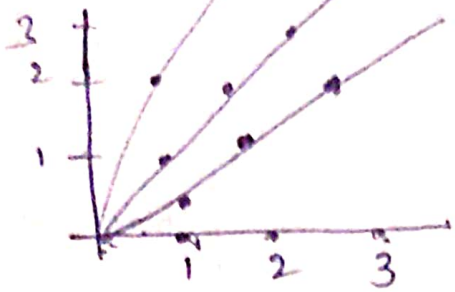
ML

① Linear Regression

$\hat{y} = \beta_0 + \beta_1 x$ ($\beta_0 \downarrow 0$)

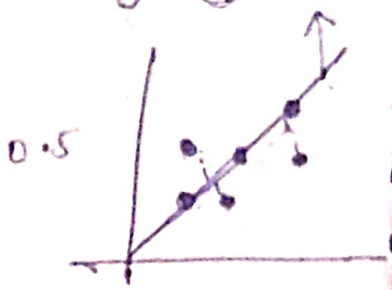
$\hat{y} = \beta_1 x$

$\beta_1 = 1$



$\beta_0 = 0$

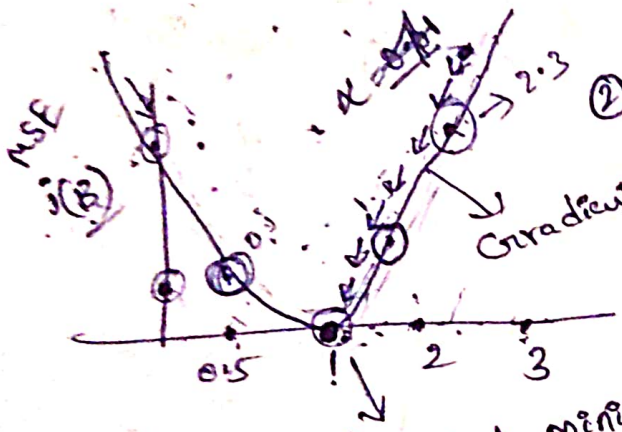
$\hat{y} = \beta_0 + \beta_1 x$



① $J(\beta_1) = \frac{1}{2m} \sum_{i=1}^{3} (\hat{y}_i - y_i)^2$

$= \frac{1}{2m} (1-1)^2 + (2-2)^2 + (3-3)^2 = \frac{0}{2m} = 0$

0.01



② $J(\beta_1 = 0.5) = \frac{1}{2m} (0.5-1)^2 + (1-2)^2 + (1.5-3)^2$

$= \frac{1}{2m} (0.25 + 1 + 2.25$

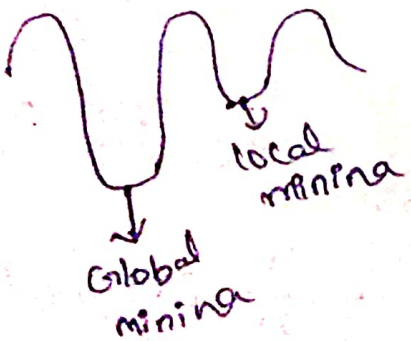
$= \frac{3.5}{6} = 0.58$

α.

③ $J(\beta_1 = 2) = \frac{1}{2m} (2-1)^2 + (4-2)^2 + (6-3)^2$

$= \frac{1}{m} \frac{1+4+9}{2}$

$= \frac{14}{6} = \frac{7}{3} = 2.3$



$\theta_j = \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} \frac{dj}{d\theta_j}$

Gradient Descent for Linear Regression

$$\beta_0 = \beta_0 - \alpha \cdot \frac{\partial}{\partial \beta_0}\left(\frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2\right)$$

$$= \beta_0 - \alpha \cdot \frac{\partial}{\partial \beta_0}\left(\frac{1}{2m}\sum_{i=1}^{m}(\beta_0 + \beta_1 x - y_i)^2\right)$$

$$= \beta_0 - \alpha \cdot \left(\frac{1}{2m} \cancel{2}\,(\beta_0 + \beta_1 x - y_i)\right)$$

$$= \beta_0 - \alpha \cdot \frac{1}{m}(\hat{y}_i - y_i)$$

$$\frac{\partial}{\partial x}(ax+b)^2 = 2(ax+b)\cdot\frac{\partial ax}{\partial x}$$

$$\beta_1 = \beta_0 - \alpha \cdot \frac{\partial}{\partial \beta_1}\left(\frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2\right)$$

$$= \beta_0 - \alpha \cdot \frac{\partial}{\partial \beta_1}\left(\frac{1}{2m}\sum_{i=1}^{m}(\beta_0 + \beta_1 x - y_i)^2\right)$$

$$= \beta_0 - \alpha\left(\frac{1}{2m} \cancel{2}\,(\beta_0 + \beta_1 x - y_i)\cdot\frac{\partial \beta_1 x}{\partial \beta_1}\right.$$

$$= \beta_0 - \alpha \cdot \frac{1}{m}(\hat{y}_i - y_i)\cdot x_i$$

$$\beta_j = \beta_j - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)\cdot x_j^{(i)}$$

for all $j \in \{1, 2, 3, \ldots n\}$

$\alpha = 1/10^n$

0.1

② Logistic Regression

Cost function

$$J(\beta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y_i \log(\hat{y_i}) + (1-y_i) \log(1-\hat{y_i}) \right]$$

$y_i \in \{0, 1\}$

$$\hat{y_i} = h_\beta(x_i) = \frac{1}{1+e^{-x_i^T \beta}} \quad \text{(Predicted probability)}$$

Where, m = no. of training observations

y_i = Actual class label (0 or 1)

$\hat{y_i}$ = Predicted probability

β = model parameter.

likelihood function.

$$P(y_i / \hat{y_i}) = (\hat{y_i})^{y_i} * (1-\hat{y_i})^{1-y_i}$$

* $y_i = 1 \rightarrow \hat{y_i}$
* $y_i = 0 \rightarrow 1-\hat{y_i}$

Log. likelihood Function.

$$L(\theta) = \prod_{i=1}^{n} P(y_i / \hat{y_i})$$

$\because (\log(a^b) = b \log a)$

$(\log(ab) = \log a + \log b)$

Taking log on both sides,

$$\Rightarrow \log L(\theta) = \log \left[\prod_{i=1}^{n} (\hat{y}_i)^{y_i} \cdot (1-\hat{y}_i)^{(1-y_i)} \right]$$

$$\Rightarrow \log L(\theta) = \sum_{i=1}^{n} \left[\log (\hat{y}_i)^{y_i} + \log (1-\hat{y}_i)^{(1-y_i)} \right]$$

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

Negative log-likelihood,

$$J(\beta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y_i \log(\hat{y}_i) + (1-y_i) \log (1-\hat{y}_i) \right]$$

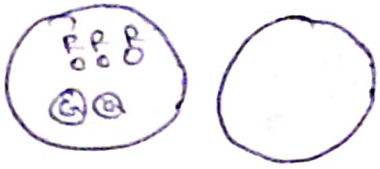
↓

Gradient descent optimization

⇓

$$\theta \beta_j = \beta_j - \alpha \frac{\partial J}{\partial \beta_j}$$

(3) Naive Bayes ⇒ ~~Binary~~ classification



$P(A \text{ and } B) = P(A) * P(B/A)$

$P(B \text{ and } A) = P(B) * P(A/B)$

$P(R) = 3/5$

$P(G) = 2/4$

Equalling.

$P(A) * P(B/A) = P(B) * P(A/B)$

$$P(B/A) = \frac{P(B) * P(A/B)}{P(A)} \rightarrow \text{Conditional probability.}$$

⇓ Baye's Theorem.

Naive Baye's classifier.

$x_1, x_2, x_3, \ldots x_n$ — Indepent var

y — dependent var ⇒ $y=1$ Yes, $y=0$ NO

$$P(y/(x_1, x_2, \ldots x_n)) = \frac{P(y) * P((x_1, x_2, x_3 \ldots x_n)/y)}{P(x_1, x_2, \ldots x_n)}$$

$$= \frac{P(y) * P(x_1/y) * P(x_2/y) * P(x_3/y) * \ldots * P(x_n/y)}{P(x_1) * P(x_2) * P(x_3) * \ldots * P(x_n)}$$

Constant

$$P(y=NO/x_1, x_2 \ldots x_n) = \frac{P(N) * P(x_1/N) * P(x_2/N) * \ldots * P(x_n/N)}{P(x_1) * P(x_2) * P(x_n)}$$

Constant — # fixed

$P(y=yes/x_i) = 0.13$ $\quad P(y=No/x_i) = 0.05$

⇓

Need to do Normalization.

$\geq 0.5 \Rightarrow$ class 1

$< 0.5 \Rightarrow$ class 0.

$$P(y=yes/x_i) = \frac{0.13}{0.13+0.05} = 0.72 = 72\%$$

$$P(y=No/x_i) = 1-0.72 = 0.28 = 28\%.$$

Normalized.

$$P(y/x_i) = P(y) * \prod_{i=1}^{n} P(x_i/y) \qquad y = C_k \text{ } \} \text{ class labels}$$

$$= P(C_k) * \prod_{i=1}^{n} P(x_i/y)$$

$$P(C_k/x_i) = P(C_k) * \prod_{i=1}^{n} P\left(\frac{x_i}{C_k}\right)$$

- $P(C_k/x_i)$ ↓ Pro of class given input features
- $P(C_k)$ ↓ Probability of class C_k
- $P\left(\frac{x_i}{C_k}\right)$ ↓ Pro of i^{th} feature x_i given by class C_k

$$P(x_i/C_k) = \prod_{i=1}^{n} P(x_i/C_k) \Rightarrow \text{naive Assumption}$$

(4) KNN Algorithm (used for both)

classification

Regression.

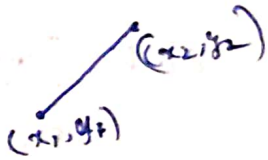
K nearest neighbours

K=5 => 5 neighbours => K is hyperparameter

* New point goes to which class decision taken by distance.

Eucledian distance

(x_1, y_1) (x_2, y_2) $\Rightarrow \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$



Manhatten distance

$|(x_2-x_1)+(y_2-y_1)|$

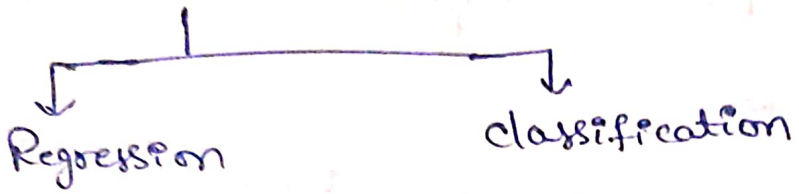
Regression

K Nearest work bad with

* Outliers
* Imbalanced dataset.

---

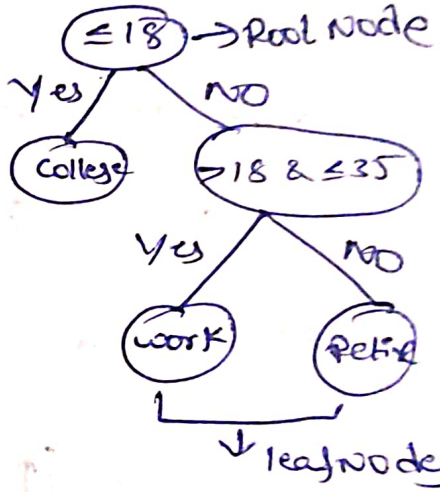
(5) Decision Tree (used for both)



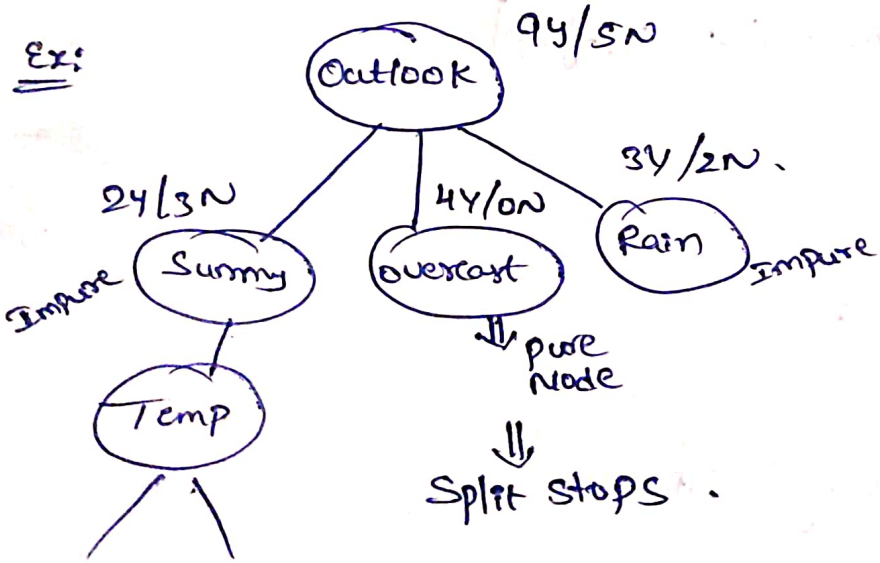

```
if (age ≤ 18):
    print ("college")
elif (age > 18 & age ≤ 35):
    print ("work")
Else:
    print (retire)
```

⇒

Decision Tree



Ex:



⇓ Split stops.

Pure split ✓ zero NO's

Impure split. ~~✓~~ Yes.

⇒ until pure node i.e leaf node comes this process repeats.

① How we Calculate Purity ?

① Entropy

② Gini Coefficient (or) Gini Impurity

② How next node or root node features are selected.

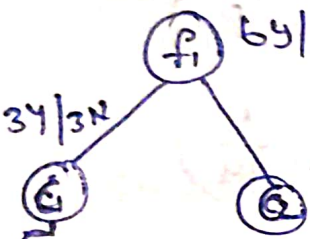
① Information Gain.

Entropy

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

Gini Impurity

$$G.I = 1 - \sum_{i=1}^{n} (P)^2$$



P_+ = probability of Yes.

P_- = probability of NO.

Entropy ⇓

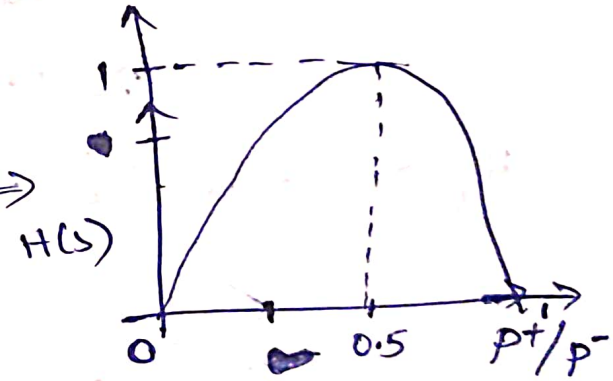
$$H(S) = -\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3}$$

$$= -1 \log_2 1 = 0$$

$$\Rightarrow H(S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$$

$$= 1$$

* Entropy is Betwee 0 & 1.



$p^+ = 0.5$

$p^- = 0.5$

⇓

$H(S) = 1$.

⇓

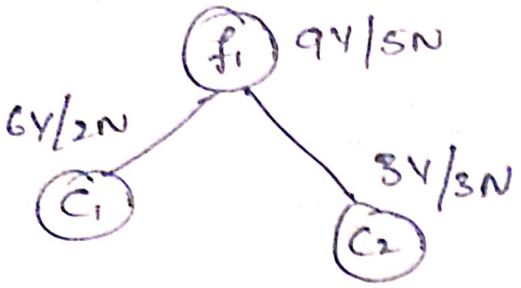
Impure split.

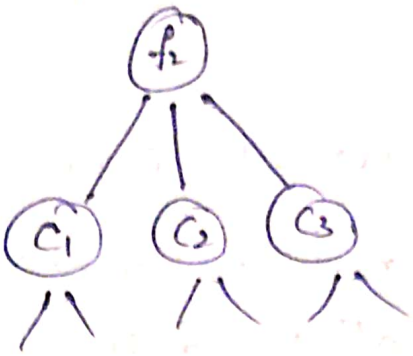
$H(S) = 0$

⇓

Pure split.

② feature selection - to Split:





Information Gain

$$Gain(S, f_1) = H(S) - \sum_{v \in val} \frac{|S_v|}{|S|} H(S_v)$$

Entropy of root node $= -P_+ \log_2 P_+ - P_- \log_2 (P_-)$

$$= -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right)$$

$$\approx 0.94$$

$$H(S_v) \Rightarrow H(S_{v_1}) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8}$$

$$\approx 0.81$$

$$\Rightarrow H(S_{v_2}) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{8} \log_2 \frac{3}{6} = 1.$$

$$Gain(S, f_1) = 0.94 - \left[\frac{8}{14} * 0.8 + \frac{6}{14} * 1\right]$$

$$= 0.049.$$

$Gain(S, f_2) = 0.051 \Rightarrow$ Select this.

$Gain(S, f_2) > Gain(S, f_1)$ ✓

Gini Impurity

$$Gini = 1 - \sum_{i=1}^{n} (p)^2$$

$$= 1 - \left[(p_+)^2 + (p_-)^2 \right]$$

$$= 1 - \left[\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 \right] = 1 - \left[\frac{1}{2}\right] = 0.5$$

2Y/2N → Complete → Impure split.

⇓

but in Entropy = 1.

* Entropy takes more time to compute because of log.

⇓

So, when Decision Tree is more complex

⇓

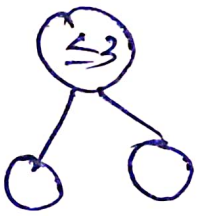
Use Gini Impurity.

Fast

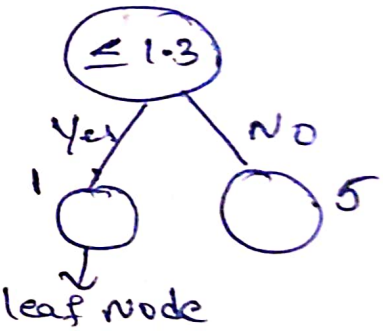
Gini > Entropy.

Numerical Features.

f_1	o/p	⇒	f_1 (sorted)
2.3			1.3
1.3			2.3
4			3
5			4
7			5
3			7



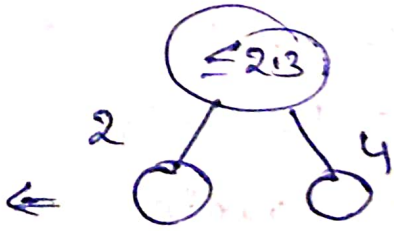
.....



leaf node

⇓

Information Gain computed ≤1.3



⇐ ⇒ IG Computed

⇓

After all this for which root node IG is high that is Selected.

Decision Tree Regressor

f_1 f_2 o/p

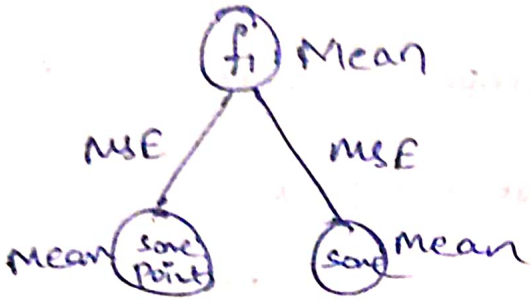
continuous → Mean Calculated

Mean

⇓

MSE (or) MAE

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



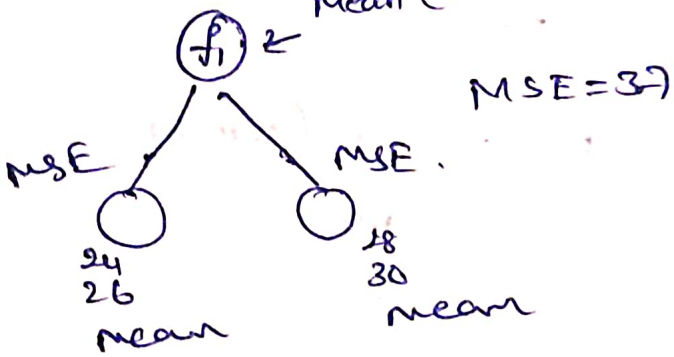
MSE reducing

⇓

going to reach leaf node.

~~Hyper~~ ~~Parameters~~

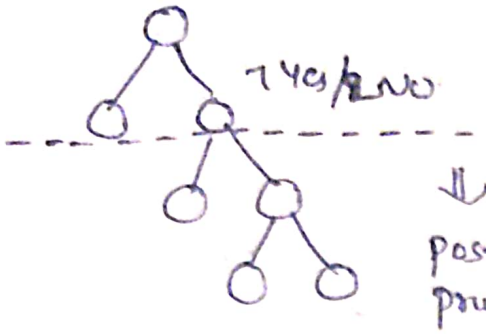
f_1	o/p
c_1	20
c_2	24
–	26
–	28
–	30



Hyperparameter

Decision Tree ⇒ Overfitting to avoid.

1. post pruning
2. pre pruning



Yes/No ⇒ 7 Yes there (more)

⇓ may chance Answer is Yes

⇓ So we stop

⇓ Post pruning

Pre pruning done by GridsearchCV

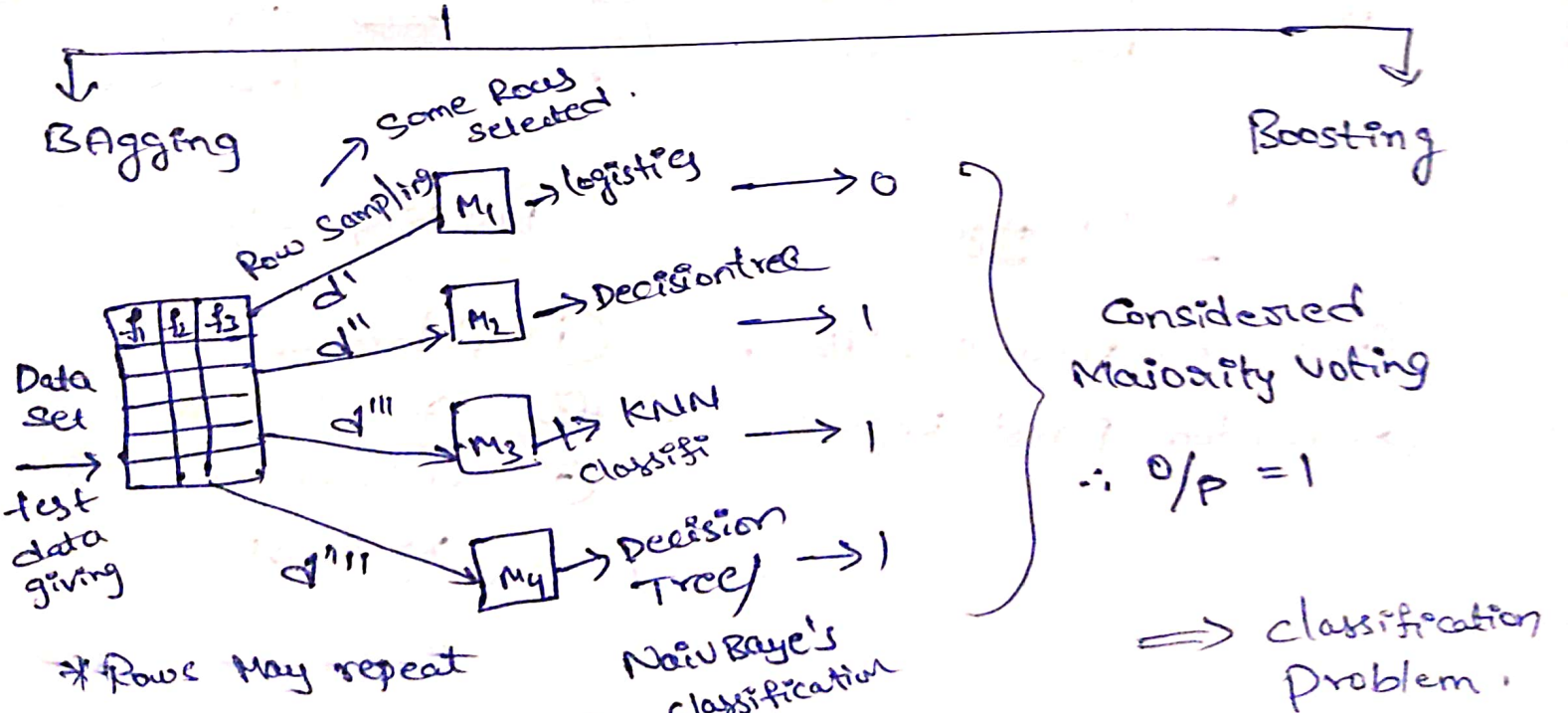
⇓

Based on max_depth, max_leaf

Ensemble Techniques.

⇓

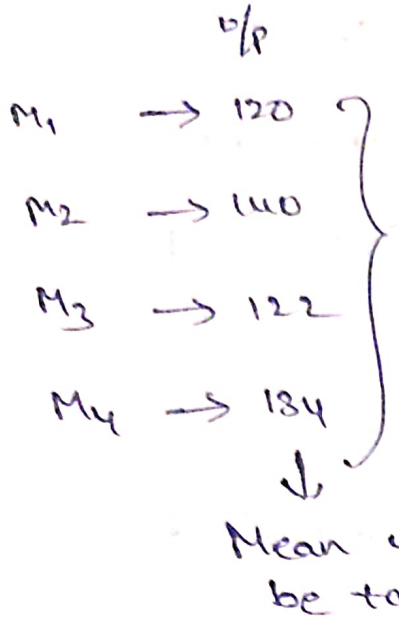
Used to Solve ~~Multip~~ a problem Using Multiple Algorithms.



⇒ classification Problem.

⇓

Bootstrap Aggregating.



o/p = Mean.

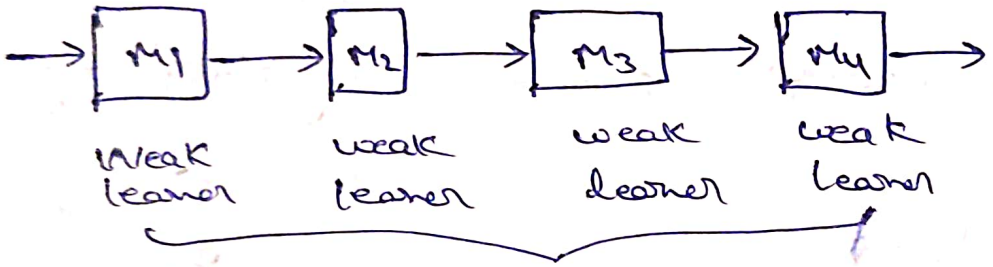
⇒ In Baggain

↓

All are parallel running Models.

Boosting

⇒ Sequential Models Combined to give strong learner.



Bagging Algo

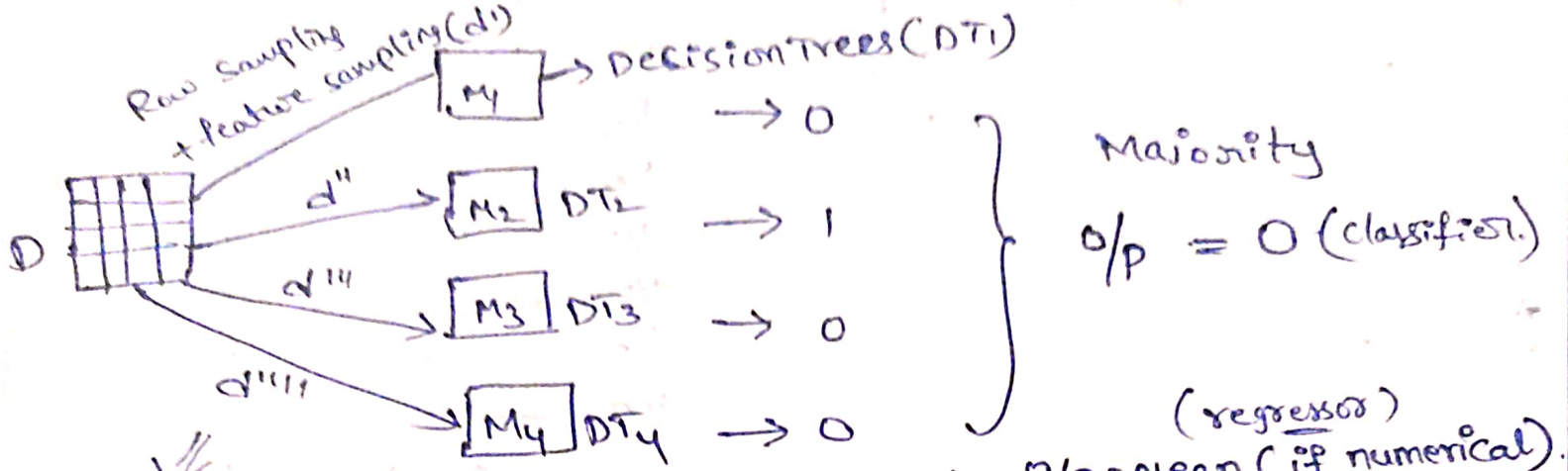
1. Random Forest classifier
2. Random Forest Regressor

Boosting Algo

1. AdaBoost
2. Gradient Boost
3. Extreme gradient xg Boost.

(6) Random Forest classifier & Regressor.

→ Bagging Technique ⇒ All Models are Decision Trees



In Each Sub dataset row and Features may Repeat.

⇓

- this all may be overfitting

⇒ Random Forest leads to Avoid overfitting

i.e low Bias
low variance

⇓ O/p model.

* Normalization Not required in Decision Tree or random forest
* Standardization needed in KNN for distances.
* Random Forest not impacted by Outliers.
* KNN impacted by Outliers.

(2) Boosting

i) AdaBoost

	f_1	f_2	f_3	f_4	O/P
	–	–	–	–	Yes
	–	–	–	–	NO
	–	–	–	–	–
7 observations	x –	–	–	–	–
	–	–	–	–	–
	–	–	–	–	–
	–	–	–	–	–

Define weight		Updated	Normalized weight
1/7	→	0.05	0.07
1/7	→	0.05	0.07
1/7	→	0.05	0.07
1/7	→	0.349	0.537
1/7	→	0.05	0.07
1/7	→	0.05	0.07
1/7	→	0.05	0.07
		0.649	⇓

(f_1) (f_2) (f_3)

Buckets

[0 - 0.07]
[0.07 - 0.14]
[0.14 - 0.21]
✓ [0.21 - 0.7497] → diff 0.349
[0.7497 - 0.751]

1 wrong

↕ one level
depth = 1
⇓
Called Stumps
⇓
Weak learner] one level decision tree

Correct Records

(3) New Sample weight = weight of e^{-PS}

$= \frac{1}{7} \times e^{-0.895} = 0.05$

Incorrect recors = weight * e^{PS}

$= \frac{1}{7} * e^{0.895} = 0.349$

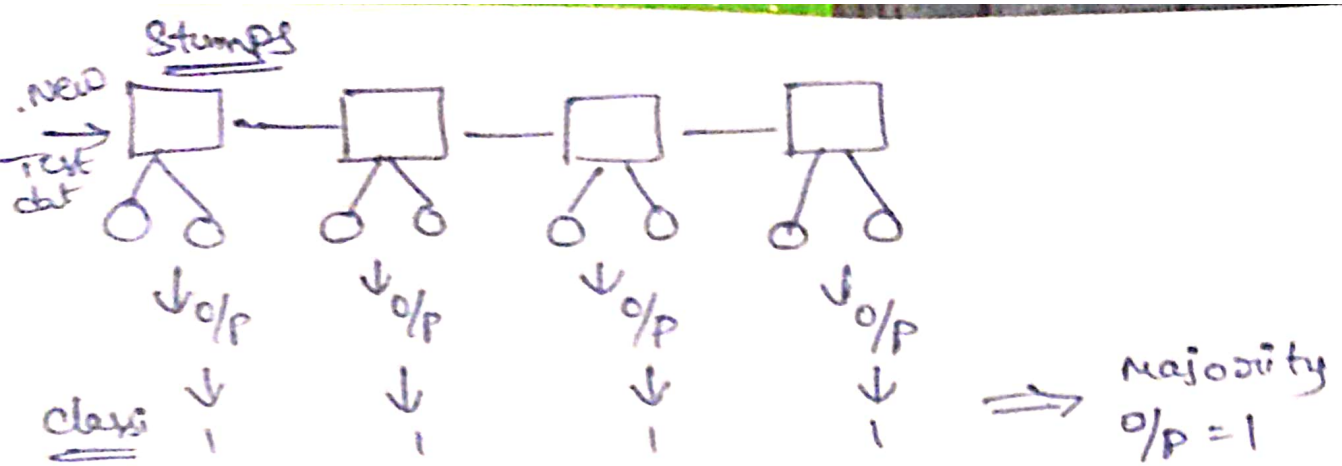
(1) Total Error = $\frac{1}{7}$

(2) performance of stump

$= \frac{1}{2} \log_e \left(\frac{1-TE}{TE} \right)$

$= \frac{1}{2} \log_e \left(\frac{1-1/7}{1/7} \right)$

$= 0.895$



Regression ⇒ o/p = mean.

Black box models vs white box Models

- Linear Regression → white box (able to see $\theta \cdot B_i$ changing)
- Random Forest → Black box (impossible to see all Decision Trees & working)
- Decision Tree → white box (Able to know how data Splitting)
- ANN → Black box (nodes & weights not table to see).

Probabilities.

① Classical probability $P(E) = \frac{\text{No. of favorable outcomes}}{\text{Total no. of outcomes}}$

Ex: $P(H) = 1/2$
↓
Head

② Empirical:

$$P(E) = \frac{\text{No. of times times Event occured}}{\text{Total No of trails}}.$$

Ex: If you roll a dre 100 times get 4 on 18 times

$$P(4) = \frac{18}{100} = 0.18$$

③ Subjective

⇒ Based on personal belief, intuition or Expert judgment, not actual data.

Ex: A doctor believes there is 70% chance a patient will recovel.

④ Axiomatic.

Based on mathematician Andorey Set of axioms & rules.

Axioms

1. $0 \le P(E) \le 1$
2. $P(S) = 1$, where S is the Sample space
   ↓
   possible outcomes set
3. For any two mutually Exclusive Events A and B.
   ↓ Not occured at same time.

$$P(A \cup B) = P(A) + P(B)$$

Bayesian Statistics

(1) prior probability - $P(A)$

* The probability of an event before observing new data

$$P(\text{Disease}) = 0.01.$$

(2) Likelihood - $P(B/A)$

* data under the assumption. (probability based on condition)

(3) posterior - $P(A/B)$

* updated probability of the hypothesis after seeing new data.

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

(4) Marginal probability. - $P(B)$

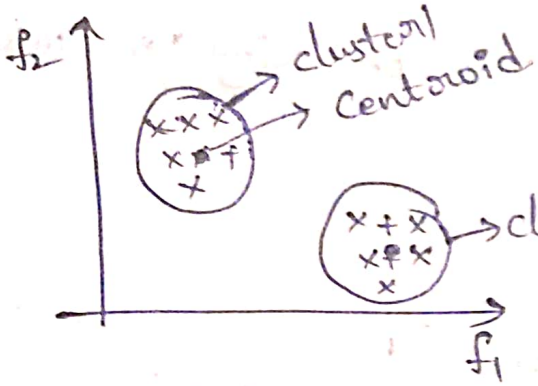
The total probability of observing the Evidence, Considering all possible Causes.

Unsupervised Algorithms :- Don't have Specific output

(8) K-Means clustering ⇒ clustery

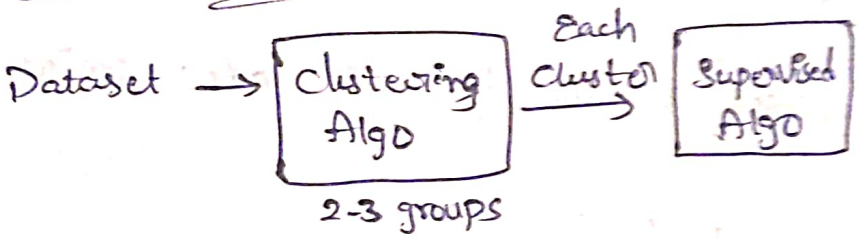
↓

Similar kind of data groups.



⇒ K=2

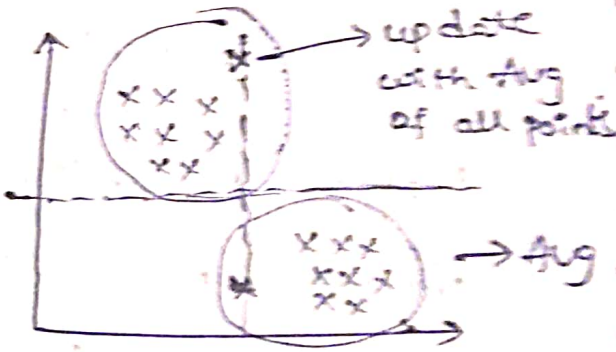
Custon Ensembel Technique.



K Means

↓

K = Centroids.



(1) We try differen K values ⇒ suitable K=2

(2) Initialize K no.of Centaroids.

(3) Groups Centroids nearest datapoints as one cluster using Euclidian distance formula.

(4) Compute the Avg to update centroids.

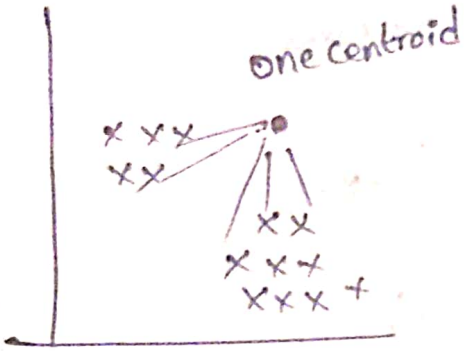
* It's better to initialize Centroids very far to get Centroid Exactly in center. But it may be chance of getting More clusters. So, KMeans ++ (Ensures Centroids are far)

How we decide K Value ?

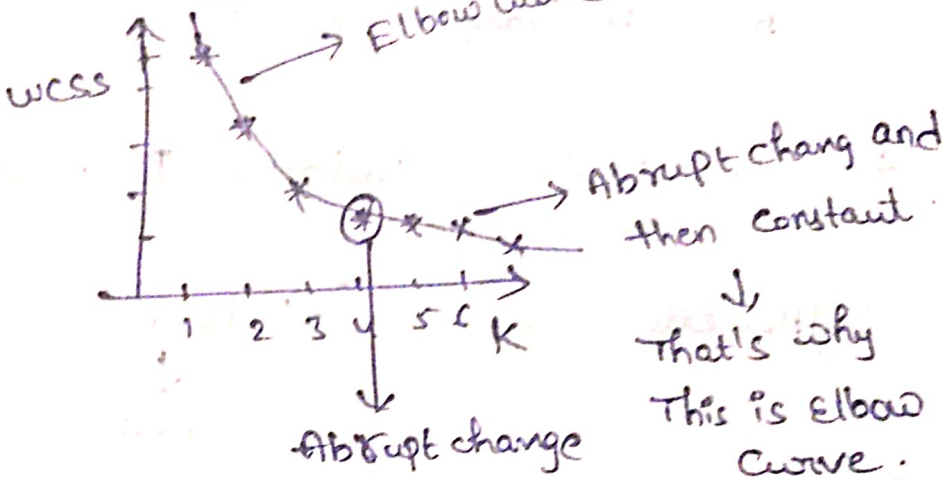
⇓

Elbow Method.

WCSS = within cluster Sum of Square.



for K=1,10



↓

That's why This is Elbow Curve.

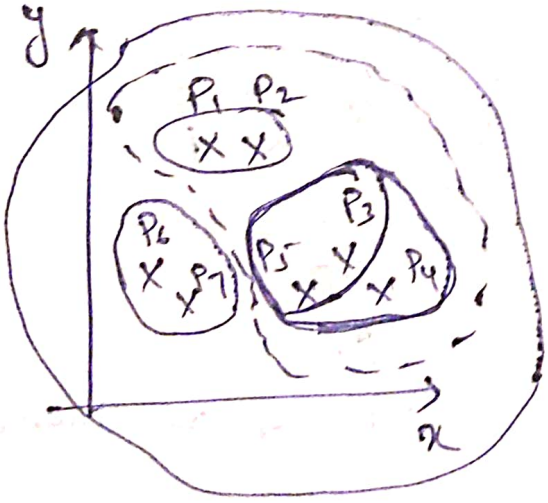
↓

So, it may be our K value.

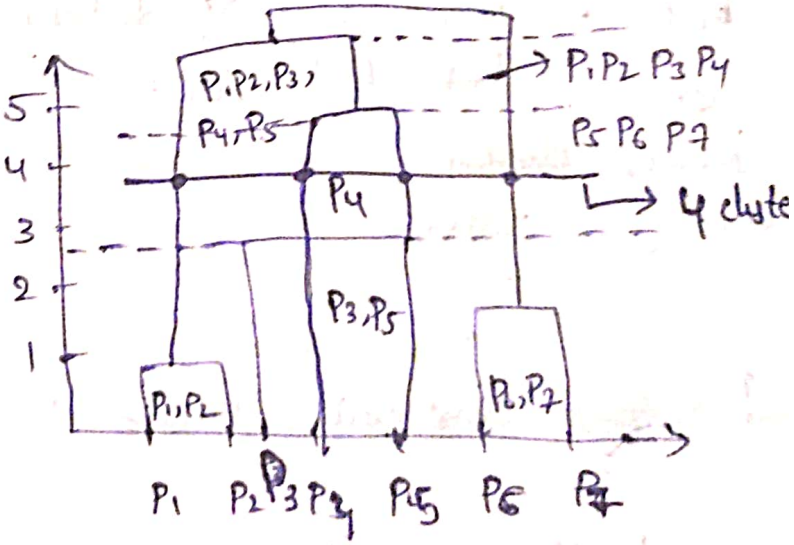
K=4 (4 clusters)

⑨ Hiraarichal clustering

↳ step by step process. ⇒ ~~Evaluate or~~ Identify nearest points & Combine



⟹



⇓

Dendogram.

* How to find how many groups there?

⇒ Need to find the longest vertical line that has no horizontal line passes through it.

* Max time is taken by Hierarchical clustering than K-Means clustering.

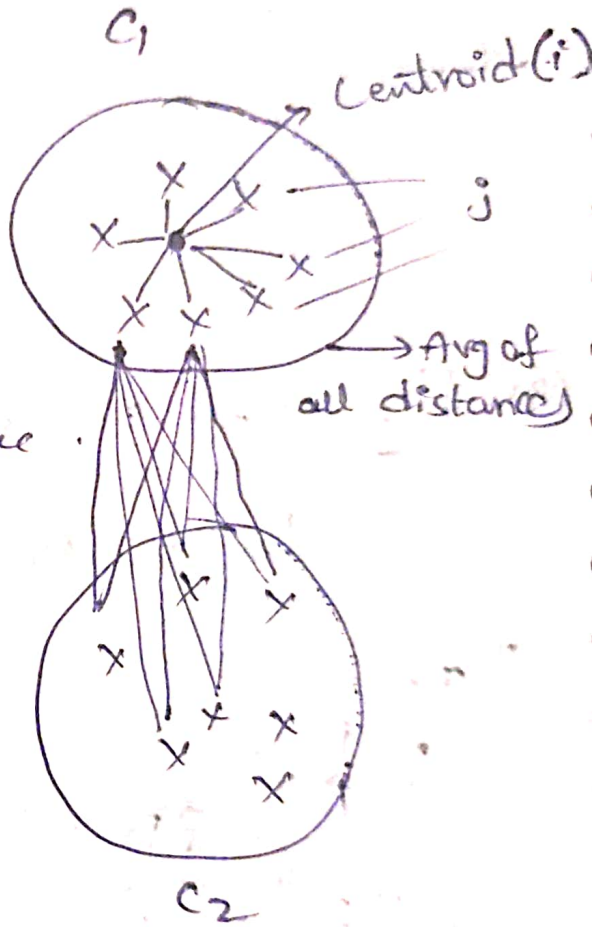
Dataset is small ⇒ go with Hierarchical
" is large ⇒ " " K-Means

How we validate clustering problem (Evaluation Metric?)

Silhouette score. (-1 to 1)
↓
Best.

⇓

For K-Means



① $$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_j, i \neq j} d(i,j)$$ ⇒ Avg distance.

$$b(i) = \min_{i \neq 1} \frac{1}{|C_j|} \sum_{j \in C_j} d(i,j)$$

↓
Nearest cluster among all clusters

⇓

Avg of all distances.

For good clustering Model.

$$b(i) >> a(i)$$

⇓

This means distance b/w clusters are more that is clearly grouped as diff clusters.

* ~~(~~$a(i) >> b(i)$ × bad.

Silhouettee clustering formula.

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_I| > 1$$

$b(i) >> a(i)$ ↑

$\Rightarrow -1 \text{ to } 1$

↓

$a(i) >> b(i)$

(10) DBscan clustering

Density - Based Spatial clustering of Applications with Noise

⇓

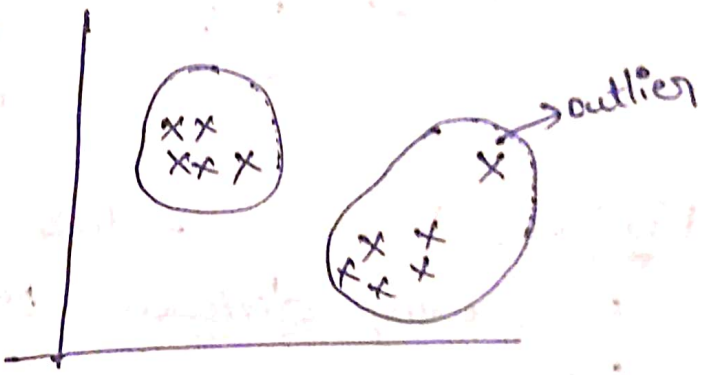
· It skips outliers & give clusters.

⇒ 1) Min points

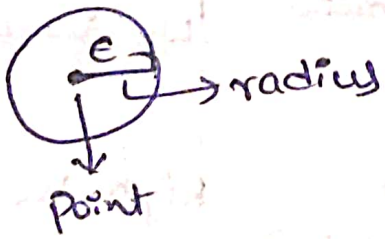
2) Core points

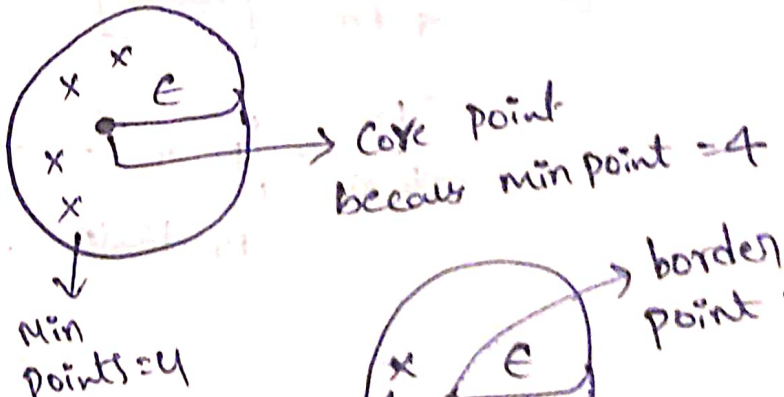
3) Border points

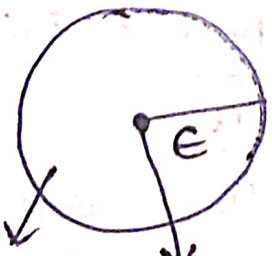
4) Noise points.



Min pts = 4 ⇒ Hyperparameter.







→ never taken into group.

Depth of Bias and Variance.

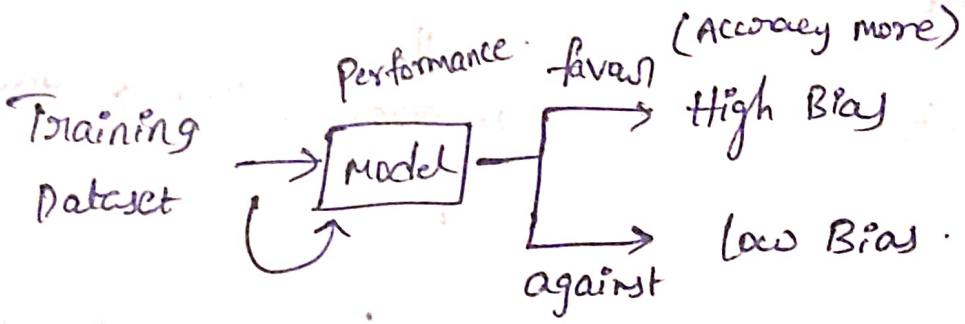
Training Dataset = 90% } ⇒ overfitting
Test dataset = 10% }

⇓

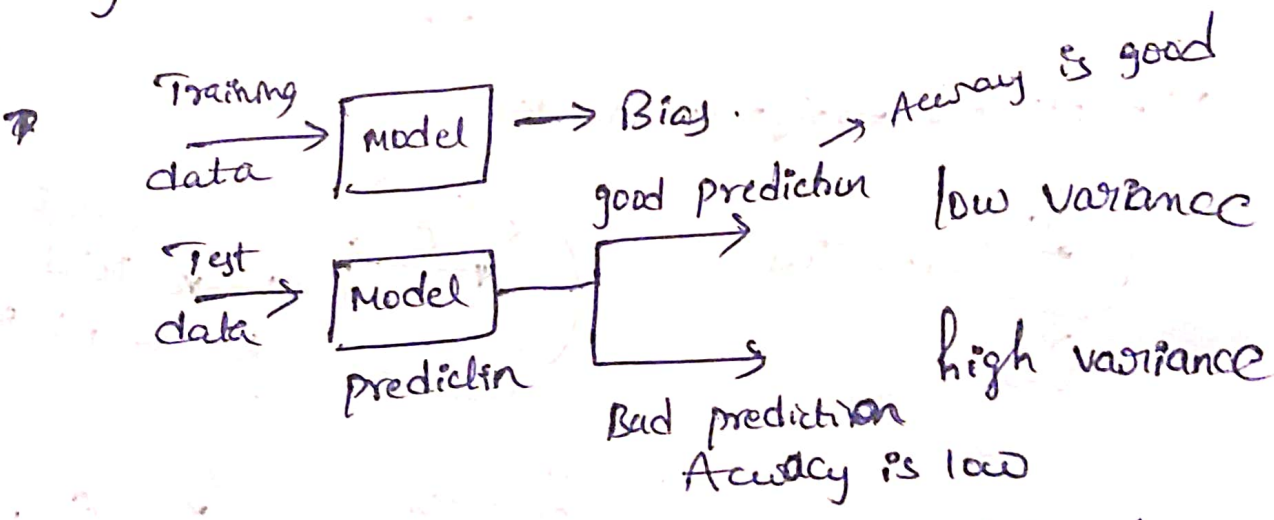
low Bias
high Variance.

Bias :- It is a phenomenon that skews the result of an algorithm in favour or against an idea.

↳ Training dataset.



Variance: Variance refers to the changes in the model when using different portions of the training or test data.



Model 1

Training Acc = 90%
Test Acc = 75%

⇓

low Bias
High Variance

Model 2

Train acc = 60%
Test acc = 55%

⇓

High Bias
High variance

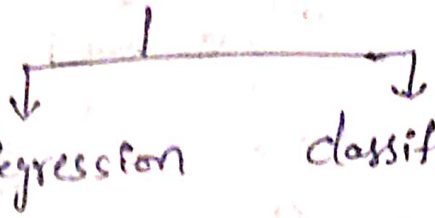
Model 3 (Genralize model)

Train acc = 90%
Test acc = 92%

⇓

Low Bias
low Variance

(11) Xgboost classifier → xtreme Gradient Boosting



Salary	credit	Approval
<=50	B	0
<=50	G	1
<=50	G	1
>50	B	0
>50	G	1
>50	Nomal	1
<=50	N	0

Base model — probability = 0.5 (for classification)

Weak learner

Residual

0-0.5=0.5
0.5
0.5
-0.5
0.5
0.5
-0.5

Creating decision tree after base model.

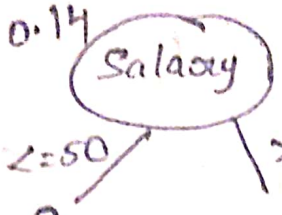
① Create a Binary Decision Tree using the features.

② Calculate Similarity weight.

$$= \frac{\sum (\text{Residual})^2}{\sum (Pr(1-Pr) + \lambda)}$$

(3) Calculate Information Gain

[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]

→ 

Binary decision tree

[-0.5, 0.5, 0.5, -0.5] [-0.5, 0.5, 0.5]

Residual = Approval - probability

⇒ Calculate similarity weight

$$= \frac{[-0.5+0.5+0.5-0.5]^2}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)} = 0 \quad \boxed{\lambda=0}$$

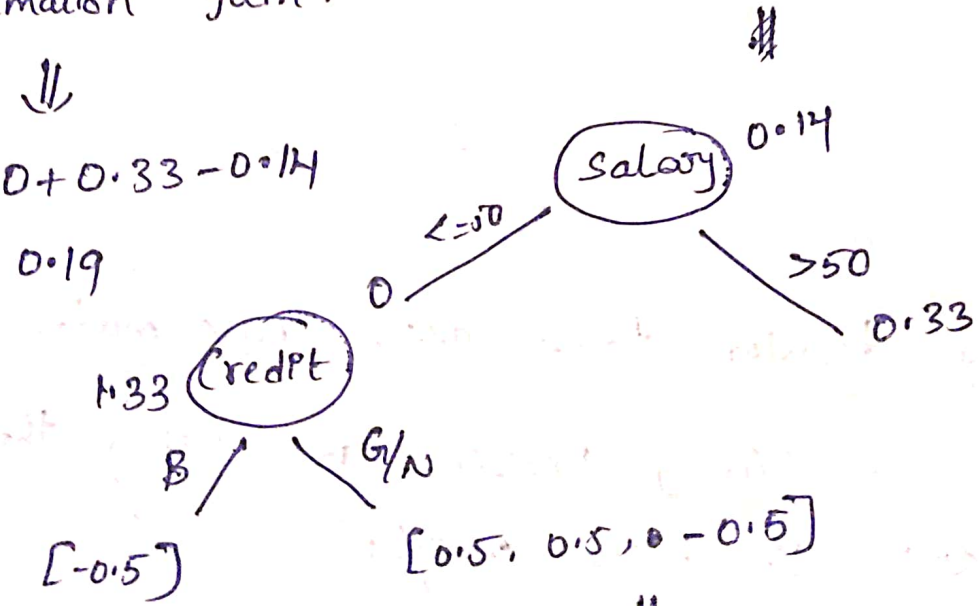
$$= \frac{(-0.5+0.5+0.5)^2}{0.5(1-0.5)+0.5(1-0.5)+0.5(1-0.5)} = \frac{0.25}{0.75} = 0.33$$

Root node similarity weight $= \frac{0.25}{1.75} = \frac{1}{7} = 0.14$

⇒ Information gain.

$= 0 + 0.33 - 0.14$

$= 0.19$



$Smw = \frac{0.25}{0.5(1-0.5)} = \frac{0.25}{0.25} = 1$

$\frac{0.25}{0.75} = 0.33$

⇒ 1 + 0.33 − 0 = 1.33

In Case of Base model.

$$\log\left(\frac{P}{1-P}\right) = \log\left(\frac{0.5}{0.5}\right) = 0$$

$\sigma[0 + \alpha(1)]$ ← similarity weight

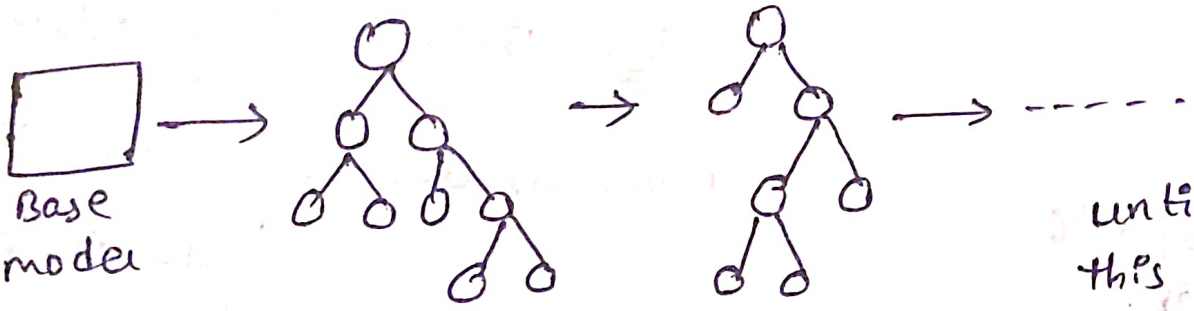
↓ ↳ learning rate (0 to 1)

activation function.

$\lambda \Rightarrow$ fixed by Cross Validation.

$$\sigma\left[0 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3) + \alpha_4(DT_4) + \cdots \alpha_n(DT_n)\right]$$

* xgBoost is a Black Box model (prone to overfitting)



xgboost Regressor (Bagging)

Exp	Grad	Sal	Resi
2	Yes	40K	-11K
2.5	Yes	40K	-10
3	No	52K	1
4	No	60K	9
4.5	Yes	62K	11

Base model → Avg Sal → 51K.

Exp (-11, 9, 1, 9, 11)

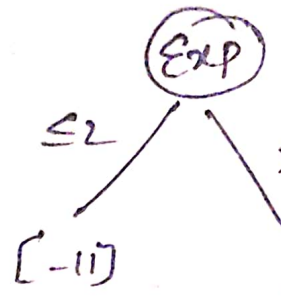
≤ 2 : [-11]

> 2 : [-9, 1, 9, 11]

$$\text{Simi weight} = \frac{\sum(\text{Residuals})^2}{\text{No. of Residuals} + \lambda}$$

↓

$\lambda = 0$



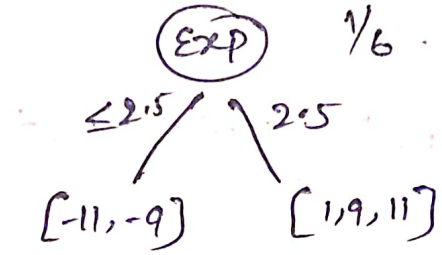
[-11, -9, 1, 9, 11] $\Rightarrow \frac{-11-9+1+9+11}{5+1} = \frac{1}{6}$

$\frac{121}{1+1} = 60.5$ $\frac{(-9+1+9+11)^2}{4+1} = \frac{144}{5} = 28.5$ $(\because \lambda = 1)$

Infomain Gain = 60.5 + 28.8 − 1/6

~~= 89.34~~

= 89.13



$Simw = \frac{(-11-9)^2}{2+1} = \frac{400}{3} = 133.33$

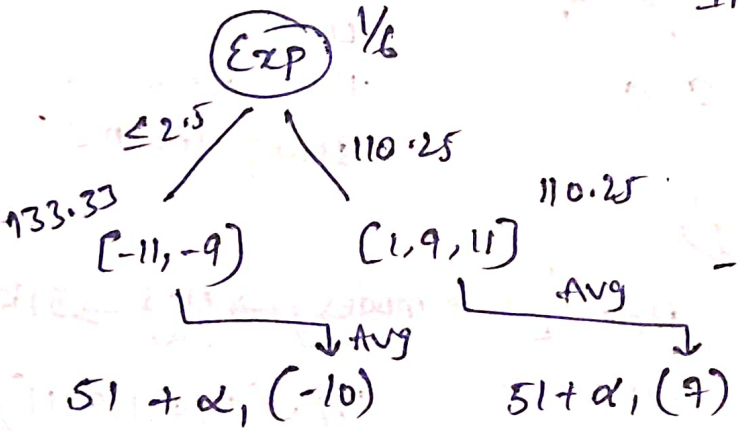
$Sim = \frac{(1+9+11)^2}{3+1} = \frac{21^2}{4} = 110.25$

Infomain Gain = (1.33 + 110.25 − 1/6)

↓ greater than previous

↓

So we choose this split.



$\frac{-11-9}{2} = \frac{-20}{2} = -10$

51 + α, (−10)

51 + α, (7)

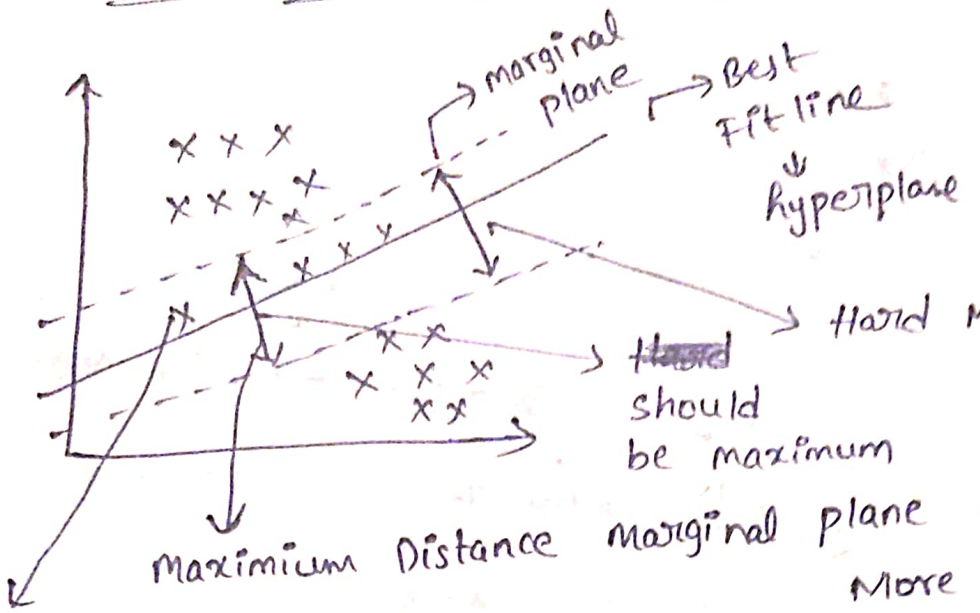
↓

Bayes model

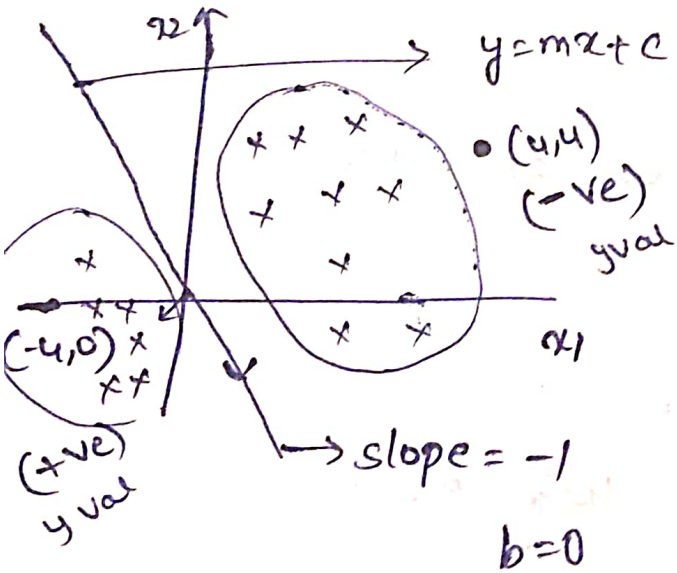
$O/p = 51 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3) + \cdots + \alpha_n(DT_n)$

⇒ This is also Black box model (Can't visualize all of this).

(12) SVM (Support Vector Machine)



* Aim is Creating marginal plane with maximum distance Even though there are some Errors we Consider it in soluing by proving some hyperparameter



$ax+by+c=0$ ⇒ Equation of straight line

$y = -\frac{c}{b} - \frac{ax}{b}$

$m = -a/b$

$c = -c/b$

$y = mx + c$

$\Downarrow$

$y = w_1 x + w_2 x_2 + \cdots w_n x_n$

$\boxed{y = w^T x + b}$

$b = 0$

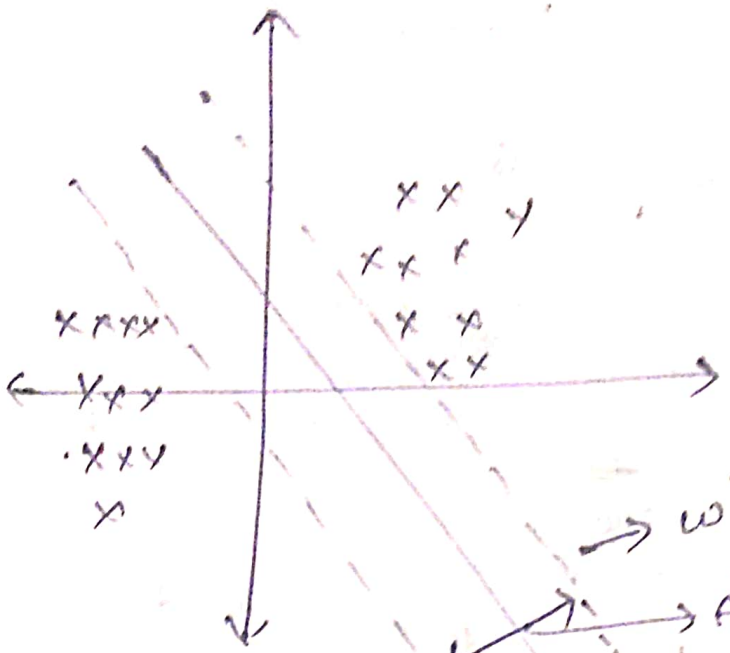
$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [-4, 0]$

$\downarrow$ w^T $\quad$ $\downarrow$ x

$= 4$ (matrix multiplication) ⇒ +ve value.

$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4, 4)$

$\downarrow$ slope & intercept $= -4$



$-K$

$\rightarrow w^T x + b = -1$ (-ve value)

$\rightarrow$ Aim to increase this distance

$\rightarrow w^T x + b = 0$ $(ax + by = 0)$

$\rightarrow w^T x + b = 1$ (+ve value)

K

$w^T x_1 + b = 1$

$w^T x_2 + b = -1$

(-) (-) (+)

$w^T (x_1 - x_2) = 2$

$\frac{}{|w|}$ $\frac{}{|w|}$ $\rightarrow$ magnitude of vector.

Aim is maximize $\frac{2}{\|w\|}$

$\Downarrow$

using (w, b)

Such that y_i $\begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$

$\Downarrow$

Major aim $y_i * (w^T x_i + b) \geq 1$

$\Downarrow$

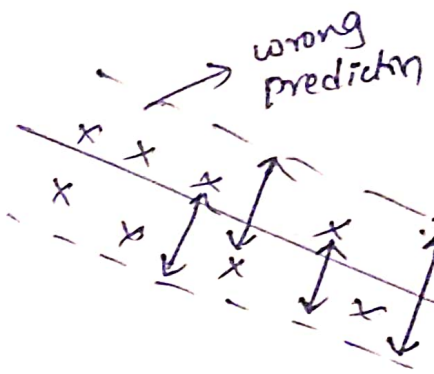
for Correct point

$$\underset{(w,b)}{\text{maximize}} \quad \frac{2}{\|w\|} \iff \underset{(w,b)}{\min} \quad \frac{\|w\|}{2}$$

$$\underset{(w,b)}{\min} \quad \frac{\|w\|}{2} + C_i \sum_{i=1}^{n} \xi_i$$

C_i ↓ How many Errors we can have

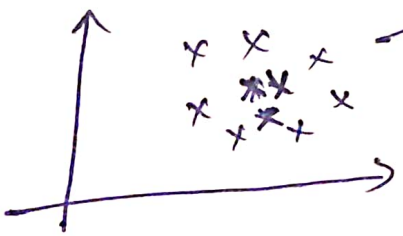
ξ_i ⇒ summation of the distance of the wrong point



SVR

$C_i \sum_{i=1} \xi_i$ ⇒ only will change

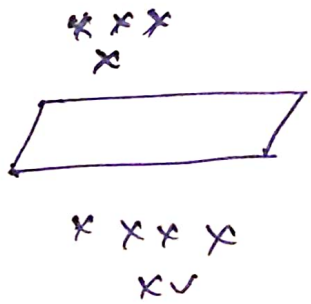
SVM Kernel



→ Cannot do with straight line

⇓

so we convert it int 3 dimension



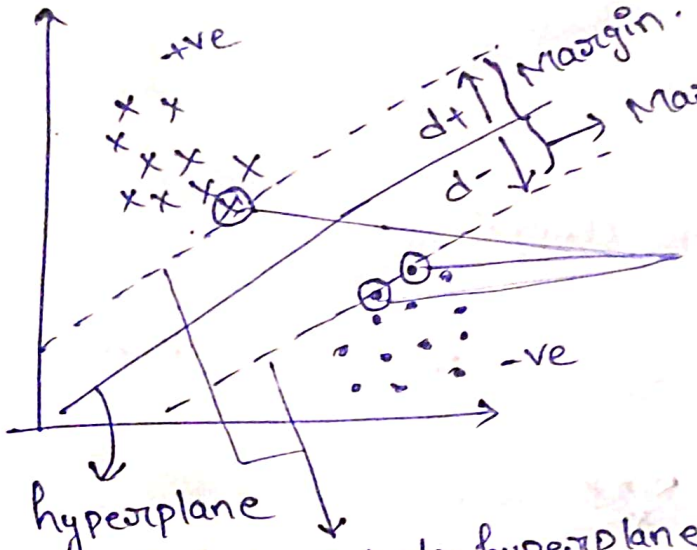
→ hyperplane.

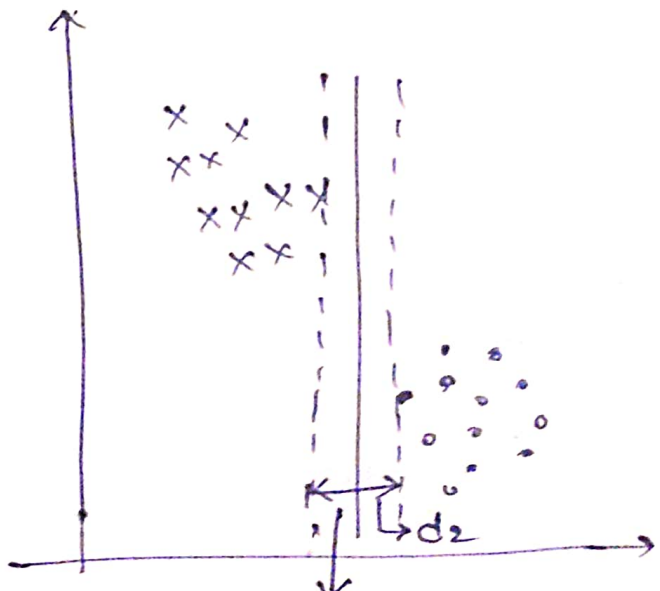
Support Vector Machines.

→ It is used to solve both classification & regression problems.

1) Support Vectors
2) Hyperplanes
3) Marginal Distance
4) Linear separable points
5) Non linear Separable points.

Geometrical Intuition.



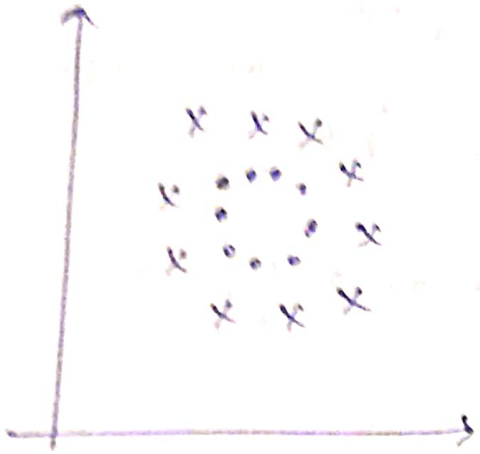


* Aim is to Maximize Marginal distance. That becomes best hyperplane.

$d_1 >> d_1$

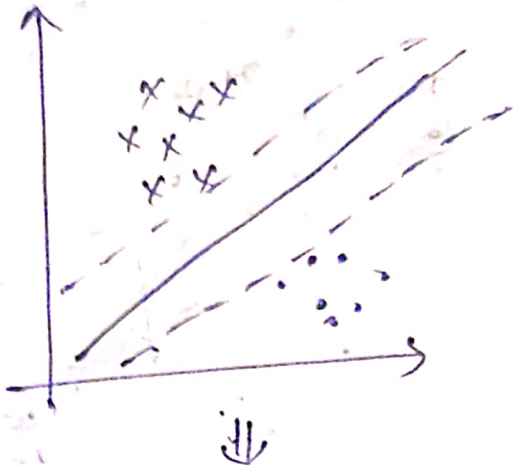
is a plane

* In 3D we say hypeorplane which separate the classes Maximally. But in 2D hypeorplane is straight line.



⇓

* Not linearly separable because we can't divide this 2 classes by a straight line



⇓

* linearly separable because we can divide 2 classes by a straight line

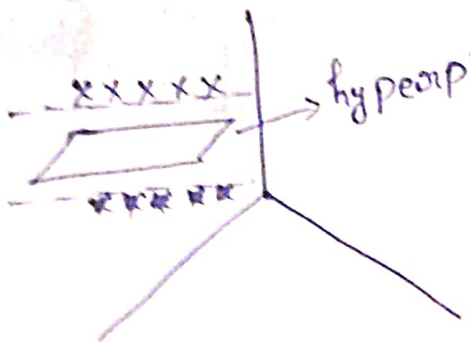
⇓

We can solve this type of problem using SVM Kernels.

⇓

SVM Kernels Converts low Dimension to high dimension.

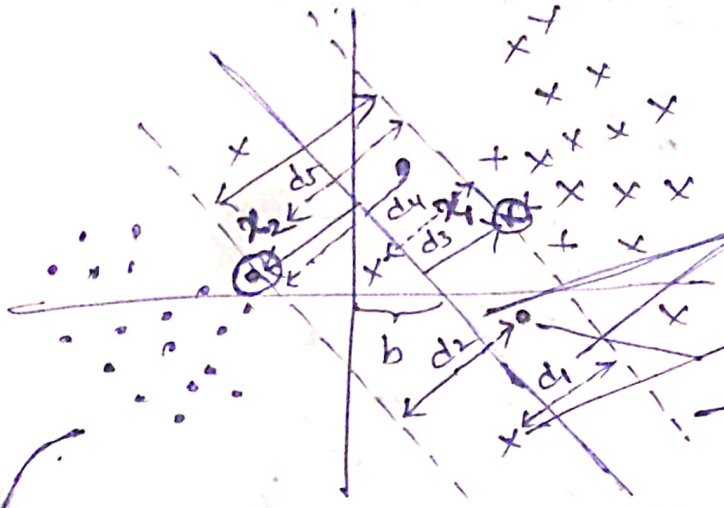
⇓



hypeorplane which helps to ~~linearly~~ separable the classes.

* Aim is to identify a hypeorplane that have maximum marginal distance which separates classes.

SVM Math Intution.



value of the errors (distance of all sum).

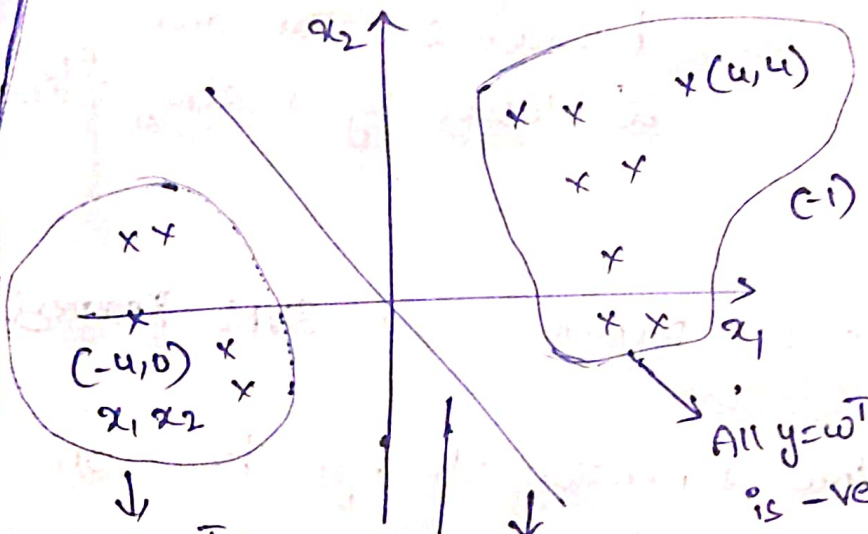
Errors (2-3 Errors OK)
4-5

Errors

$\omega^T x + b = -1$ ⇒ Negative plane

$\omega^T x + b = 0$

$\omega^T x + b = +1$ (+ve plane)



$m = -1$

c or $b = 0$

$y = \omega^T x + 0$

$= \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} -4 & 0 \end{bmatrix} = 4$ +ve value

$y = \omega^T x$

$= \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} 4, 4 \end{bmatrix} = -4$ -Ve

* $x_2 - x_1$ = Maarginal distance.

$\omega^T x_1 + b = -1$

$\omega^T x_2 + b = 1$

$\omega^T (x_2 - x_1) = 2$ ⟹ $\frac{\omega^T}{\|\omega\|} (x_2 - x_1) = \frac{2}{\|\omega\|}$

to remove ω^T

* optimization function is $\frac{2}{\|w\|}$ → we need to Maximize this.

update $(\vec{w}^*, b^*)$ max $\frac{2}{\|w\|}$

$$\text{Such that } y_i \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$$

$$\Downarrow$$

$$\boxed{y_i * w^T x_i + b_i \geq 1}$$

↓ if it is <1 it is Misclassification.

* In real world scenario lot of overlapping of points will be there. we can't simply classify data.

$$(\vec{w}^*, b^*) = \min \frac{\|w\|}{2} + \boxed{C \sum_{i=1}^{n} \xi_i}$$

→ Hinge loss.

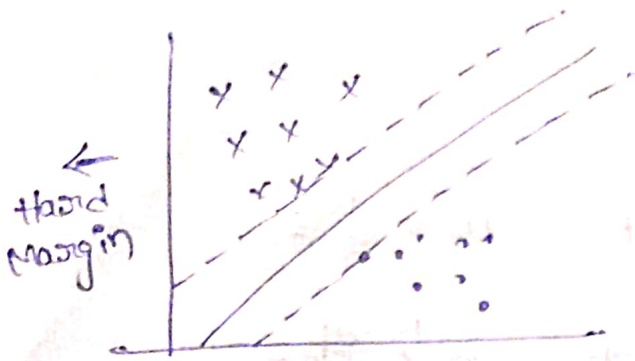
→ value of the error

How many Errors model Can Consider

Regularization. (we get by hyperparameter tuning).

* Hard SVM ⇒ strictly divides points into different classes.

SVM Kernels Indepth Intuition.



① soft Margin (Have Errors of C=1,2,3--)

② Hard Margin (NO Errors)

1) Polynomial Kernels
2) RBF Kernel
3) Sigmoid Kernel.

⇓

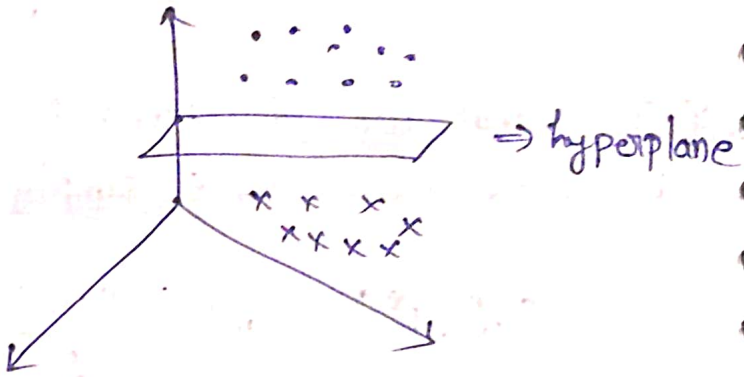
Not linearly separable

⇓

SVM Kernels.

⇓

Converts low to high dimension



⇓

Here 2D to 3D.

⇓

SVM Kernels does this by some Mathematical Formulas.

⇓

Not Exactly separable.

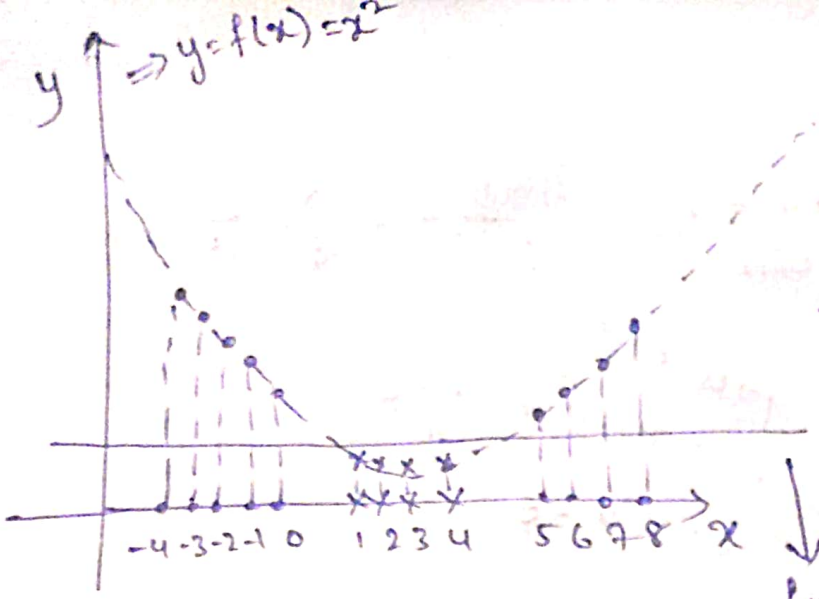
1d → 2d

⇓

using some kernel.

$y = f(x)$

$= x^2$

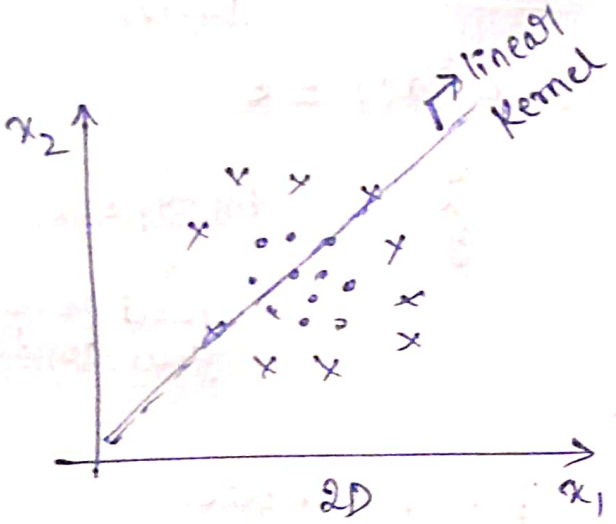


$\Rightarrow$ projected 1D to 2D

$\Downarrow$

Kernel used $= x^2$.

hyperplane which classifies data into classes.



$\Rightarrow$ linear kernel for this 50% accuracy only.

$\Rightarrow$ Now, polynomial Kernel Apply

$p.k = (x_1^T \cdot x_2 + 1)^d$ $\rightarrow$ dimensions

$f(x_1, x_2)$

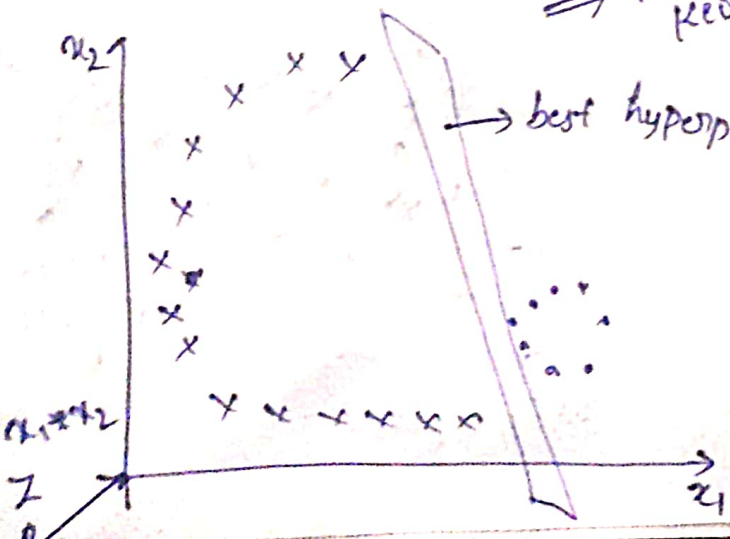
x_1	x_2	y_0	$y = f(x_1, x_2)$

$f(x_1, y_2) = (x_1^T \cdot x_2 + 1)^d$

$\Rightarrow$ x_1 x_2 y_0 x_1^2 x_2^2 $x_1 x_2$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

$\Downarrow$ 3D



$\Rightarrow$ polynomial kernel.

$\rightarrow$ best hyperplane

$$= \begin{bmatrix} x_1^2 & x_1 x_2 \\ x_2 x_1 & x_2^2 \end{bmatrix}$$

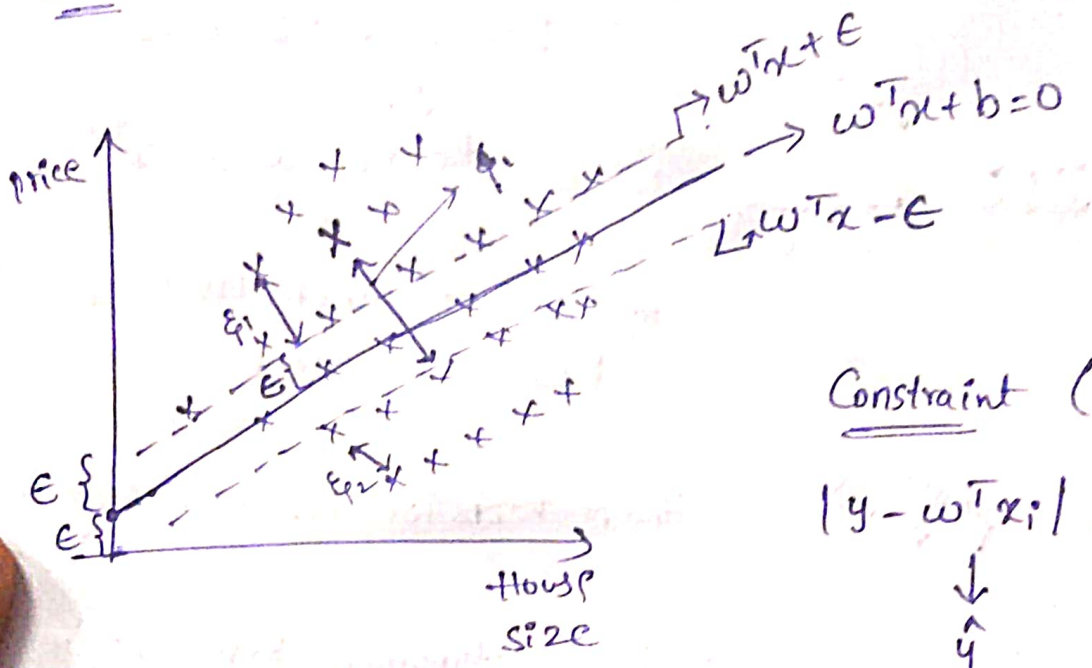
$\Rightarrow$ Type of kernel

1) Polynomial
2) RBF
3) Sigmoid

} determined by hyperparameter Tuning.

Support Vector Regression

SVC ⇒ Cost Function = $\underset{w,b}{\text{minimize}} \quad \frac{\|w\|}{2} + C \sum_{i=1}^{n} \xi_i$ → Extra.



Model Constraint (without outside Points).

$|y - w^T x_i| \leq \epsilon$

$\downarrow$

$\hat{y}$

$|y - \hat{y}| \leq \epsilon$

$\downarrow$

good point.
good prediction.

Cost Function for outside points.

$\underset{(w,b)}{\text{minimize}} \quad \frac{\|w\|}{2} + \boxed{C \sum_{i=1}^{n} |\xi_i|}$

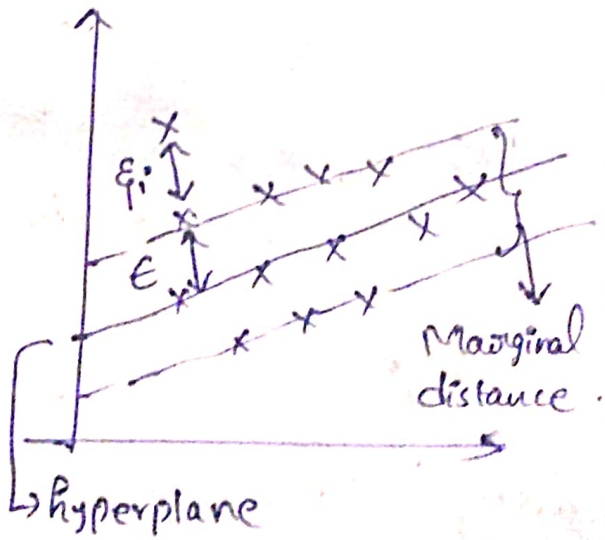
→ hyperparameter.

→ outside points distance to b

→ How many points Can have outside Marginal plane.

Constraint

$|y_i - w_i x_i| \leq \epsilon + |\xi_i|$



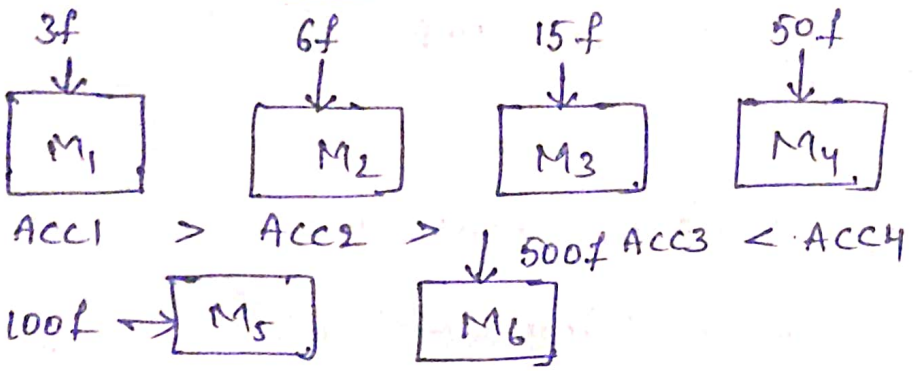
Principal Component Analysis (PCA) [Dimensionality Reduction]

Why should we use PCA?

① Curse of Dimensionality

Dimensions is features

Dataset :- 500 features



Price of House

House size
No. of bedrooms
No. of bathrooms

* ACC1 > ACC2 > ACC3 < ACC4 < ACC5 < ACC6.

3f 6f 15f 50f 100f 500f

Model is overfeeded

② Model performance Degrade as no. of features increasing.

⇓

Model performance reduces because many Mathematical formulas run behind at one point of time give confused data.

* Two different ways to remove Curse of Dimensionality.

1) Feature selection.
   ⇒ Take only Imp features ⇒ Train our model.
2) Dimensionality Reduction (PCA) ⇒ Feature Extraction.
   ⇒ Derive a feature from a set of features.

f_1 f_2 f_3 f_4 ⇒ o/p

⇓ Feature Extraction.

Df_1 Df_2 ⇒ o/p

* Feature selection & Extraction helps to reduce the no.of features or to Extract Imp features from older features.

@ why we use Dimensionality reduction?

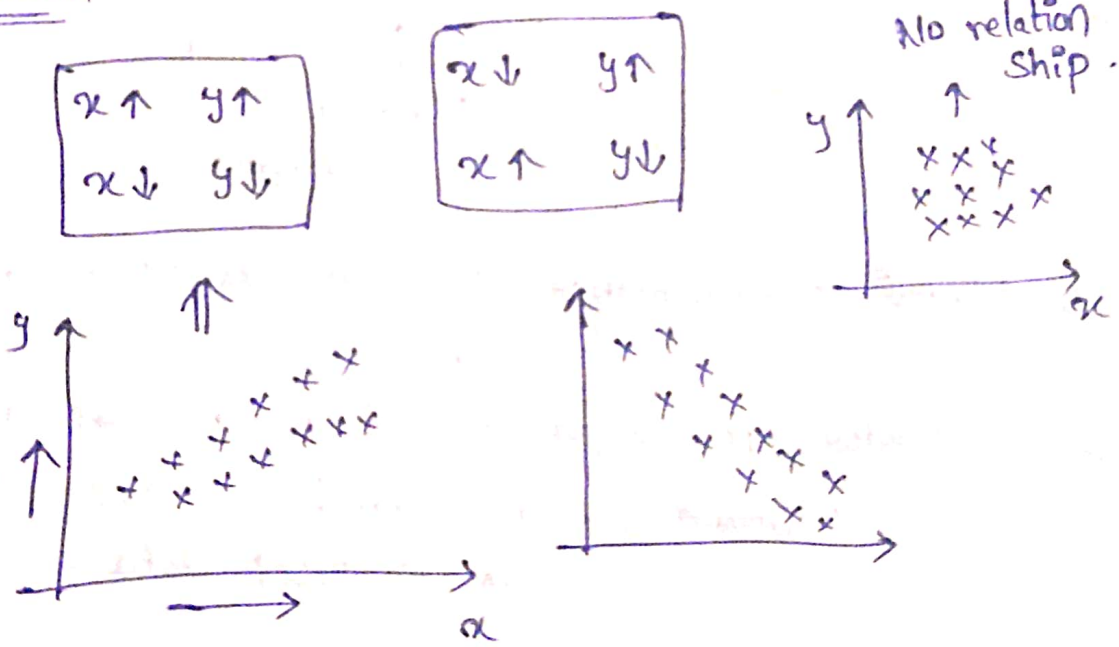
⇒ prevent → Curse of Dimensionality.

⇒ Improve the performance of the model. (reduce model time of Execution)

⇒ visualize the data → understand the data.

Feature selection

I/P	O/P
x	y
-	-
-	-
-	-



* $Cov(x,y) = \frac{\sum_{i=1}^{n} (x-\bar{x}) * (y_i-\bar{y})}{n-1}$ → for sample data

⇓

+ve → linear Relatiship follows ⇒ x↑ y↑ / x↓ y↓

-ve → inverse linear Relationship follows. ⇒ x↑ y↓ / x↓ y↑

- Covariance ≈ 0 $\Rightarrow$ No linearship b/w x & y.

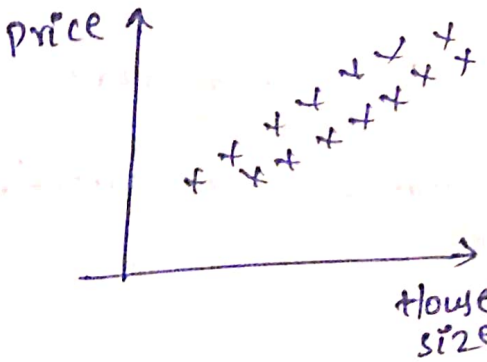
* Pearson Correlation $= \frac{Cov(x,y)}{\sigma_x \sigma_y}$ $= -1$ to 1

$\Rightarrow$ The more towards the value $+1$ $\Rightarrow$ The more +ve Correlated & vice versa

$\Rightarrow \approx 0 \Rightarrow$ No correlation.

Data Housing — Independent feature

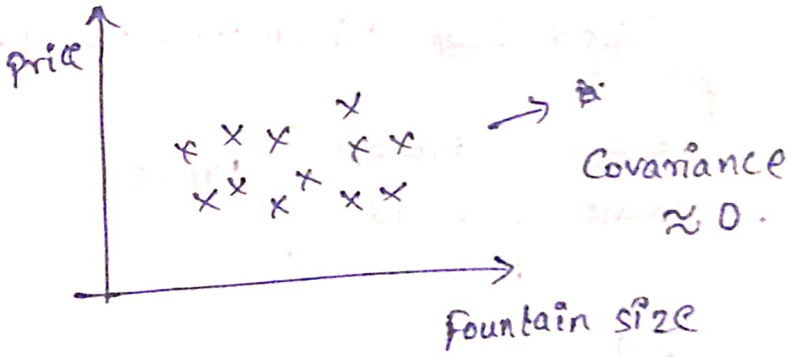
House size — Fountain size — price → dependent feature.



Linear relationship

Covariance = +ve

↓ selected



No relationship.

↓

No need to select as a feature.

* This is called feature Selection.

⇓

Using Covariance & Correlation we do this.

Feature Extraction.

Room ~~House~~ size No. of rooms ⇒ Price.

⇓

Apply Transformation to Extract New feature.

⇓

House size ⇒ price.

Dimensionality reduction

⇓

2 features

↓

1 feature

* Here feature selection couldn't helpful bcoz 2 features are super imp to predict house price.

PCA Geometric Intuition.

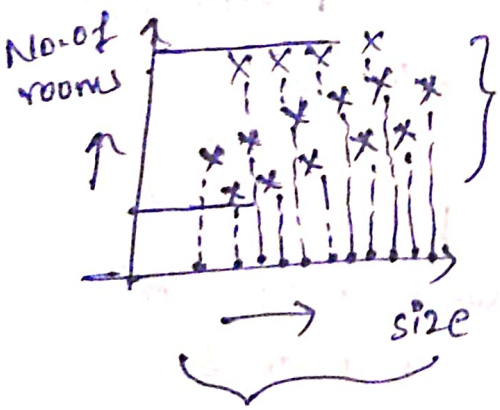
⇓

Used for Dimensionality Reduction.

size of House	No. of rooms	price.

PCA Aim Here

2 dimension → 1 dimension



spread ↑

⇓

Neglating this points by projecting in 1D.

2D → 1D

* loss of information (No. of rooms)

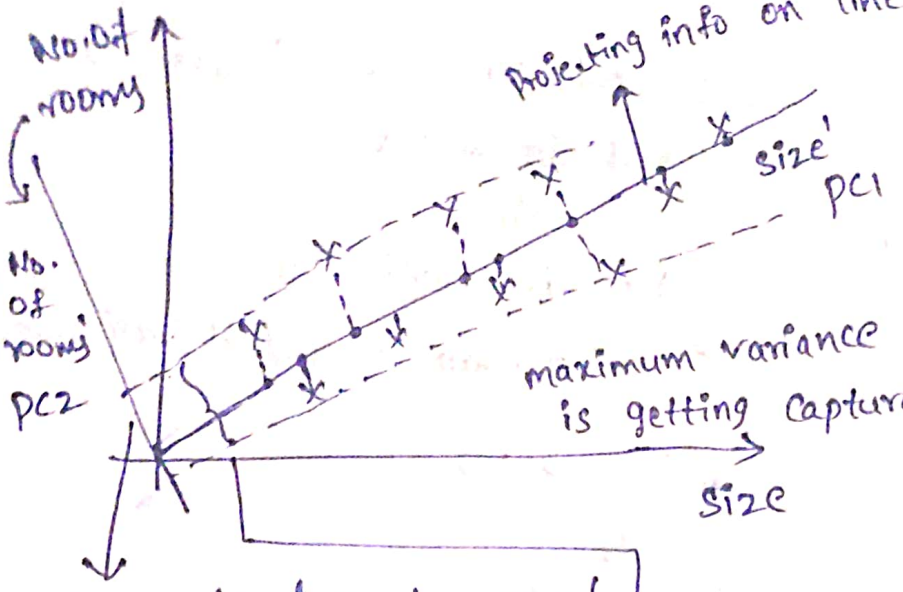
⇓

Freature Extraction with lot of information lost.

Spread of data points.

spread ↑

variance ↑



Eigen decomposition on Matrix.

⇑

Apply Transformation.

Perpendicular to size' ↓

spread size very less i.e loss of information is less.

* Here 2D ⟶ 1D.

⇓

Such that much information is not lost.

⇒ PC1 → captures Maximum Amount of variance.

⇒ PC2 → captures second maximum Amount of variance.

Goal of PCA Algorithm

* To find out the principal components in such a way that maximum variance will be captured.

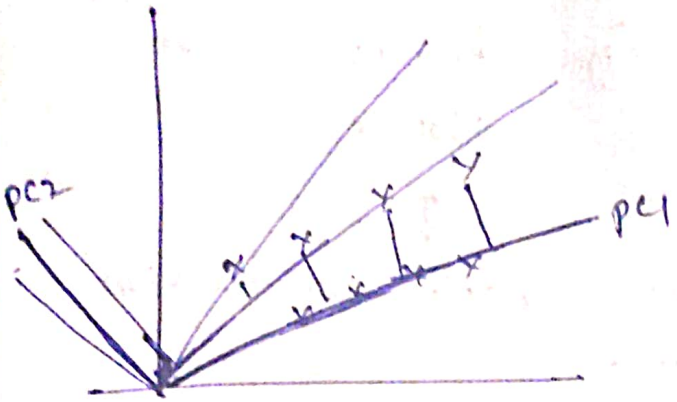
PC1, PC2 ← 2 dimensions

var(PC1) > var(PC2)

3 Dimension.

⇓

var(PC1) > var(PC2) > var(PC3).



* 2 Best principal Components Need to select

⇓

depends which Captures Maximum amount of variance.

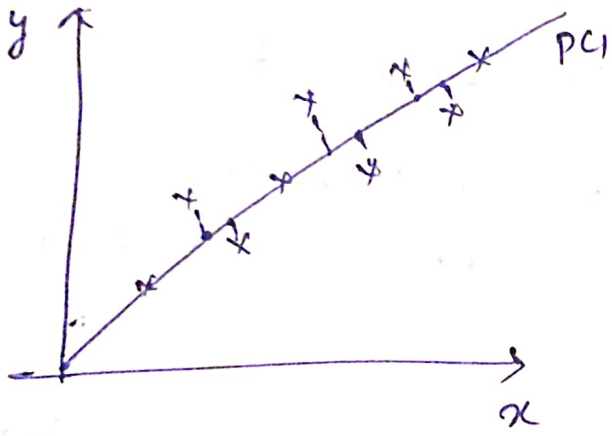
3D → 1D.

⇓

PC1, PC2, PC3.

Var(PC1) > var(PC2) > var(PC3)

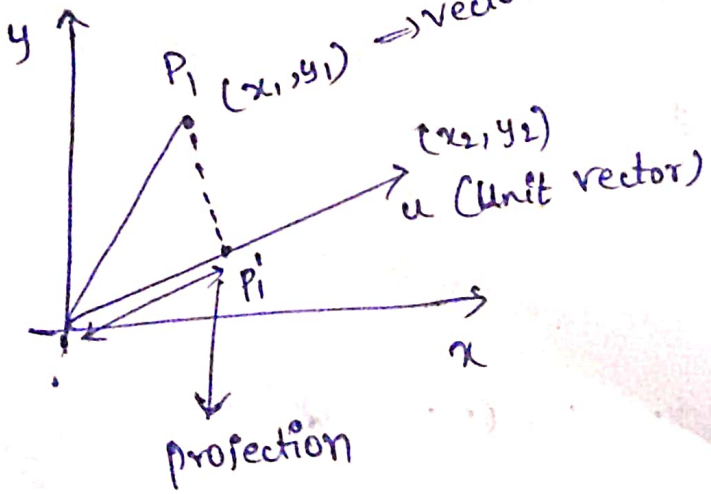
Math Intuition behind PCA Algorithm.



PCA comes to conclusion About Best PCA based on

① projection

② cost function.



$$\text{proj}_{P_1} u = \frac{P_1 \cdot u}{\|u\|}$$

$\|u\| = 1$ ⇒ unit vector.

$$\boxed{\text{prod}_{P_1} u = P_1 \cdot u}$$

⇓

Scalar value.

$P_0', P_1', P_2', P_3', \cdots, P_n'$

⇓

scalar values.

$x_0', x_1', x_2', x_3', \cdots x_n'$

Max variance $= \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$ [goal is to find best unit vector which captures maximum variance].

⇓

Cost function.

* It's not possible to find Every unit vector for each point & choose best.

⇓

Eigen Decomposition helps Here.

⇓

Eigen vectors And Eigen values

① Covariance Matrix between features need to find.

② Eigen vectors and Eigen values will found out from this Covariance Matrix.

③ Eigen vector → Highest Eigen value → Magnitude of the Eigen vector.

↓

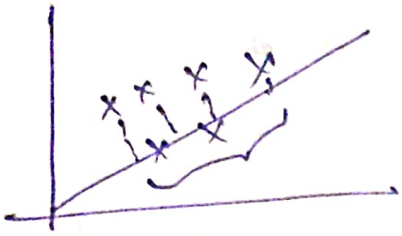
Captures the maximum variance.

Eigen vectors & Eigen values. [Linear Transformation]

Eigen decomposition of Covariance Matrix

⇓

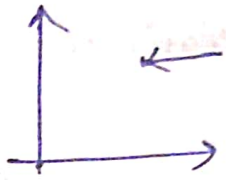
Eigen vector & Eigen values.

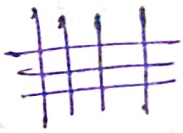


$$\begin{bmatrix} \quad \end{bmatrix} * [v] = \lambda * v$$

↓

Eigen value.





$$A * v = \lambda * v$$

↓ Matrix

↓ linear Transformator on vector

⇓

Among Eigen vector ⟹ which have maximum Magnitude.

⇓

That principal Component Captures maximum variance.

* Eigen vector ⟶ with Max Magnitude. (Max Eigen Valles)

⇓

Max Eigen vector

⇓

selected as Best principal Component

⇓

bcoz Captures. max variance.

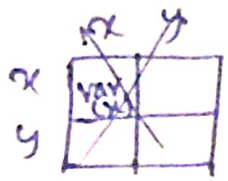
Steps to calculate Eigen value & vectors.

① Covariance of features.

[x, y] → x'

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

* $Cov(x,x) = Var(x)$



$$A = \begin{array}{c|cc} & x & y \\ \hline x & Var(x) & Cov(x,y) \\ y & Cov(y,x) & Var(y) \end{array} \quad 2\times2$$

$$\begin{array}{c|ccc} & x & y & z \\ \hline x & Var(x) & Cov(x,y) & Cov(x,z) \\ y & Cov(y,x) & Var(y) & Cov(y,z) \\ z & Cov(z,x) & Cov(z,y) & Var(z) \end{array} \quad 3\times3$$

② Eigen values.

$$A \cdot V = \lambda \cdot V$$

⇓

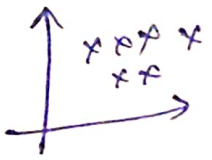
$\lambda_1, \lambda_2 \Rightarrow$ Eigen values.

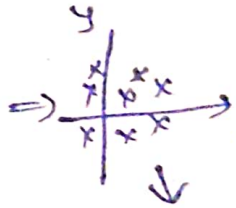
f_1 & f_2 → λ_1, λ_2 → Eigen values.

⇓ ⇓

PC_1 PC_2

① Standardize data. ⇒ means





zero-centered.

② Covariance Matrix of x & y. ⇒

$$\begin{array}{c|cc} & x & y \\ \hline x & Cov(x) & Cov(x,y) \\ y & Cov(y,x) & V(x) \end{array} \quad 2\times2$$

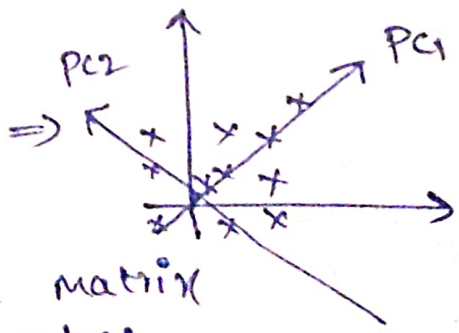
③ Find out Eigen vectors & values.

$$A\nu = \lambda\nu$$

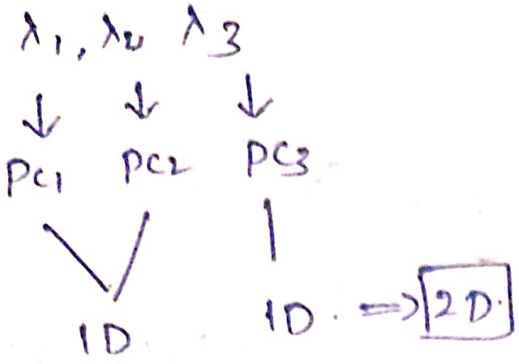
⇓

$\lambda_1, \lambda_2 \Rightarrow$ for 2×2 matrix Eigen values.

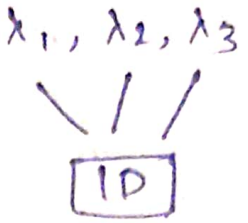
PC_1 PC_2



⇒ 3D → 1D.



3D → 1D.



2D → 1D