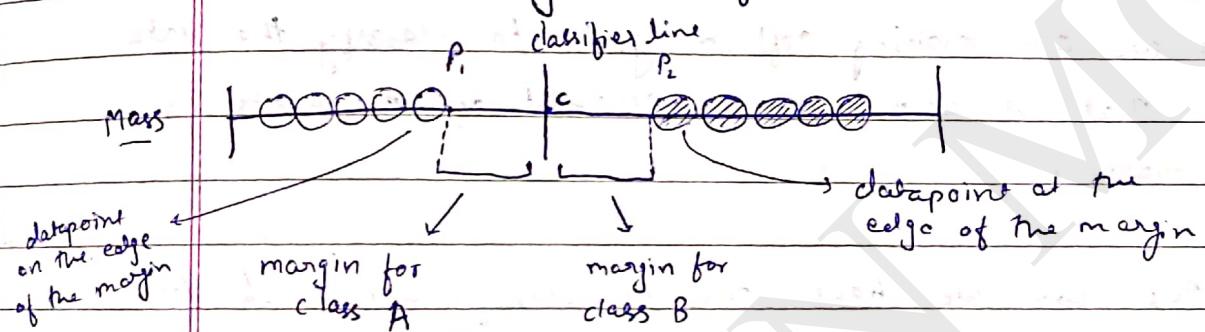
$$p(u, i) = \frac{\sum_{j \in S} r_{u,j} * \text{sim}(i, j)}{\sum_{j \in S} \text{sim}(i, j)}$$

* Support Vector Machines

lets assume data that consist of ~~two~~ one attribute and one target class

Mass	Obese	classification problem
(Attribute)	(Target class)	

→ Maximum Margin Classifier

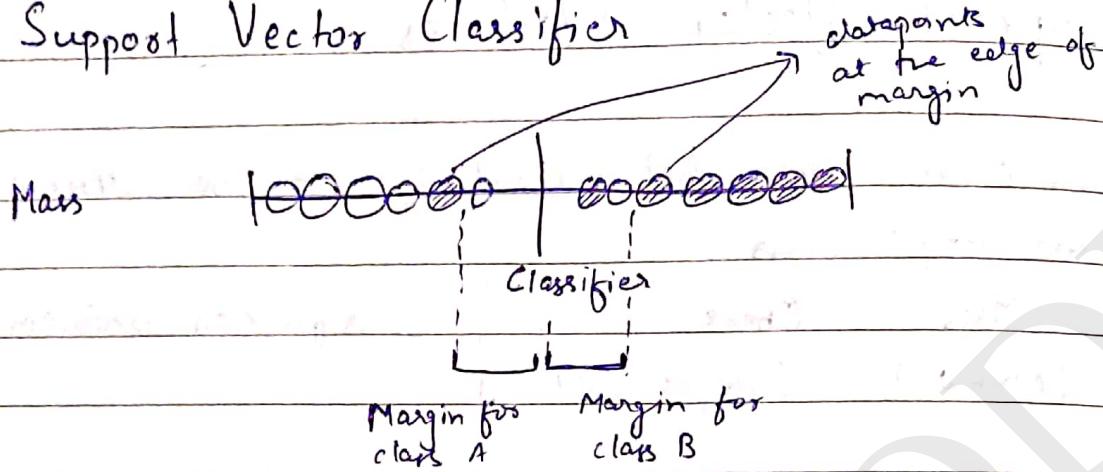


We make classifier line such that most of the points are correctly classified & the margins for target classes should be as large as possible.

Here we could have made P_2 the margin boundary between c & P_2 rather than at P_1 but then it would not be maximum margin. Similar thing applies for P_1 also.

Here there is no datapoint inside the margins thus it is also called hard margin classifier.

→ Support Vector Classifier



Here we are making soft margins to classify the data such that we allow a set amount of ~~in~~ points in the margins.

In this case we have one missclassification & two correct classification in the data. Here there are datapoints within the margin, thus it is called soft margin classifier.

These margins are decided using cross-validation. The datapoints at the edge of the margin are called support vector & the method is also called support vector classifier.

This is a 1 dimensional space, similar functionality can be obtained in a 2D space, ..., no space as well.

→ Support Vector Machines

In the previous examples the data is linearly separable. Let's assume the data is or linearly non-separable. Let's assume the following data.

Dosage	Cured
Attribute	(Yes/No) Target Class
1 00000	00000

1 00000 00000 00000 1

Here we transform these datapoints to higher dimensional vector space and then try to classify using n -dimensional hyperplane.

For eg - we have the values on the n -axis. We can get all the corresponding y -axis values such that $y = n^2$.

Here we transformed the data using $y = n^2$, but in practice there are Kernel Functions which help us do that.

- (1) Polynomial Kernel
- (2) Radial Basis Function Kernel

* Polynomial Kernel

Formula $\rightarrow (a \times b + r)^d$

✓ ↓ d → degree of polynomial
 datapoint 1 datapoint 2 ↓ → coefficient of polynomial

lets say, $d=2$ & $r=\frac{1}{2}$

$$\therefore (a \times b + \frac{1}{2})^2 = a^2 b^2 + 2ab \cdot \frac{1}{2} + \frac{1}{4}$$

$$= a^2 b^2 + ab + \frac{1}{4}$$

$$= ab + a^2 b^2 + \frac{1}{4}$$

$$= (a, a^2, \frac{1}{2}) \cdot (b, b^2, \frac{1}{2})$$

Since 3rd dimension is same ie - $\frac{1}{2}$ we consider only 2 dimensions

$$(a, a^2) \quad \& \quad (b, b^2)$$

↓ ↑ ↓ ↑
 x-axis y-axis x-axis y-axis

Similarly we can transform the entire data into a higher dimension

In practice we only need the ~~not~~ higher dimensional relation between the datapoints. It is calculated as follows

Usage of datapoint
of 1 & 2

$$(a \times b + r)^2 = (9 \times 14 + 1)^2 = 126.5^2 = 16002.25$$

Here r & d are determined using cross-validation

* Radial Basis Function Kernel

RBF kernel doesn't actually transform the data. It finds the relation between datapoints in infinite dimension

Formula $\rightarrow e^{-r(a-b)^2}$

~~Extra info for understanding~~
Now, let's take polynomial kernel with $r=0$, $d=1$
then it will be $(a) \cdot (b) = ab$

$$\text{error, } \delta_{xx} \rightarrow K(x, x) = (a^0, b^0) \cdot (a^0, b^0) = a^2 b^2$$

$$\text{error, } \delta_{xy} \rightarrow K(x, y) = (a^0, b^0) \cdot (a^1, b^1) = a^2 b^3$$

$$r=0, d=2 \rightarrow (a^2) \cdot (b^2) = a^2 b^2$$

$$r=0, d=3 \rightarrow (a^3) \cdot (b^3) = a^3 b^3$$

and so on ..

Now, if we add all these values

$$ab + a^2 b^2 + a^3 b^3 + \dots + a^\infty b^\infty$$

if we break this into dot product we get

$$(a, a^2, \dots, a^\infty) \cdot (b, b^2, \dots, b^\infty)$$

The datapoints are transformed to an infinite vector space

Now, for RBF kernel -

$$e^{-\gamma(a-b)^2}, \text{ if } \gamma = 1/2$$

$$e^{-\frac{1}{2}(a-b)^2} = e^{-\frac{1}{2}(a^2 + b^2 - 2ab)} = e^{\underbrace{-\frac{1}{2}(a^2 + b^2)}_{1 \text{ term}}, \underbrace{-ab}_{2 \text{ term}}}$$

Now using Taylor series on 2 term

$$\bullet f(n) = f(a) + \frac{f'(a)}{1!}(n-a) + \frac{f''(a)}{2!}(n-a)^2 + \dots + \frac{f^{(\infty)}(a)}{\infty!}(n-a)^\infty$$

$$\bullet e^n = e^a + \frac{e^a}{1!}(n-a) + \frac{e^a}{2!}(n-a)^2 + \dots + \frac{e^a}{\infty!}(n-a)^\infty$$

Taking $a=0$ & Simplifying

$$\bullet e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots + \frac{1}{\infty!}x^\infty$$

Replacing x with ab

$$e^{ab} = 1 + \frac{1}{1!}ab + \frac{1}{2!}a^2b^2 + \dots + \frac{1}{\infty!}a^\infty b^\infty$$

If we break this to dot product we get

$$(1, \sqrt{\frac{1}{1!} a}, \sqrt{\frac{1}{2!} a^2}, \dots, \sqrt{\frac{1}{\infty!} a^\infty}) \quad \left. \right\} \text{dot product}$$

$$(1, \sqrt{\frac{1}{1!} b}, \sqrt{\frac{1}{2!} b^2}, \dots, \sqrt{\frac{1}{\infty!} b^\infty})$$

Now,

$$e^{-\frac{1}{2}(a-b)^2} = e^{-\frac{1}{2}(a^2+b^2)} \quad (\text{dot product})$$

- Let $e^{-\frac{1}{2}(a^2+b^2)} = s^2$

$$\therefore s = \sqrt{e^{-\frac{1}{2}(a^2+b^2)}}$$

Thus,

$$e^{-\frac{1}{2}(a-b)^2} = (s, s\sqrt{\frac{1}{1!} a}, s\sqrt{\frac{1}{2!} a^2}, \dots, s\sqrt{\frac{1}{\infty!} a^\infty})$$

$$(s, s\sqrt{\frac{1}{1!} b}, s\sqrt{\frac{1}{2!} b^2}, \dots, s\sqrt{\frac{1}{\infty!} b^\infty})$$

In practice, we just calculate

$e^{-\frac{1}{2}(a-b)^2}$ to get relationship between these dimensions in infinite space.