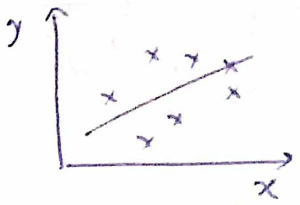
Linear Regression

Best fit Line



$\hat{y} = mx + c$

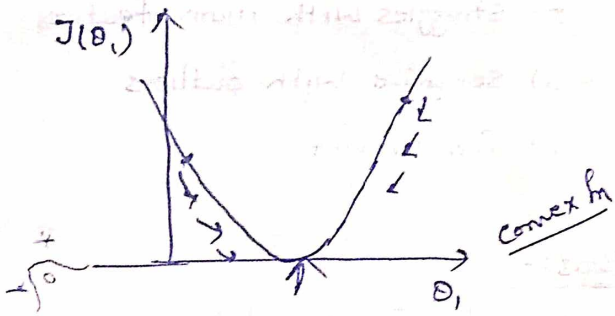
$\Rightarrow h_\theta(x_i) = \theta_0 + \theta_1 x_i$

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \theta_n x_n$

Cost fn :- (Residual err)

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{m} [h_\theta(x_i) - y_i]^2$$

$$= J(\theta_0, \theta_1)$$



$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} [J(\theta_1)]$$

- **mean squared err-**

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

⊖: 1) Not Robust to outliers
2) It changes it's unit.

- **mean Absolute Err-**

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

⊕: 1) Robust to outliers.
2) Same unit

⊖ : 1) Convergence usually takes more time

⊖: 2) optimization is a Complex proces

- **RMSE:-** $RMSE = \sqrt{MSE}$

⊕ :

⊖ :

* Ridge and Lasso Regression

overfitting

Train Accuracy - 90%
Test Accuracy - 70%
Low Bias
High variance

Under fitting

Train Accuracy = 60%
Test Accuracy = 62%
High bias
High variance

Generalized model → Train Acc - 90%
Test Acc - 89%
↳ Low Bias & Low Var.

- **Ridge Regression** (L2 Regularization)

Loss = MSE + λ (slope)^2
penalizing

↳ This prevent overfitting by Adding penalty.

- **Lasso Regression** (L1 Regularization)

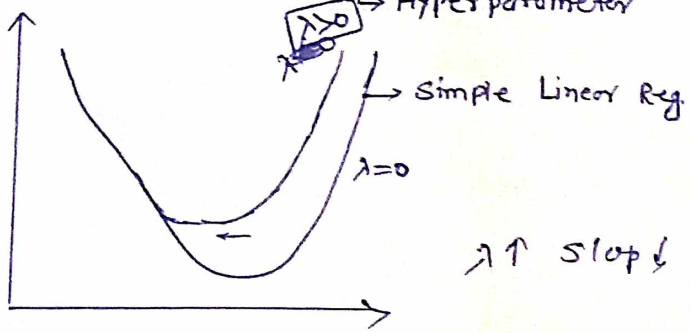
Loss = MSE + λ |slope|

→ prevent overfitting and helps in feature selection.

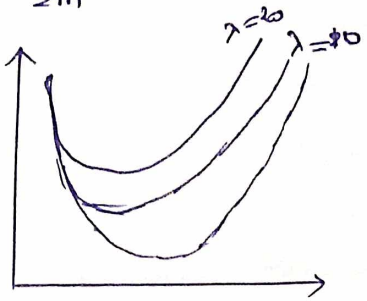
$\Rightarrow$ Loss = MSE + $\lambda \sum$ |slope$_i$|

if Cost fn (Ridge) = $\frac{1}{2m} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{n} (\text{slope})^2$

λ → Hyperparameter



if Cost fn $= \frac{1}{2m}\sum(y_i-\hat{y}_i)^2 + \lambda \sum |\text{slope}|$



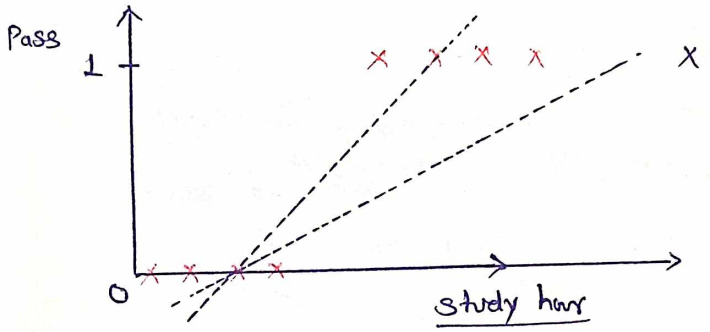
- Elastic-Net Reg.

  1) Reduce overfitting

  2) feature selection

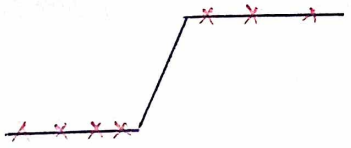
Cost fn $= \frac{1}{2m} MSE + \lambda_1 \sum (\text{slope})^2 + \lambda_2 \sum |\text{slope}|$

- Logistic Regression → Classification



Problem with Linear Regression
= | i) outliers
  | ii) sometimes >1 or <0

→ so for best fit line we have squash our line with sigmoid fn.



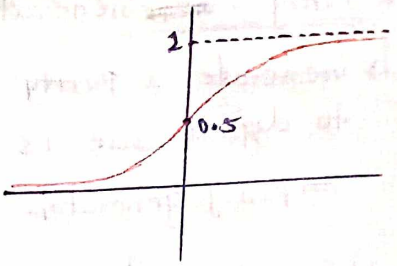
$h_\theta(x) = g(\theta_0+\theta_1 x)$

$\because \boxed{g(z) = \frac{1}{1+e^{-z}}}$

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}$$

g : Sigmoid fn

$g = \frac{1}{1+e^{-z}}$



$\because z = \theta_0+\theta_1 x$

$z = \theta^T x$

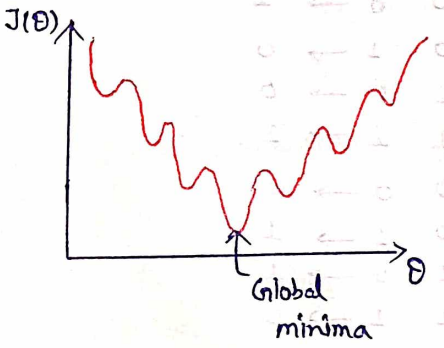
$$h_\theta(x) = \frac{1}{1+e^{-(\theta^T x)}}$$

Cost fn :

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m} [h_\theta(x_i) - y_i]^2$$

($h_\theta(x_i) \to g(\theta_0+\theta_1 x)$)

↓ Non-Convex fn



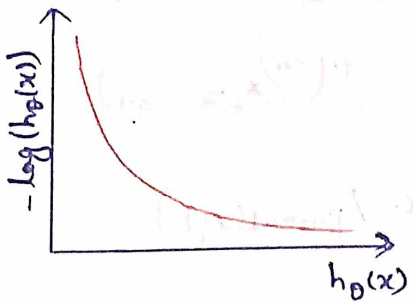
$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum [h_\theta(x_i) - y_i]^2$$

$[h_\theta(x_i) - y_i]^2 \Downarrow$ Cost $(h_\theta(x_i), y_i)$

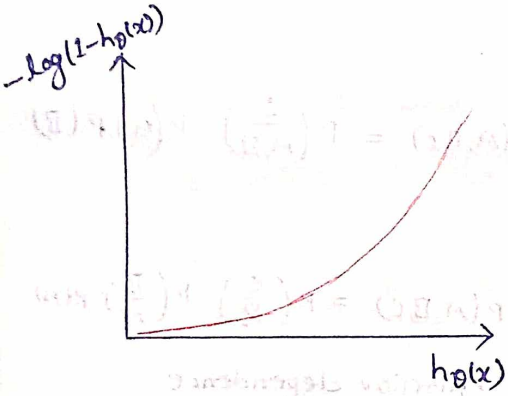
$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{if } y=1 \\ -\log(1-h_\theta(x)), & \text{if } y=0 \end{cases}$$

$$\text{Cost}(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

if $y=1$



if $h_\theta(x) = 1 \Rightarrow \text{cost} = 0$



$$J(\theta_0, \theta_1) = \frac{-1}{2m} \sum \left(y_i \log(h_\theta(x_i)) + (1-y_i) \log(1-h_\theta(x_i)) \right)$$

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

* Naive Baye's Algorithm :-

$$P(B/A) = \frac{P(B) \cdot P(A/B)}{P(A)}$$

If x_1, x_2, x_3 are Events and y is an output

$$P\left(\frac{y}{x_1 x_2 x_3}\right) = \frac{P(y) \cdot P\left(\frac{x_1 x_2 x_3}{y}\right)}{P(x_1, x_2, x_3)}$$

$$= P(y) \cdot P\left(\frac{x_1}{y}\right) \cdot P\left(\frac{x_2}{y}\right) \cdot P\left(\frac{x_3}{y}\right)$$

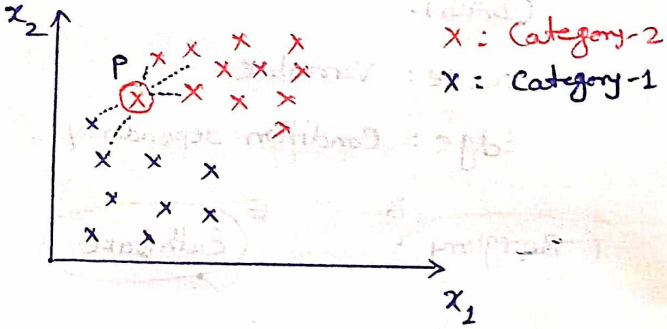
Day	Outlook	Temp	Humidity	Play Tennis
D1	Sunny	Hot	High	No
D2				
D3				
D4				

* KNN (K-Nearest Neighbor) :-

→ Used in both classification and Regression Problem.

→ It Works by finding the 'K' closest data Points (Neighbors) to a given input and makes a predictions based on the majority class (for classification) or the Average value (for Regression).

→ Lazy ~~Lea~~ Learner Algorithm because it does not learn from the training set immediately, despite ~~of~~, it stores entire dataset and perform Computation at the time of classification



→ Point P Classified to ~~Categ~~ Category-2 because ~~in~~ for K=5, 3 points of Category 2 are Nearest

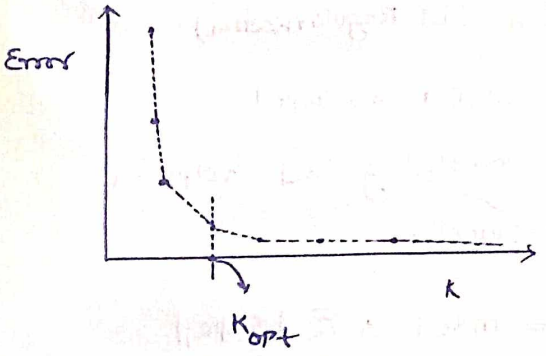
statistical method for selecting 'K'

- **Cross-Validation:-**

This method divides the dataset into K parts. The model is trained on some of these parts and tested on remaining ones. This process is repeated for each part. The K value that gives the highest avg accuracy during these tests is usually the best one to use.

- **Elbow method:-**

We draw a graph showing the error rate or accuracy for different K values. As K increases the error usually drops at first. But after a certain point error stop decreasing quickly. The point where the curve changes the direction and looks like an 'elbow' is usually best choice for K.



- **odd values of K**

To Avoid draws

* Distance metrices :

1) Euclidean distance :

$$dist(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{ij})^2}$$

2) Manhattan distance

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

3) MinKowski distance

$$d(x,y) = \left[\sum_{i=1}^{n} (x_i - y_i)^p \right]^{1/p}$$

⊖:- 1) Slow with large data
2) Struggles with many features
3) Sensitive with outliers
4) Can overfit

* Bias:-

→ Related to Training data

→ difference b/w actual or expected values and predicted values are known as Error or bias Error.

→ Let y : true value

$\hat{y}$: Estimate value

$$Bias(\hat{y}) = E(\hat{y}) - y$$

Low Bias : Training dataset Error ≈ 0

High bias : Training dataset Error is high.

↳ model will not match the training dataset closely.

*Variance :

→ Related to Test data.

→ This is the measure of spread in data from its mean position.

→ Variance is the variability of the model that how much it is sensitive to another subset of training dataset.

→ Let y : Actual value

$\hat{y}$: predicted value

$$\text{Variance} = E\left[(\hat{y} - E(\hat{y}))^2\right]$$

~~$MSE = (y - \hat{y})^2$~~

* Performance metrices:-

Dataset

x_1	x_2	y	$\hat{y}$
–	–	0	1
–	–	1	1
–	–	0	0
–	–	1	1
–	–	1	1
–	–	0	1
–	–	1	0

A) Confusion matrix :-

→ Compares the prediction made by the model with the actual results.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative
Actual Negative	False positive (FP)	True Negative

According to Exa–

Actual \ Predicted	1	0
1	3	1
0	2	1

1) Accuracy $= \dfrac{TP + TN}{TP + TN + FP + FN}$

$$Acc = \frac{4}{7} = 57\%$$

→ This can be misleading when one class is more dominant over the another.

2) Precision :- Used to minimize false positives

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

3) Recall :- used to minimize FN.

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

4) F-Beta score :-

$$= (1 + \beta^2) \frac{\text{Precision} * \text{Recall}}{\beta^2 * (\text{Precision} + \text{Recall})}$$

if FP & FN both are important

↳ $\beta = 1$

$$\text{F1 score} = \frac{2PR}{P + R}$$

if FP >> FN, FP is more important

$\beta = 0.5$

$$= [1.25] \frac{PR}{0.25(P + R)}$$

if FN >> FP

$\beta = 2$

$$= \frac{5PR}{4(P+R)}$$

5) Specificity :-

→ It measures the ability of a model to correctly identify negative instances.

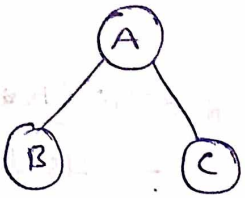
$$\text{Specificity} = \frac{TN}{TN + FP}$$

6) Type1 Error $= \frac{FP}{FP+TN}$

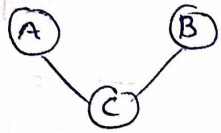
$$\text{Type2 Error} = \frac{FN}{TP+FN}$$

$$P(x_1, x_2, x_3 \ldots x_n) = P(x_1) \cdot P\left(\frac{x_2}{x_1}\right) \cdot P\left(\frac{x_3}{x_1 \cdot x_2}\right) \ldots P\left(\frac{x_n}{x_1, x_2 \ldots x_{n-1}}\right)$$

$$= \prod P\left(x_i / \text{parent}(x_i)\right)$$



$$\Rightarrow P(A,B,C) = P\left(\frac{B}{A}\right) P\left(\frac{C}{A}\right) P(A)$$



$$\Rightarrow P(A,B,C) = P\left(\frac{C}{A,B}\right) \cdot P(A)\, P(B)$$

$$\Rightarrow P(A,B,C) = P\left(\frac{C}{B}\right) \cdot P\left(\frac{B}{A}\right) \cdot P(A)$$

↳ markov dependence

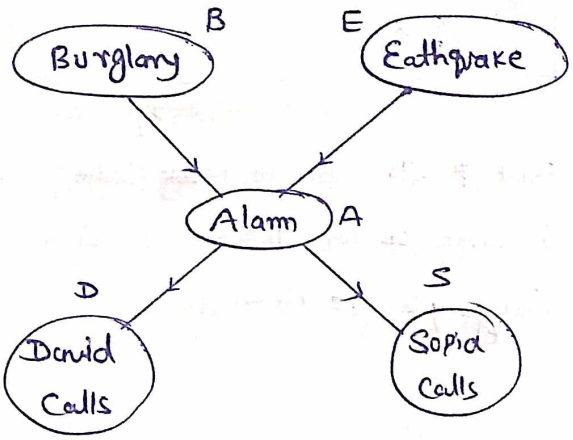
* Bayesian Belief Network (BBN)

→ Bayesian Network or Belief Network

→ It is a graphical model that represents a set of variables and their conditional dependencies using a directed acyclic graph (DAG).

Node: Variable

Edge: Condition dependency.

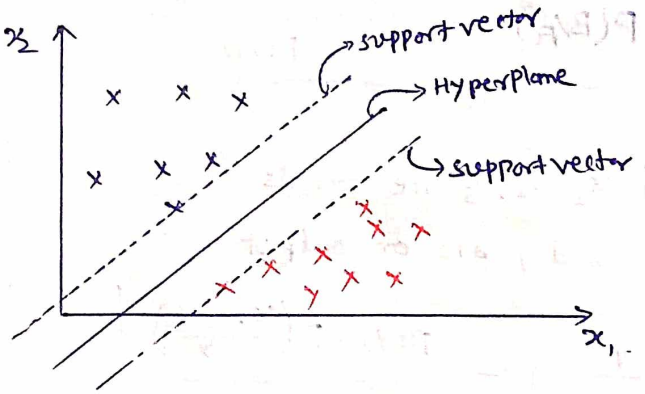


→ Each Node has Conditioned Probability Table

* Support vector machine (SVM) :-

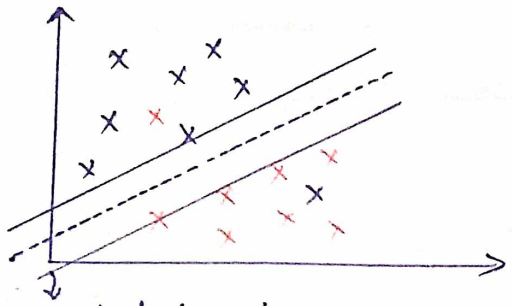
A) Support vector classifier :-

→ The key idea behind the SVM Algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them.

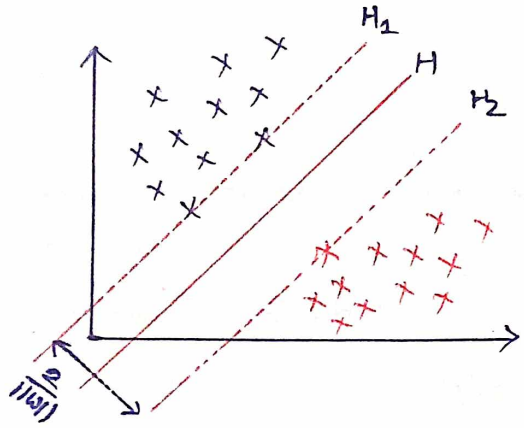


Hard margin :- A maximum margin hyperplane that perfectly separates the data without misclassification.

Soft-margin: Allows some misclassification.



marginal plane / support vector



Eqn of hyperplane –

$H: \quad W^T x + b = 0$

$H_1: \quad W^T x + b = +1$

$H_2: \quad W^T x + b = -1$

$$dist(H_1, H_2) = \frac{2}{||w||}$$

We have to maximize marginal plane distance ⇒ minimize $||w||$

$$\because \quad y_i = \begin{cases} +1, & w^T x + b \geq 1 \\ -1, & w^T x + b \leq -1 \end{cases}$$

for all accurate datapoint

$$y * (w^T x + b) \geq 01$$

$$\text{Hinge loss} = \max(0, 1 - y_i (w^T x_i + b))$$

$$\text{Cost fn} = \frac{1}{2} ||w|| + C \left[\frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i (w^T x_i + b)) \right]$$

$$\Rightarrow \underset{w,b,\zeta}{\text{Minimize}} \quad \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} \zeta_i$$

No. of misclassified point

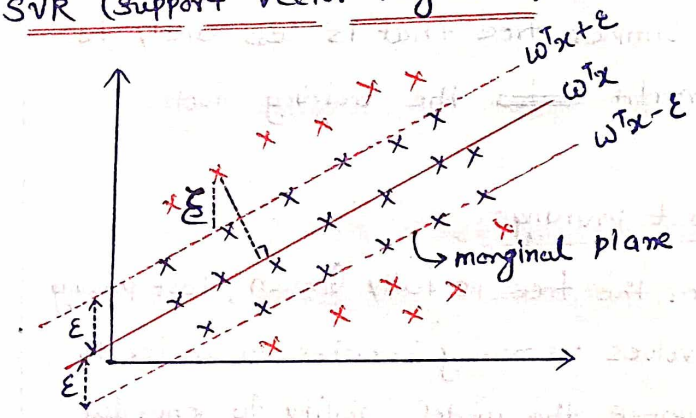
$$\Rightarrow \quad y_i (w^T x_i - b) \geq 1 - \zeta_i$$

$$\Rightarrow \quad \zeta_i \geq 0$$

C: This is a regularization parameter that controls the trade-off b/w margin maximization and penalty for misclassificatio

ζ: It is a slack variable that represent the degree of violation of the margin by each data point.

* **SVR (Support vector Regression):-**



∵ We have to minimize – $||w||$

And – $|y_i - w^T x_i| \leq \varepsilon$

$|y_i - \hat{y}| \leq \varepsilon$

for All Points –

$$|y_i - w^T x_i| \leq \varepsilon + |\varepsilon_i|$$

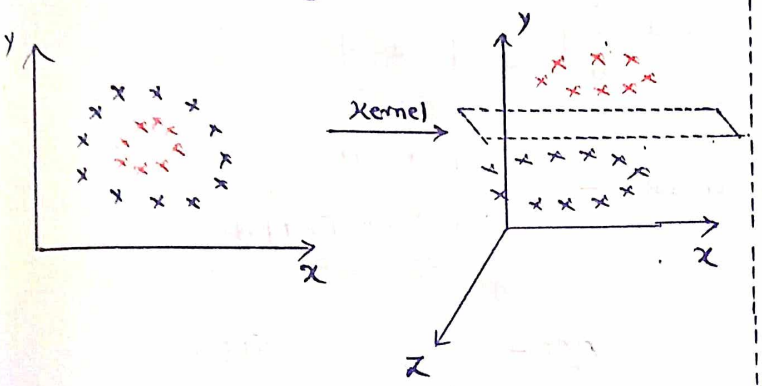
$$\text{Cost fn} = \underset{(w,b)}{\text{Minimize}} \quad \frac{1}{2} ||w||^2 + C \sum_{i=1}^{n} |\varepsilon_i|$$

Hinge loss

OR Draw an ER diagram of Railway Management System

SVM Kernels :-

→ A Kernel is a function that transforms the data into a higher dimensional space where it becomes easier to separate using hyperplane.



Why use Kernels :-

1) Non-Linearity handling
2) flexibility : different Kernels can be used depending upon the nature of the data and the problem.
3) feature Extraction

→ for Transformed feature $\phi(x)$

$$x_i \longrightarrow \phi(x_i)$$

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

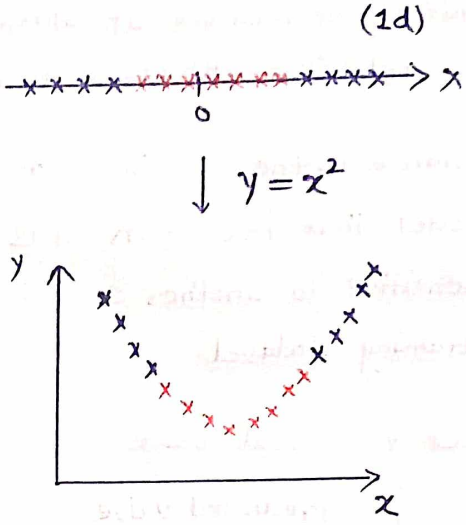
Common SVM Kernels-

1) Linear : $K(x_i, x_j) = x_i \cdot x_j$
2) Polynomial: $K(x_i, x_j) = (\gamma x_i \cdot x_j + c)^d$
3) Gaussian or Radia Basis-

$$K(x_i, x_j) = e^{(-\gamma \|x_i - x_j\|^2)}$$

4) Sigmoid : $K(x_i, x_j) = \tanh(\gamma x_i x_j + c)$

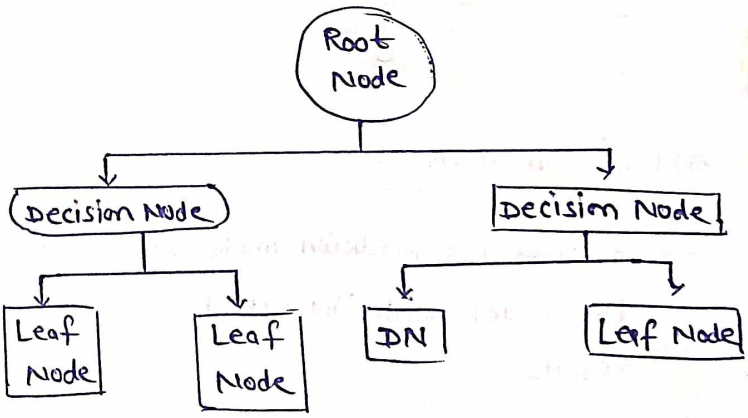
Exq -



* Decision Tree :-

→ A decision Tree is a supervised learning algorithm used for both classification and Regression tasks. It has a hierarchical tree structure which consists of root node, branches, internal Nodes and Leaf Nodes.

- Internal Nodes represent attribute test.
- Branches represent attribute values.
- Leaf Nodes represent final decisions or predictions.

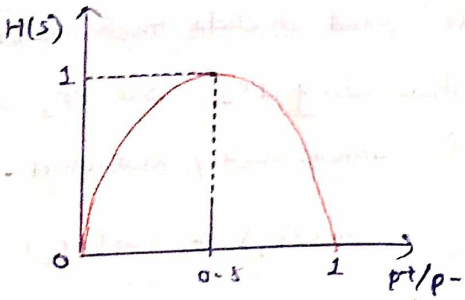


- **Entropy**: This is the measure of uncertainty of a random variable.

→ It characterizes the impurity of an arbitary collection of examples.

→ The higher the Entropy more the information Content.

$$H(S) = -P_+ \log_2(P_+) - P_- \log_2(P_-)$$

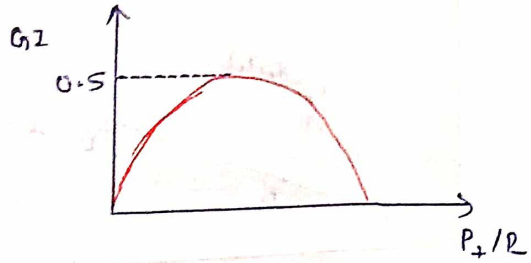


- **Gini Index**:-

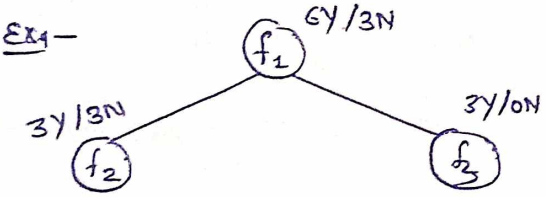
→ This is a metric to measure how often a randomly chosen element would be incorrectly identified.

→ Attribute with Lower Gini Index should be preffered.

$$G\text{-}I = 1 - \sum_{i=1}^{n} P_i^2$$



Ex1 –



$$H(f_3) = -\frac{3}{3}\log_2\left(\frac{3}{3}\right) - \frac{0}{3}\log_2\left(\frac{0}{3}\right)$$

$$= -1\log_2(1) = \underline{0} \quad (\text{Pure split})$$

$$H(f_2) = -\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right)$$

$$= -\log_2\left(\frac{1}{2}\right) = \underline{1} \quad (\text{Impure split})$$

$$GI(f_2) = 1 - [P_+^2 + P_-^2]$$

$$= 1 - (0.5^2 + 0.5^2) = 0.5$$

$$Gz(f_3) = 1$$

- **Information Gain**:-

→ Information Gain tells us how useful a question (or feature) is for splitting data into the groups.

→ measures how much uncertainty decreases after split.

→ if S is a set of Instances, A is an attribute.

$$Gain(S,A) = \underbrace{H(S)}_{\text{Entropy of Root Node}} - \sum \frac{|S_v|}{|S|} \cdot H(S_v)$$

- **Pruning in Decision Tree**:

→ Decision Pruning is a technique used to prevent decision trees from overfitting the training data

→ We remove unwanted Nodes.

1) Pre-Pruning:-

→ We set max depth to limit the decision Tree depth.

→ Set a minimum threshold for the Number of Samples in each leaf node. (minimum Samples per leaf)

→ Specify the minimal Number of Samples needed to break up a node. (minimum Samples per Split)

→ Restrict the quantity of features Considered for splitting.

→ Maximum depth.

- By Pruning Early, We Come to be with a simpler tree that is less likely to overfit the training facts.

2) Post pruning:

- After the tree is fully Grown, Post pruning Involves removing branches or nodes to Improve the model's ability to generalize.

→ Cost Complexity Pruning

→ Reduced Error pruning

→ minimum Impurity Decrease

→ minimum leaf size

- This Simplifies the tree while Preserving it's Accuracy.

* PCA (Principle Component Analysis)

→ This is a dimensionality reduction technique used in data Analysis and machine learning.

→ It helps you to reduce the Number of features in a dataset While Keeping the most Important Information.

because more Number of features after a limit degrades the performance of model.

3 feature	→	Accuracy = 60%
4 feature	→	Accuray = 80%
5 feature	→	Accuray = 90%
7 feature	→	Accuray = 85%
9 feature	→	Accuray = 70%

→ 2 Ways to remove Curse of dimensionality-

1) feature selection
2) Dimensionality Reduction (PCA)

→ We select only Important features.

→ Why dimensionality Reduction -

1) Prevent → Curse of dimensionality
2) Improve the performance of model
3) visualize the data and understand the data.

→ To see How features relate to each other Wheather they increase or decrease togeather We use Covarience -

$$Cov(x,y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

if $Cov(x,y) = (+)ive \Rightarrow x \uparrow \; y \uparrow$

$Cov(x,y) = (-)ive \Rightarrow x \uparrow \; y \downarrow$

if $Cov(x,y) = 0 \Rightarrow$ No relation bw x, y

• Pearson Co-relation $= \frac{Cov(x,y)}{\sigma_x \sigma_y}$

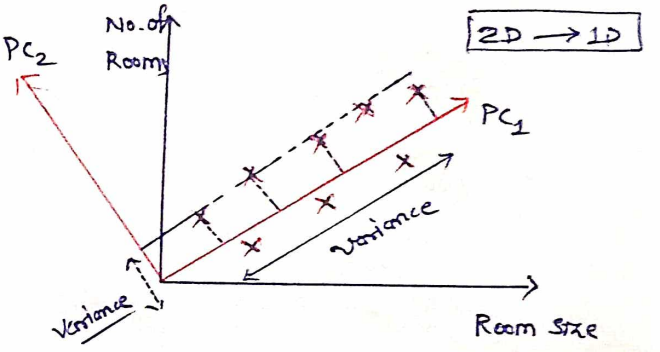
$-1 \leq$ Co-relation ≤ 1

if $\frac{Cov(x,y)}{\sigma_x \sigma_y} \longrightarrow 1 \Rightarrow$ more (+)ive Correlated

$\frac{Cov(x,y)}{\sigma_x \sigma_y} \longrightarrow -1 \Rightarrow$ more (-)ive Co-related

$\frac{Cov(x,y)}{\sigma_x \sigma_y} \longrightarrow 0 \Rightarrow$ No Corelation

→ PCA Uses linear Algebra to transform data into new features Called principal Components. It finds these by Calculating Eigen vectors (direction) and Eigen values (importance) from the Covarience matrix.



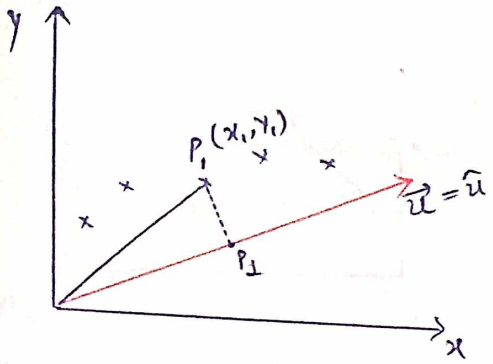
→ PCA identifies the new directions: Pc1, Pc2.

→ Pc1 Captures the maximum variance meaning it holds the most information.

Pc_2 Captures the remaining Variance and is Perpendicular to Pc_1.

→ The spread of data much wider along PC_1 than along PC_2. So PC_1 is chosen for dimensionality Reduction.

$$Var(PC_1) > var(PC_2)$$



$$Proj_{P_1}(u) = \frac{P_1 \cdot u}{\|u\|}$$

$\because \|u\| = 1$

$$Proj_{P_1}(u) = P_1 \cdot u$$

$\downarrow$

P_1'

We get $\{P_0', P_1', P_2', \dots P_n'\}$

$\Downarrow$

$\{x_1', x_2', x_3', \dots x_n'\}$

$$\text{Max variance} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

↳ Cost fn

Eigen vector or Eigen values -

1) find Covariance matrix between feature
2) Eigen vector and Eigen value will be found out from Covariance matrix
3) $AV = \lambda V$
3) Eigen vector with high Eigen value should be chosen.

→ for 2 features - (x, y)

Covariance matrix -

$$A = \begin{array}{c|cc} & x & y \\ \hline x & Var(x) & Cov(x,y) \\ y & Cov(x,y) & Var(y) \end{array}$$

for 3 features -

$$A = \begin{bmatrix} Var(x) & Cov(x,y) & Cov(x,z) \\ Cov(x,y) & Var(y) & Cov(y,z) \\ Cov(x,z) & Cov(y,z) & Var(z) \end{bmatrix}$$

↳ $AV = \lambda V$

↳ find λ

$\lambda_1 \quad \lambda_2 \quad \lambda_3$

$\downarrow \quad \downarrow \quad \downarrow$

$PC_1 \quad PC_2 \quad PC_3$