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

$$Y = \alpha_0 + \alpha_1 X$$

↓ regression coefficients

} following eqⁿ is used for simple linear regression

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

* Statistical Method

$$A = \begin{bmatrix} \Sigma x_1^2 & \Sigma x_1 x_2 \\ \Sigma x_1 x_2 & \Sigma x_2^2 \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

As seen,

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

we can also say

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

using elimination in (1) & (2)

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

for simplicity

$$y = \alpha_1 x$$

$$x^T y = \alpha_1 x^T x$$

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

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

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

Now,

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

Note →

$$\alpha_1 = \underbrace{\frac{\text{cov}(x, y)}{\text{var}(x) \times \text{var}(y)}}_{\text{correlation coefficient}} \times \left(\frac{\text{Standard deviation}(y)}{\text{Standard deviation}(x)}\right)$$

The above formula is equivalent to derived formula

(2) Multiple Linear Regression

→ The number of independent variable is more than 1

let's 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

$$Y = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 \quad - \quad (1)$$

we can also write

$$\alpha_0 = \bar{Y} - \alpha_1 \bar{X}_1 - \alpha_2 \bar{X}_2 \quad - \quad (2)$$

using elimination on (1) & (2)

$$(Y - \bar{Y}) = \alpha_1 (X_1 - \bar{X}_1) + \alpha_2 (X_2 - \bar{X}_2)$$

for simplicity

$$y = \alpha_1 x_1 + \alpha_2 x_2 \quad - \quad (3)$$

$$\underset{4\times1}{y} = \underbrace{\underset{4\times1}{x_1} \quad \underset{4\times1}{x_2}}_{4\times2} \cdot \underbrace{\begin{matrix}\alpha \\ 1\end{matrix}}_{2\times1}$$

It can also be written as

$$\alpha = \left([x_1 \; x_2]^T [x_1 \; x_2] \right)^{-1} \cdot [x_1 \; x_2]^T y$$

$$= \begin{bmatrix} \Sigma x_1^2 & \Sigma x_1 x_2 \\ \Sigma x_1 x_2 & \Sigma x_2^2 \end{bmatrix}^{-1} \begin{bmatrix} \Sigma x_1 y \\ \Sigma x_2 y \end{bmatrix}$$

$$= \frac{\begin{bmatrix} \Sigma x_2^2 & -\Sigma x_1 x_2 \\ -\Sigma x_1 x_2 & \Sigma x_1^2 \end{bmatrix} \begin{bmatrix} \Sigma x_1 y \\ \Sigma x_2 y \end{bmatrix}}{\Sigma x_1^2 \Sigma x_2^2 - (\Sigma x_1 x_2)^2}$$

$$\therefore \alpha_1 = \frac{\Sigma x_2^2 \Sigma x_1 y - \Sigma x_1 x_2 \Sigma x_2 y}{\Sigma x_1^2 \Sigma x_2^2 - (\Sigma x_1 x_2)^2}$$

$$\alpha_2 = \frac{\Sigma x_1^2 \Sigma x_2 y - \Sigma x_1 x_2 \Sigma x_1 y}{\Sigma x_1^2 \Sigma x_2^2 - (\Sigma x_1 x_2)^2}$$

Now,

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

Note -

- $$\Sigma x_i^2 = \Sigma (X_i - \bar{X}_i)^2 = \Sigma X_i^2 - N \bar{X}_i^2$$

  $$= \Sigma X_i^2 - N \left(\frac{\Sigma X_i}{N} \right)^2$$

from (3)

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

- $\Sigma x_i y = \Sigma (x_i - \bar{x}_i)(y - \bar{y})$

from (3)

$$= \Sigma x_i y - \Sigma \bar{x}_i y - \Sigma x_i \bar{y} + \Sigma \bar{x}_i \bar{y}$$

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

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

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

$$= \Sigma x_i y - N \frac{\Sigma x_i}{N} \frac{\Sigma y}{N}$$

$$= \Sigma x_i y - \frac{\Sigma x_i \Sigma y}{N}$$

Linear Regression (Model Based)

$y = w_1 x + w_0$ — Linear

$y = w_2 x^2 + w_1 x + w_0$ — Quadratic

$y = w_3 x^3 + w_2 x^2 + w_1 x + w_0$ — 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

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

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

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

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

$$= \frac{1}{2N} \sum -2 (y_i - \hat{y}_i) x_i$$

$$= -\frac{1}{N} \Sigma (y - \hat{y}_i) x_i$$

generalizing

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

$$w_1 \rightarrow -\frac{1}{N} \Sigma (y - \hat{y}_i) x_i$$

$$w_2 \rightarrow -\frac{1}{N} \Sigma (y - \hat{y}_i) x_i^2$$ and so on.

→ Steps for Linear Regression

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

Lets 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,

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

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

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

- $\frac{\partial \hat{y}_i}{\partial z} = \frac{1}{\frac{1+e^{-z}}{\partial z}} = \hat{y}_i (1-\hat{y}_i)$

- $\frac{\partial z}{\partial w_0} = 1$

similarly $\frac{\partial z}{\partial w_1} = x$

$\therefore \frac{\partial L}{\partial w_0} = -\frac{(y_i - \hat{y}_i)}{\cancel{\hat{y}_i (1-\hat{y}_i)}} \cancel{\hat{y}_i (1-\hat{y}_i)} \cdot 1$ for w_0

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

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

Validation

→ Normal Validation

Divide the dataset into 3 parts

Train, Validation, Test
(Train + Validation: 70%; Test: 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

Lets say 70% of the train split contains 200 samples

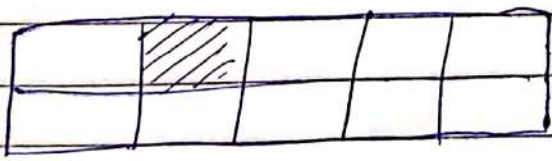
In K-Fold cross validation the data is divided in bins/partitions

eg- K = 10

thus, 10 partitions & 200 data points

} 10 partitions with 20 datapoints each

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

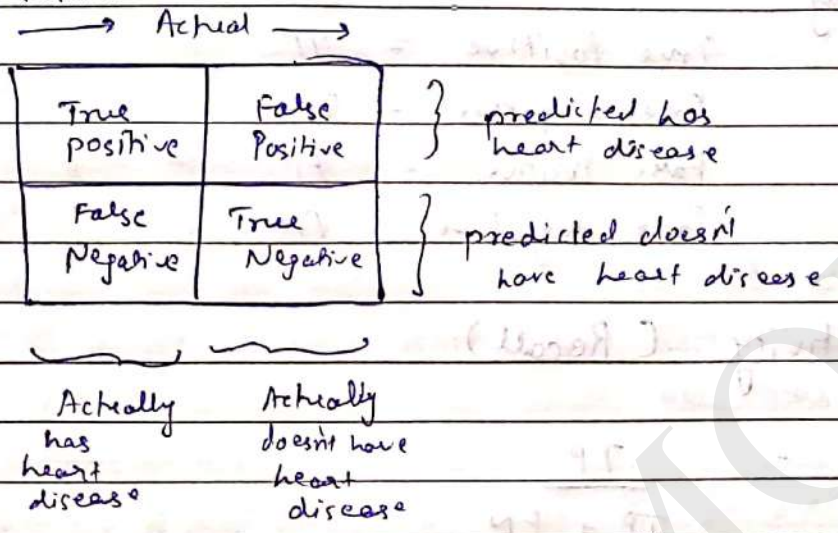


} take highlighted as validation & other as training.

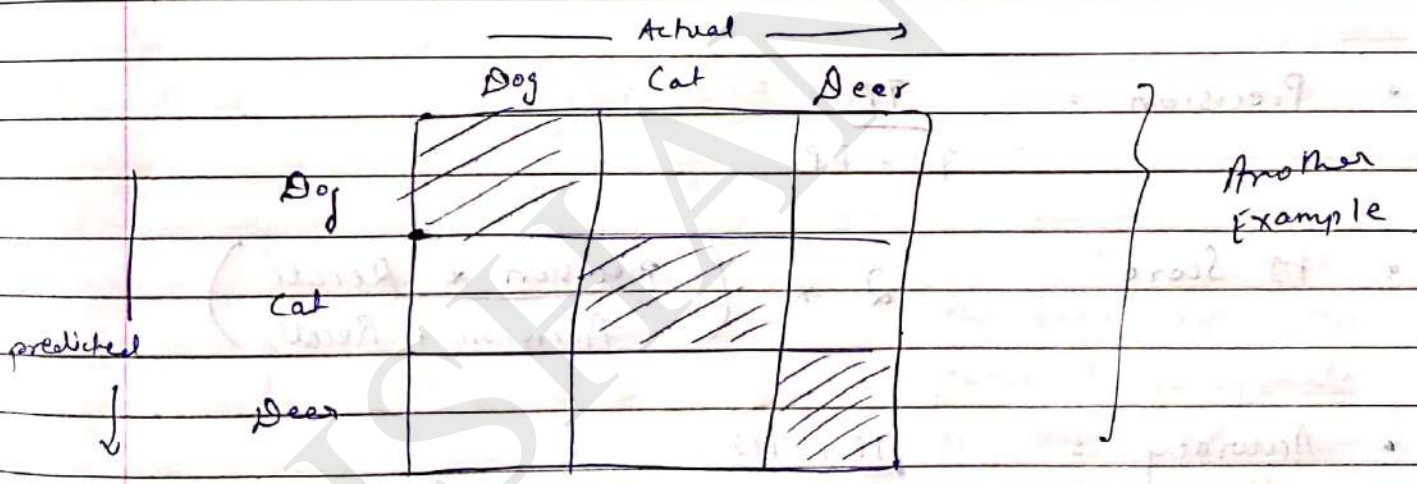
Similarly for all partitions & average all accuracy to get results.

Evaluating Model Results

→ Confusion Matrix



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



Here the highlighted region is correct prediction others are incorrect.

→ Some Formula's

Let's say,

True Positive = TP
True Negative = TN
False Positive = FP
False Negative = FN

- Sensitivity (Recall)

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

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

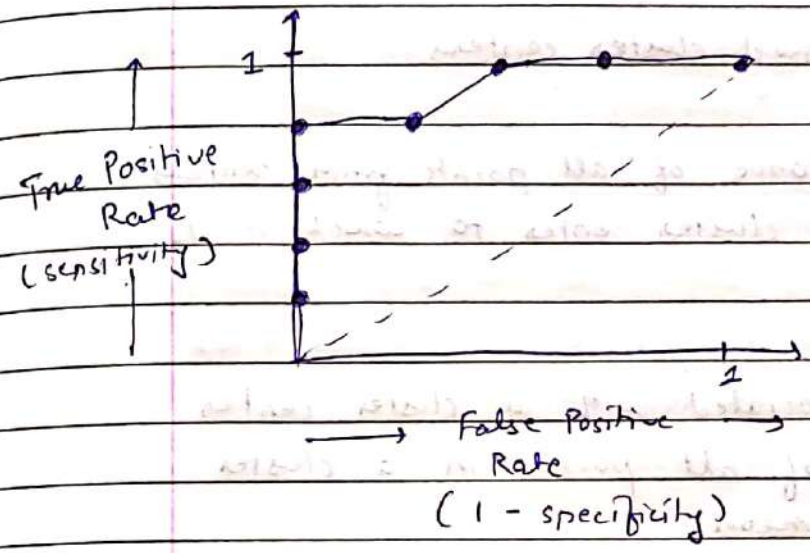
- Precision $= \frac{TP}{TP + FP}$

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

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

→ ROC & AUC

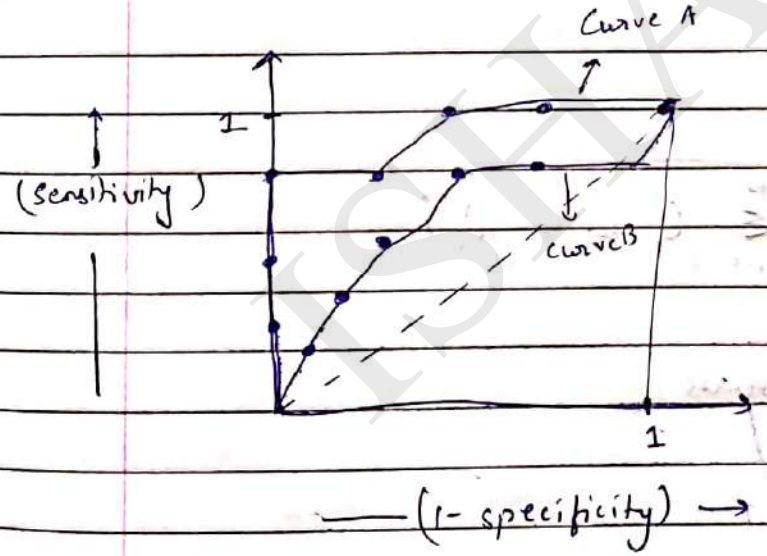
- ROC (Receiver Operator Characteristic)



Here we want the false positive rate to be minimum & true positive rate to be maximum

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

- AUC (Area Under Curve)



Here in the figure the area under the curve A is greater the area under curve B and thus the ML algorithm used to generate models of 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

* KNN (K-Nearest Neighbours Classification)

Supervised learning

Split the Data

train, test, validation
↓
70% 10% 20%

Now, validation is of two types
(1) Normal validation
(2) k-fold cross validation

let's say

	A	B	C	D (Target)	
1	1	1	2	0	training data
2	2	2	2	0	
3	3	3	1	1	

features

	A	B	C	
1	2	3	1	test data

Euclidean distance with training (from training data) → target class

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

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

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

Arrange distances in sorted order

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

→ Now, if $k = 1$

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 NN

∴ predicted target class is (0)

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

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

* K-Means Clustering (unsupervised learning)

K = 2 } → 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 center at the mean.

Loop until
Cluster centers don't change significantly

→ Formula's

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

N = no. of features
x = datapoint
x' = center

- Center of cluster } $= \frac{1}{N} \sum_{i=1}^{N} x_i$

N = no. of datapoints in the cluster

x_i = individual datapoint of the cluster

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

  break the loop

* 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 (Boundary point) → If not a core point but is at a distance less than d from a core 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

(3) Out of remaining core points do the same and form clusters

(2) .....

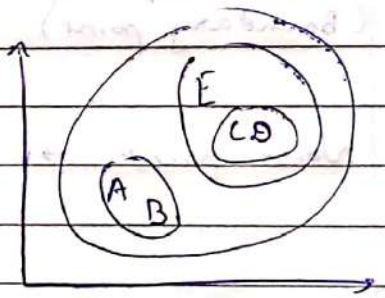
(1)

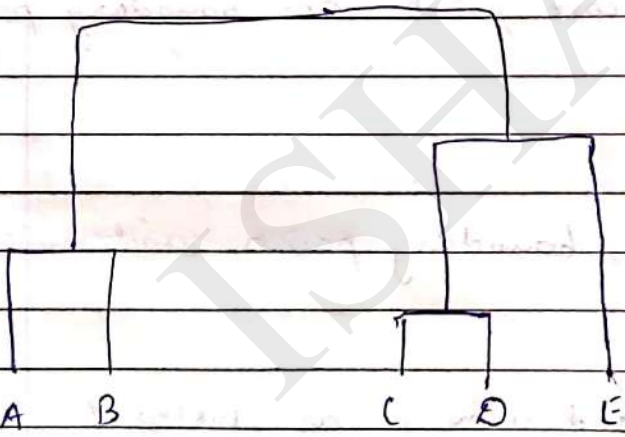
(3) Now add all the boundary points that can be added to cluster 1. Note that boundary points can't extend the cluster like core points

(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 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

distance of point A from point [C, D] = min (distance of point A from point C , distance of point A from D)

(2) Complete linkage

distance of point A from point [C, D] = max (distance of point A from point C , distance of point A from point D)

(3) Average linkage

distance of point A from point [C, D] = avg (distance of point A from point C , distance of point A from point D)

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

$$P(Y|X) = \frac{P(X|Y)\ P(Y)}{P(X)}$$ } Bayes Theorem

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

* Now, lets say Target class has 2 values
C_1 & C_2

Expressing in terms of probability

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

$P(C|X)$ → target class values; X → attributes

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

$$\therefore \quad P(C|X) = P(X|C)\, P(C)$$

Now, ideally

$$P(C \mid X_1, X_2, X_3 \ldots X_n) = \Big(P(X_1 \mid X_2, \ldots X_n, C) * P(X_2 \mid X_1, \ldots X_n, C) * P(X_3 \mid X_1, \ldots X_n, C) * \vdots * P(X_n \mid X_1, \ldots X_{n-1}, C) \Big) * P(C)$$

* By chain rule

But we assume that $X_1, X_2 \ldots X_n$ are independent (thus method is called "naive" bayes)

So,

$$P(X_1 \mid X_2 \ldots X_n, C) = P(X_1 \mid C)$$

and thus the above equation becomes as follows

$$P(C \mid X_1, X_2 \ldots X_n) = \Big(P(X_1|C) * P(X_2|C) * P(X_3|C) * \ldots\ldots * P(X_n|C) \Big) * P(C)$$

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

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

$$\underset{k \in \{1,2,\ldots k\}}{\operatorname{argmax}} \; P(C_k) \prod_{i=1}^{n} P(X_i | C_k) \quad - (1)$$

→ Log Trick (apply log in (1))

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

* 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

Lets say, $P(X_1|C) = \frac{0}{9}$, $P(X_2|C) = \frac{5}{9}$, $P(X_3|C) = \frac{4}{9}$

After laplace smoothing

$$P(X_1|C) = \frac{0+1}{9+(3\times1)}, \quad P(X_2|C) = \frac{5+1}{9+(3\times1)}, \quad P(X_3|C) = \frac{4+1}{9+(3\times1)}$$

3 is used because X_1, X_2, X_3 (3 variables are there)

$$= \frac{1}{12} \qquad = \frac{6}{12} \qquad = \frac{5}{12}$$

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)

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

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

- $X = 1$ success & $X = 0$ failure

Then X has a bernoulli's distribution

Now,

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

(2) Multinomial Naive Bayes

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

$$P(X_1 = x_1, X_2 = x_2, \ldots X_k = x_k) = \left(\frac{\left(\sum_{i=1}^{k} x_i\right)!}{\prod_{i=1}^{k} x_i!} \right) \times \left(\prod_{i=1}^{k} p_i^{x_i} \right)$$

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

$$P(X_1 = 2, X_2 = 3, X_3 = 1) = \left(\frac{6!}{2!\,3!\,1!}\right) \times \left((0.44)^2 (0.55)^3 (0.01)^1\right)$$

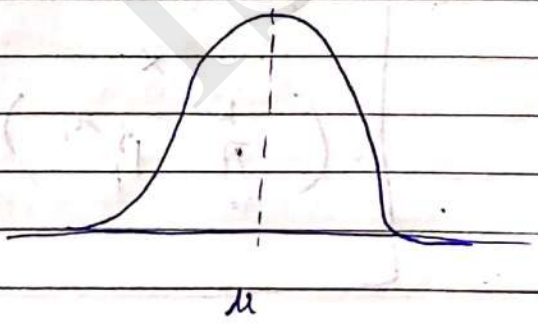
(3) Gaussian Naive Bayes

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

$$P(X_1 = x_1, X_2 = x_2 \ldots, X_k = x_k) = \prod_{i=1}^{m} g(x_i, \mu_{i,c}, \sigma_{i,c})$$

Here, $$g(x_i, \mu_{i,c}, \sigma_{i,c}) = \frac{1}{\sqrt{2\pi\sigma}}\, e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

$$-\infty < x < \infty$$



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.

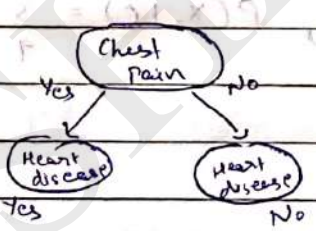
→ Leaf Node

The bottom most ~~nodes~~ nodes of a tree are leaf nodes. These nodes don't have child nodes. Tree can have multiple leaf node

* Decision Trees can be divided into two types

→ Classification Trees (used to solve classification task)

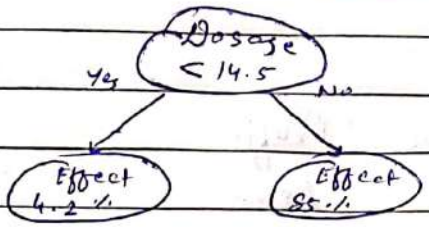
leaf nodes of these trees are associated with target class



} Target class [Heart disease (Yes/No)]

→ Regression Trees (used to solve regression tasks)

leaf nodes of these trees are associated with a number



leaf nodes denote 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.

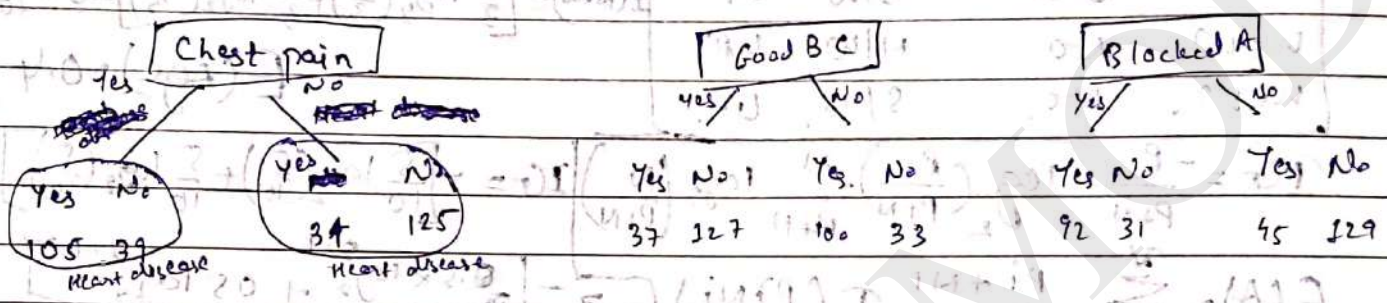
(1) Gini Impurity

(2) Entropy

DECISION TREES

- GINI IMPURITY

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	Yes	Yes
etc	etc	etc	etc



Gini Impurity Formulas:

$\rightarrow 1 - \sum_{i=1}^{\text{No. of classes}} P_i^2$ } $\Rightarrow$ Gini impurity

$\rightarrow \sum_{j=1}^{\text{No. of nodes}} \left(\frac{P_j + N_j}{P + N}\right) \times$ Gini impurity } total impurity count

Now, for chest pain

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

$= 0.395$

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

$= 0.336$

Total impurity count

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

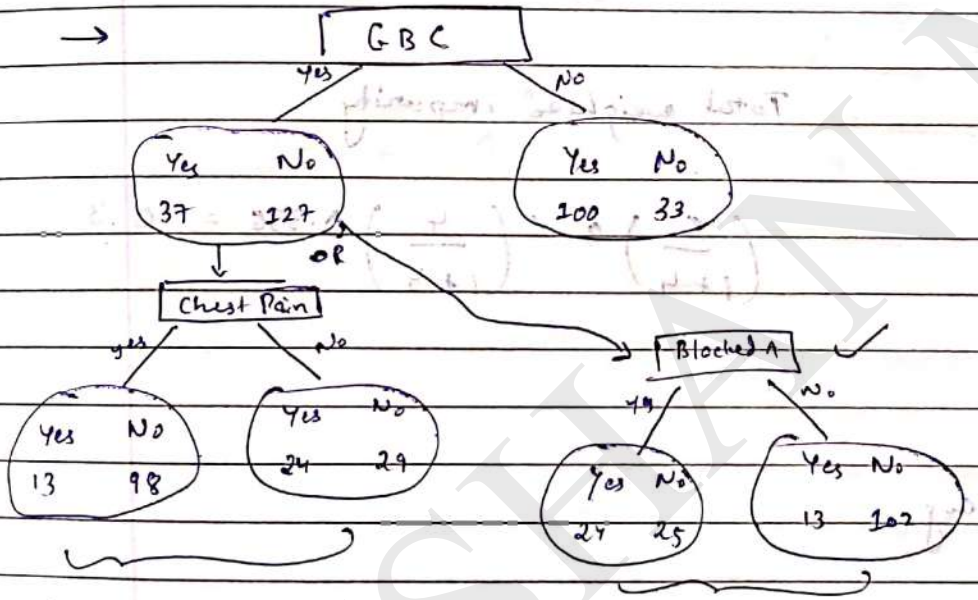
$$= 0.364$$

Similarly for others

GBC = 0.360
BA = 0.381

} minimum out of 3 is GBC so GBC is root node

→



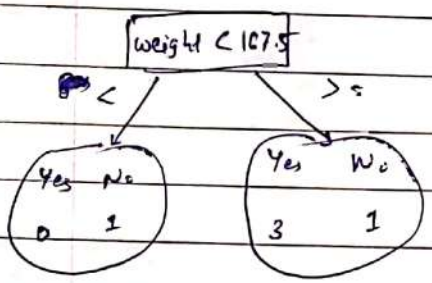
impurity 0.3

impurity 0.290 ✓ is chosen

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

Weight	Heart Disease		
155	No		
Avg 167.5		→	0.3
180	Yes		
Avg 185		→	0.47
190	No		
Avg 205		→	0.27 (minimum) → split using < 205
220	Yes		
Avg 222.5		→	0.4
225	Yes		



impurity = 0 0.375

Total weighted impurity

$$\left(\frac{1}{1+4}\right)0 + \left(\frac{4}{1+4}\right)0.336 = 0.3$$

Similarly for all avg

Decision Trees

ENTROPY

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

- $IG = -\left(\left(\frac{P}{P+N}\right)\log_2\left(\frac{P}{P+N}\right) + \left(\frac{N}{P+N}\right)\log_2\left(\frac{N}{P+N}\right)\right)$
- $E(A) = \sum_{i=1}^{\text{no. of Nodes}} \left(\frac{P_i + N_i}{P+N}\right) I(P_i, N_i)$
- Gain = IG − E(A)

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

→ Now, solving the above example

Page No.
Date

Age :

	Down	up
old	3	0
Mid	2	2
New	0	3

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

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

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

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

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

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

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

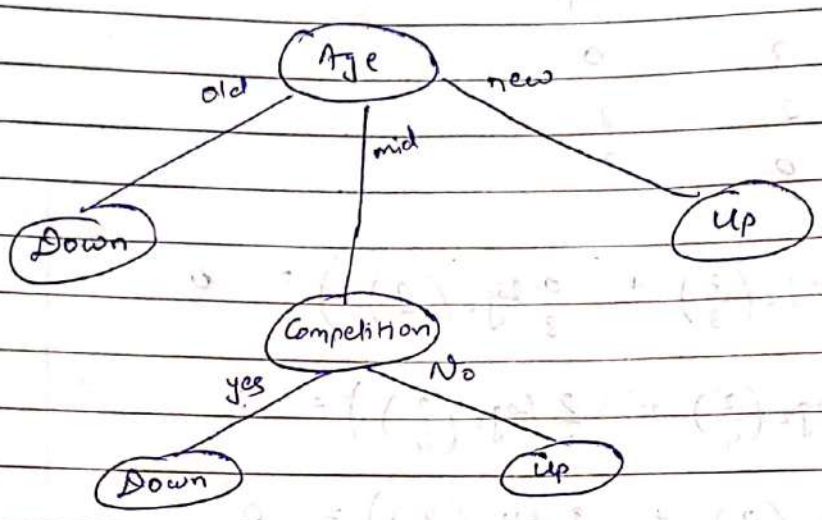
$$= -(-1) = 1$$

$$Gain(Age) = 1 - 0.4 = 0.6$$

Similarly Gain (Competition) = 0.124

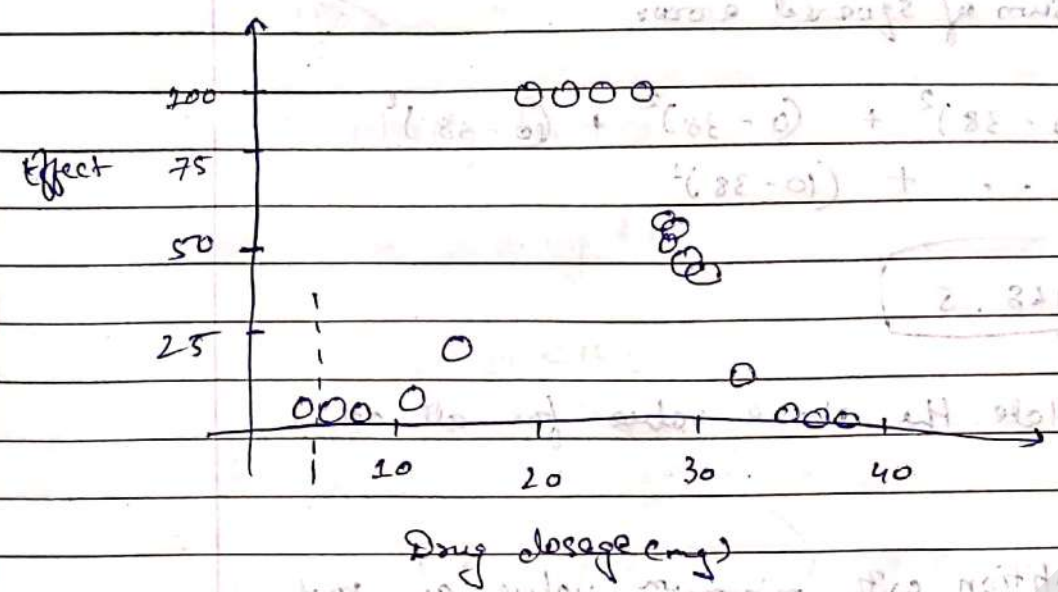
Gain (Type) = 0

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



	Dosage	Effect
1 —	5	0
2 —	6	0
3 —	7	0
	11	10
	15	22
	⋮	⋮
	35	50
	36	100
n —	etc	etc

Now,

form partition

1 & 2 . . . n

1,2 & 3 . . . n

⋮

1 . . . n-1 & n

Serial no. of the Data points

Here seperation is dosage $< \frac{5+6}{2}$ (avg of the two points which are partitioned)

dosage < 5.5

→ For 1st partition

1 & 2 . . . n

avg of effect = $\frac{0}{1} = 0$

avg of effect = $\frac{0+0+10+22+\cdots+10}{n}$

let say = 38.8

Now, we find sum of squared error

$$\underbrace{(0-0)^2}_{1} + \underbrace{(0-38.8)^2 + (0-38)^2 + (10-38.8)^2 + \ldots + (10-38.8)^2}_{2 \ldots \ldots n}$$

$$= 27,468.5$$

Similarly we calculate above value for all partitions

Then choose the partition with minimum error as root of the tree, and split the tree (dosage < avg of points parted)

eg

		dosage
1	—	5
2	—	6
3	—	7
⋮		⋮
n	—	etc

if the partition is as follows,
(1 & 2 . . . n)
Then the split of tree will be (dosage < 6)

Till now, we have considered only one column
lets 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.

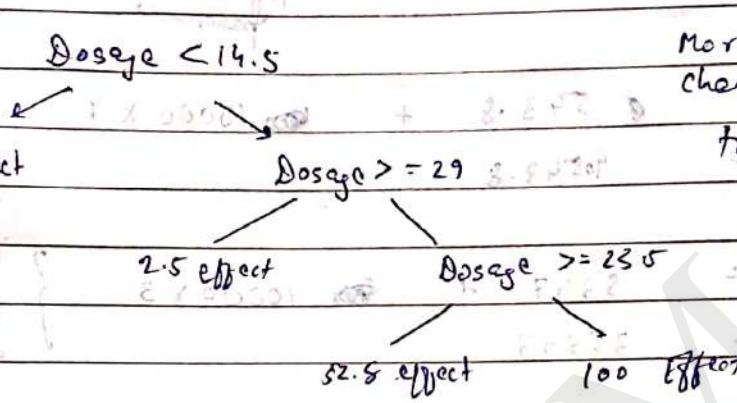
→ If we keep splitting the tree into sub nodes, we can end up overfitting on the training data

To avoid overfitting we use a technique called Pruning

Pruning :

→ Cost Complexity Pruning

Consider the following tree



More leaf nodes increase chance of overfitting to training data

Sum of squared error for < 14.5 = $(0-4.2)^2 + (0-4.2)^2 +$ + ... + (all points with dosage < 14.5 [effect] - avg effect$_{4.2}$)^2

= 320

" " " " " $>= 29$ = $(x-2.5)^2 + \cdots$

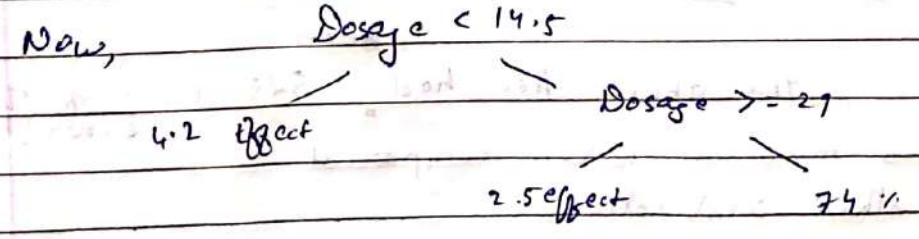
= 75

" " " " " $>= 23.5$ = 148.8

" " " " " < 23.5 = 0

→ Thus total = $320 + 75 + 148.8 + 0 = 543.8$

Now,



is more generalized

calculating total minimum squared error = 5347

Formula = Sum of squared error + α T

(α: Hyper parameter; T: no. of leaf nodes)

eg-1

= 543.8 + 10000 x 4

= 40543.8

eg-2

= 5347 + 10000 x 3

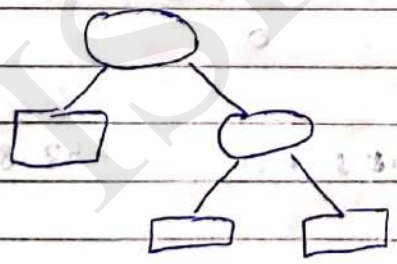
= 35347

} 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

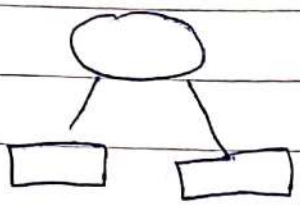
① Take the entire data, & make a tree. Now find α values at which the structure of tree changes

eg-



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

at $\alpha = 10000$



had sum of squared error 20560 ~~&~~ the above ~~tree~~ 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 crossvalidation. The tree with most occurances as the desired tree is the required tree.

* Random Forest

Steps

1. Create a bootstraped 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.

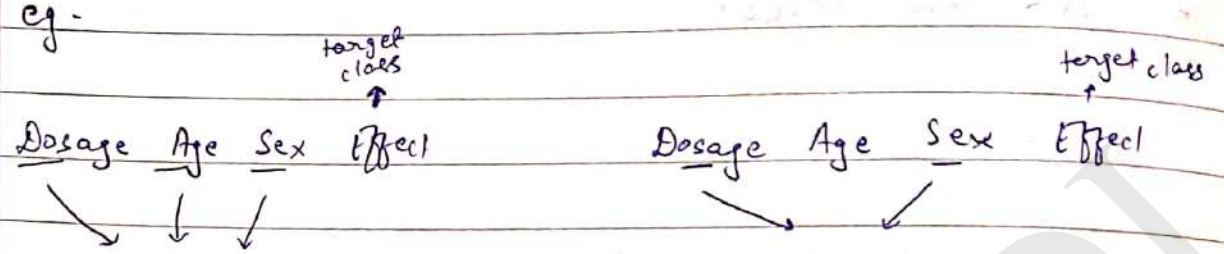
Note: Repeatation is allowed (same row can be added to the dataset) length of the bootstrapped dataset should be equal to original dataset

- Generally the all the bootstrapped dataset aggregates span over ~ 2/3 of the original data, remaining 1/3 data is called out of bag data.

→ Creating Decision Tree from ~~B~~ 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.



→ 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~~ number of times is desired output.

→ Change Parameters & go to Step 1

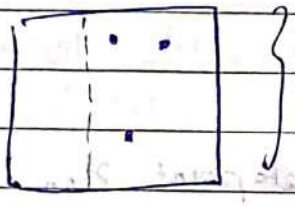
parameters like bootstrapped data coverage & number of random attributes to be selected to create tree can be changed.

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

* Boosting

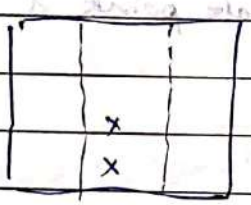
→ Decision Stump

binary tree with one root node & two child nodes

(1) 

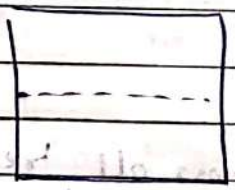
dots are miss classified

initial decision stump

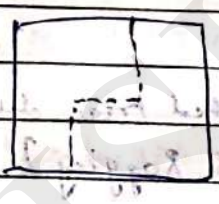
(2) 

→ decision stump such that the previous miss classification is resolved

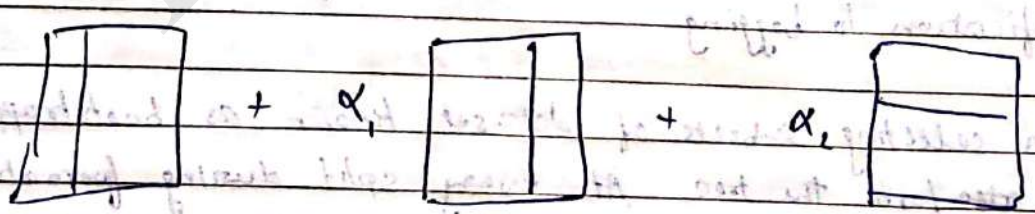
But now cross are miss classified

(3) 

All points are properly classified

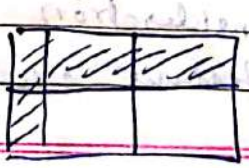
(4) 

refer to notes Boosting.png

⇒ 

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

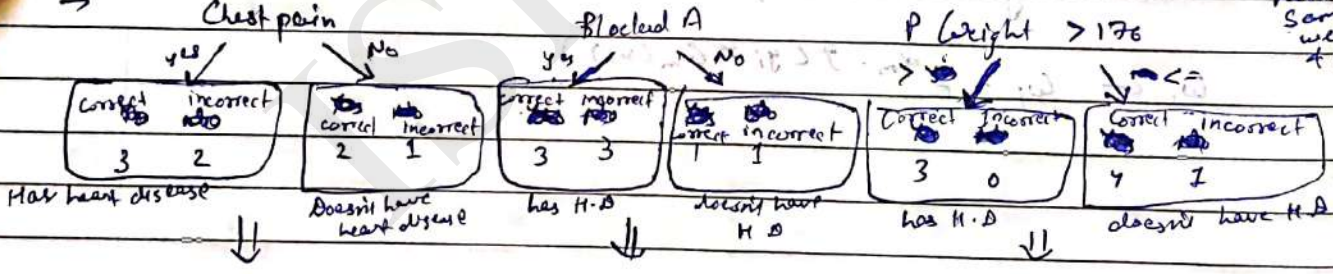
Example

	Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight (1/total no. of samples)	New sample weights	Normalised New Sample Weights (so that all sample weights add up to 1)
1	Yes	Yes	205	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	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
						0.68	1

(total no. of samples)

These now become new sample weights for next cycle

→



Gini = 0.47 Gini = 0.5 Gini = 0.2

Minimum

Thus first stump in classification

Page No.
Date

Now, error for 1st stump is

$\frac{1}{8}$ since only on value is misclassified.

ie [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 misclassified

∴ Amount of information conveyed by the stump in final output OR Amount of say $= \frac{1}{2} \log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$

∴ amount of say = 0.97 } P weight

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

→ Now lets 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 & decrease all other weights

- Formula (increase weight)

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

∴ $= 1/8 \times e^{0.97} = 0.33$

- Formula (Decrease weight)

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

$= \frac{1}{8} \times e^{-0.97} = 0.05$

→ Now second stump can be calculated by

(1) Weighted Gini index

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

$+$

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

similar for Blocked Arteries & P weight } 4 values with minimum weighted Gini index become the next split

(2)

sample	weights	Range
1	0.07	< 0.07
2	0.07	0.07 < ... < 0.14
3	0.07	0.14 < ... < 0.21
4	0.49	0.21 < ... < 0.70
5	0.07	.
6	0.07	.
7	0.07	.
8	0.07	

choose random no. between 0 & 1 ⇒ select the entry corresponding to the range in which it falls

eg ⇒ 0.4 → Make a new table & put the 4 sample in that table

Continue this process until the new table is of size same as current

reset sample weights as $1/N$ and make stump

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

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

No = $\sum_{i=1}^{2}$ 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 : $L(y_i, F(x))$ → prediction

Step 1

Height	Favourite color	Gender	Weight	$r_{i,1}$
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.

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

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

$$-\left[(88 - \text{prediction}) + (76 - \text{prediction}) + (56 - \text{prediction})\right] = 0$$

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

Step 2

(A) Residuals are calculated from latest predictions

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

residuals

$r_{i,j}$ — i: sample number, j: tree number

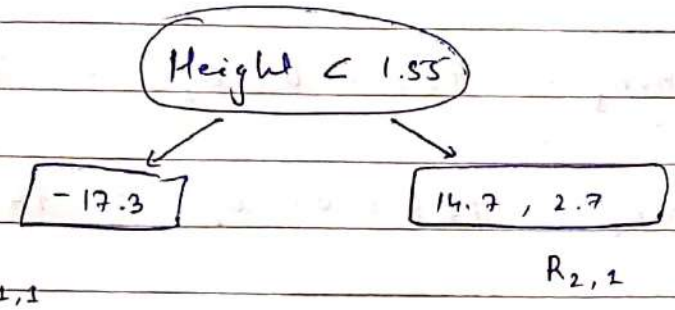
$$r_{1,1} = (y_1 - \text{prediction}) = 88 - 73.3 = 14.7$$

$$r_{2,1} = (y_2 - \text{prediction}) = 76 - 73.3 = 2.7$$

$$r_{3,1} = (y_3 - \text{prediction}) = 56 - 73.3 = -17.3$$

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



$R_{j,m}$ → Tree number
↓ index of leaf in tree

(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.

For $R_{1,1}$

$$\frac{d}{du} \sum \left(y_i - (\text{prediction} + u) \right)^2 = 0$$

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

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

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

$$u = 56 - 73.3$$

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

For $R_{2,1}$

$$2u = (88 - 73.3) + (76 - 73.3)$$

$$u_{2,1} = 8.7$$

(D) Update

$$\text{prediction}_{new} = \text{prediction}_{prev} + \eta \times \left(\text{output of the leaf node that classifies the sample}\right)$$

(η = learning rate)

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

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

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

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

Output

lets say M = 2

test ⇒ 1.4 Green Male

value = $\text{prediction}_{initial}$ +

+ 0.1 × [height < 1.555 → −17.3 , 14.7 , 2.7]

$u_{11} = -17.3$ $u_{21} = 8.7$

+ 0.1 × [height < 1.55 → −15.6 , 13.8 , 1.8]

$u_{12} = -15.6$ $u_{22} = 7.8$

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

Algorithm (Classification)

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

Loss = $L(y_i, F(x))$ → prediction

Step 1

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

Here,

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

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

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

- log (odds) = $\log\left(\frac{2}{1}\right)$ (odds of loves troll 2) or $\left(\log\left(\frac{1}{2}\right)\right.$ (odds of not loving troll 2$\left.\right)$ Not considered for this example

- $P = \dfrac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$ & $\log(\text{odds}) = \log\left(\frac{P}{1-P}\right)$

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

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

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

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

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

Now differentiating 4 equations to 0

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

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

$$p - y_i$$

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,

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

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

Step 2

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

(A) Residuals are calculated from latest prediction

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

(Sample number ←, Tree number →)

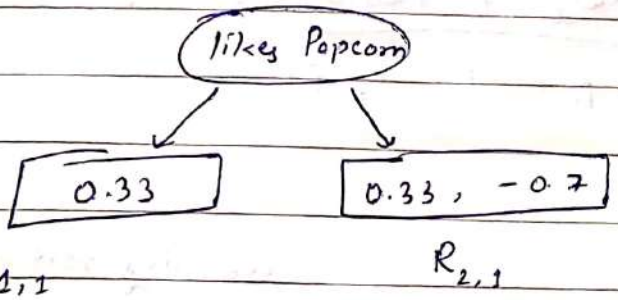
r_{21} = $(y_2 - p)$ = $1 - 0.67 = 0.33$

r_{im}

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

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

(B) Fit a classification Tree



R_{jm} → leaf node number, Tree Number

(C) Get Terminal value of leaf Node

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

For $R_{1,1}$

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

differentiating using taylor series & equating to 0

$$u = \frac{\frac{d}{du}\left(\log\left(1+e^{\log(odds)}\right) - y_i \log(odds)\right)}{\frac{d^2}{du}\left(\log\left(1+e^{\log(odds)}\right) - y_i \log(odds)\right)}$$

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

For $R_{2,1}$

Similarly,

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

(2) Update

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

(η ↑ learning rate)

$$= \log(2/1) + 0.8 \times 1.5$$

$$\log(2/1) + 0.8 \times -0.77$$

$$\log(1/2) + 0.8 \times -0.77$$

} log (odds) & thus new predictions for all samples

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

Lets say

U : set of users P : U X S → Rating

S : set of items

- learn p from data
- use p to predict utility value of each item to each user

Subcategories

(1) Content Based	(2) Collaborative Filtering
Based on previous likes & dislikes of user	Based on rating of other users which are similar to current

(2) → user based nearest neighbour
(2) → item based nearest neighbour

→ User based NN

Step (1) Use pearson's correlation coefficient

Example: lets say dataset of user & ratings is as follows

	movie 1	m2	m3	...	mm
user 1	4	3	–	. .	5
u 2	1	3	–	. .	2
u 3	–	–	3	. .	1
⋮					
u n	2	2	4	. .	4

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

u → user with which we want to compare
v → 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

v_1, v_2, v_3 which have highest similarity with u

step (2) ~~Rem~~ Recommending

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

(summation over all v's)

→ Item based NN

for above given (Example)

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

step (2) select k most nearest neighbours [k nearest j's to i's]

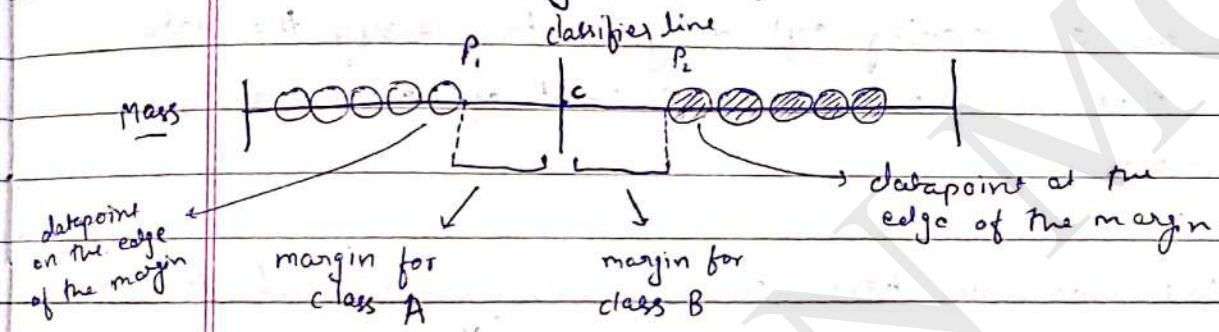
$$p(u,i) = \frac{\sum_{j \in J} r_{u,j} * \text{Sim}(i,j)}{\sum_{j \in J} \text{Sim}(i,j)}$$

* 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

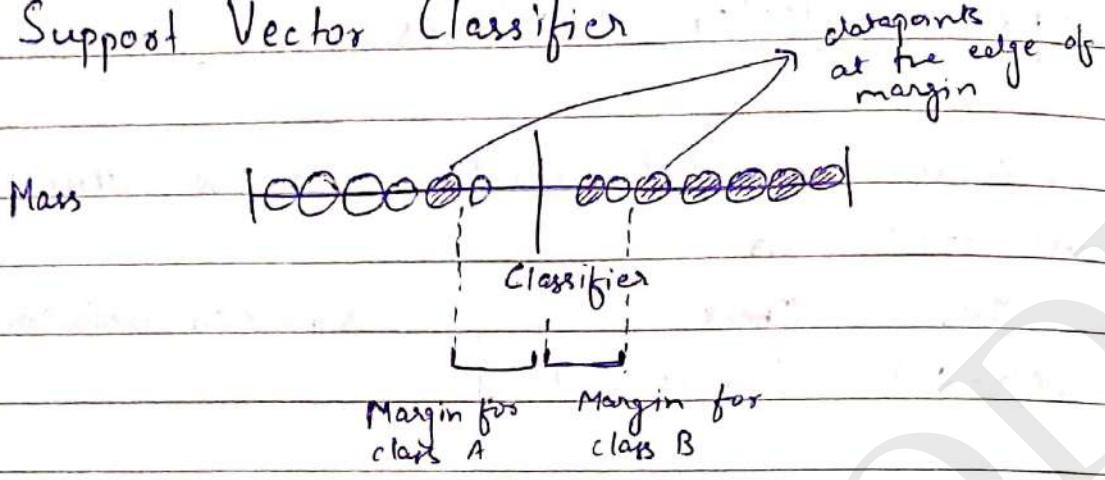


We make classifier line such that most of the points are correctly classified & the margins for target classes should be as large as possible.

Here we could have made the margin boundary between C & P_2 rather than at P_2 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 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 classification/ functionality can be obtained in a 2D space, ..., ND space as well.

→ Support Vector Machines

In the previous examples the data is linearly separable. lets assume the data is linearly non-separable. Lets assume the following data

Dosage Attribute	Cured (Yes/No) Target Class

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 x-axis. We can get all the corresponding y-axis values such that $y = x^2$.

Here we transformed the data using , but in practice there are Kernel Functions which Hep help us do that.

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

* Polynomial Kernel

Formula → $(a \times b + r)^d$ → degree of polynomial

a → datapoint 1, b → datapoint 2, r → coefficient of polynomial

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

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

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

$$= ab + a^2 b^2 + 1/4$$

$$= (a, a^2, 1/2) \cdot (b, b^2, 1/2)$$

Since 3rd dimension is same ie - 1/2 we consider only 2 dimensions

(a, a^2) & (b, b^2)

a → x-axis, a^2 → y-axis; b → x-axis, b^2 → y-axis

Similarly we can transform the entire data into a higher dimension

In practice we only need the ~~relation~~ higher dimensional relation between the datapoints. It is calculated as follows

dosage of datapoint 1 & 2

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

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 → $e^{-\gamma(a-b)^2}$ → Datapoint 2, Datapoint 1

γ → scaling coefficient

Extra Info for understanding

Now, lets take polynomial kernel with $r = 0$, $d = 1$

then it will be $(a) \cdot (b) = ab$

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

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

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

and so on..

Now, if we add all these values

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

if we break this into dot product we get

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

The datapoints are transformed to an infinite vector space

Now, for RBF kernel

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

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

Now using taylor series on 2, term

- $f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{\infty}(a)}{\infty!}(x-a)^{\infty}$

- $e^x = e^a + \frac{e^a}{1!}(x-a) + \frac{e^a}{2!}(x-a)^2 + \ldots + \frac{e^a}{\infty!}(x-a)^{\infty}$

Taking $a = 0$ & Simplifying

- $e^x = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \ldots + \frac{1}{\infty!}x^{\infty}$

Replacing x with ab

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

if we break this to dot product we get

$$\left.\begin{array}{l}\left(1, \sqrt{\frac{1}{1!}}a, \sqrt{\frac{1}{2!}}a^2 \ldots\ldots \sqrt{\frac{1}{\infty!}}a^{\infty}\right) \\ \cdot \\ \left(1, \sqrt{\frac{1}{1!}}b, \sqrt{\frac{1}{2!}}b^2 \ldots\ldots \sqrt{\frac{1}{\infty!}}b^{\infty}\right)\end{array}\right\} \text{dot product}$$

Now,

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

— let $e^{-1/2(a^2+b^2)} = s^2$

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

Thus,

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

$$\cdot$$

$$\left(s, s\sqrt{\frac{1}{1!}}b^{\bullet}, s\sqrt{\frac{1}{2!}}b^2 \ldots\ldots, s\sqrt{\frac{1}{\infty!}}b^{\infty}\right)$$

In practice, we just calculate

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