Machine Learning

→

1. Supervise learning
2. Regression & Applications
3. Clanifications & Applications
4. Un Supervised algorithm
5. Esmable learning
6. Recommender System

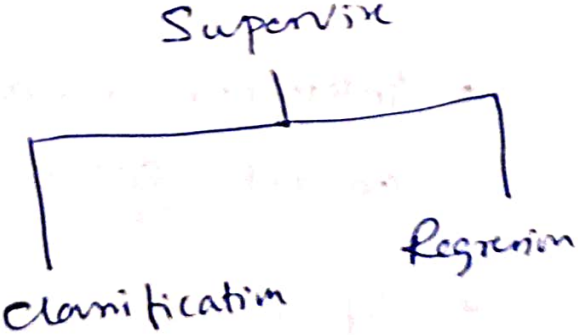
→ Supervise learning:-

Algorithm uses the data developed in Supervised Environments to deliver an output.

* It is a type of machine learning, where the model is trained on a labeled dataset.
* Here, Inputs & Outputs are known

Learning Methods:

* Linear Regression
* Logistic Regression
* Support vector machines
* Decision trees.



→ Unsupervise learning :-

* No Supervisor is required to prepare data as outputs may not be known for such algorithms.
* It's trained on unlabeled data.
* It tries to find hidden patterns & recognises their relation.

Learning methods :-

* K-means clustering
* Hierarchical clustering
*

→ Reinforcement learning :-

* The program learns from its previous errors.
* It's Type of machine learning, an agent learns to make decision by interacting with Environment.
* Interpreter rewards the algorithm when it finds the correct solution.
* If Output is not proper, the algorithm reiterates until it finds a better result.

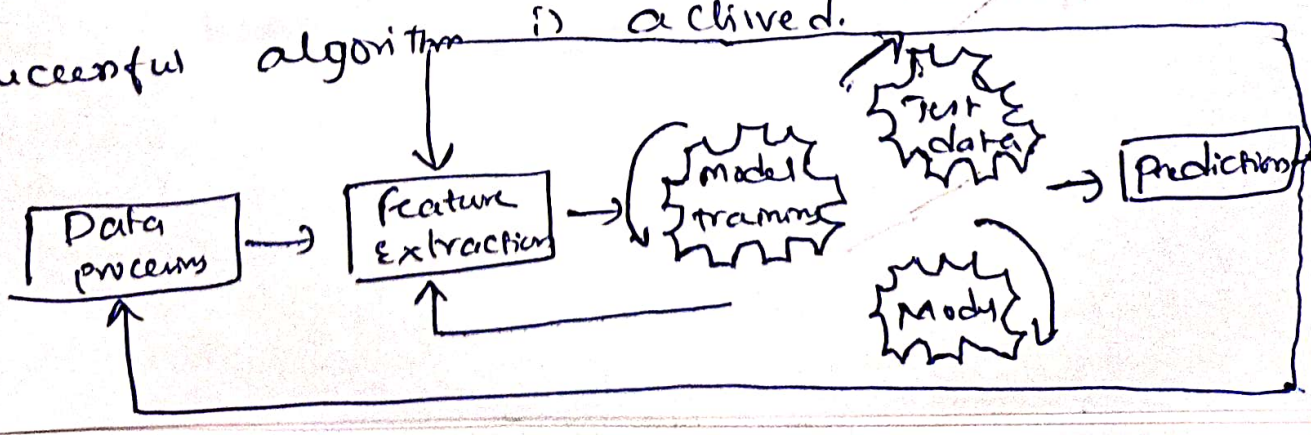
Machine learning pipeline :-

* It is a series of Sequential steps used to codify and automate ML workflows to produce ML models.

→ ML pipeline is an End-to-End construct that includes & orchestrates:

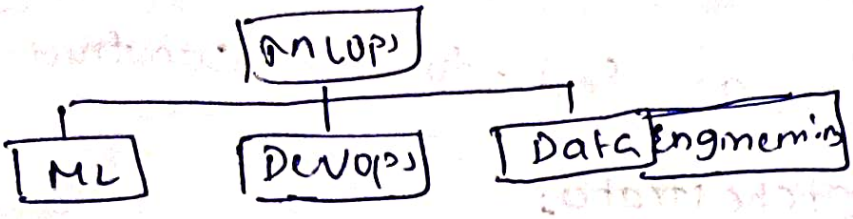
* Data extraction
* Raw data input
* Preprocessing
* Features
* Outputs
* Model parameters
* Model Training
* Deployment
* predicting outputs.

* The term pipline implies a one-way flow of data.
* ML pipelines are cyclical & iterative
* Every step in the sequence is repeated until a Succesful algorithm is achived.



→ Machine learning & operations professionals (MLOPs)

* MLOPs is a set of practices that combines,



* Ensures reliable & Efficient deployment & maintenance of ML models in production systems.

Aims to:-

* Improve communication & collaboration blw ~~devop.~~ MLOPs
* Shorten & manage complete development life cycle
* Ensure continuous delivery of high-Quality predictive services.

MLOPs Consists of three phases:-

* Design
* Model development
* Operations.

1. Design phase:-

Understands that Business and data, & then designs the ML-powered Software.

2. Model development phase:-

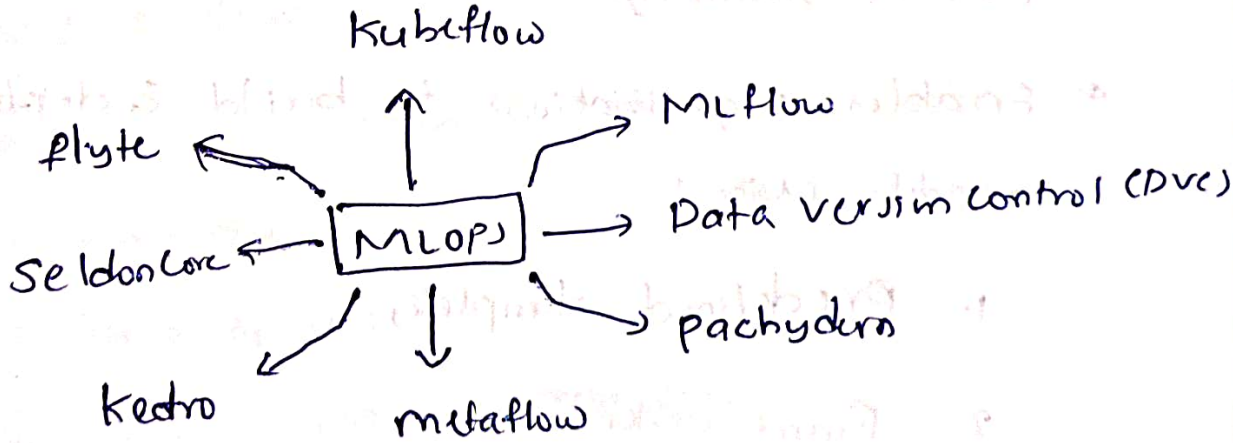
Applicability of ML for the problem is verified by implementing proof of concept (POC).

3. Operations Phase:

Delivers the developed ML model in production

All three MLOPs phase are interconnected & influence one other.

Tools are used for MLOPs :-



→ CI/CD pipeline Automation:-

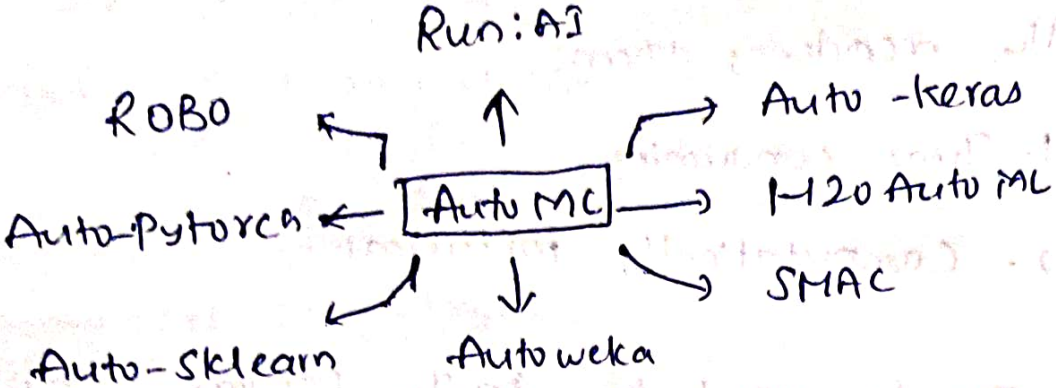
A continuous integration (CI) or continuous Delivery (CD) System is used to:

* Test and deploy new pipeline implementations automatically. By providing rapid, relaiable of the pipelines in productions.
* A CI/CD pipeline automation helps with dynamic changes in the data & Business Environment.

→ Automated Machine Learning (Auto ML):-

* Combines the best practics in automation & machine learning.
* Enables organizations to build & deploy ML models using:
  1. predefined templetes
  2. Frameworks
  3. process to speed up time to completion.
  4. Enhance functionality of ML models.

Tools of AutoML:-



Introduction to Python packages Used in ML:-

* **Python:-**

→ Python is a high-level programming language that helps programmer to write code that is:

1. Clear
2. Organited
3. Logical

for small & large scale projects., released in 1991

→ ~~1991~~, Python gained immense popularity in recent times due to:

1. Ease of use
2. Object-oriented approach
3. code readability owing to significant indentation
4. Evolving open source community.

→ Advent of open-source python packages accelerated the field of data science.

→ Earlier Machine learning (ML), tasks were coded manually, rendering them:

1. Time-consuming
2. Computationally inefficient.

→ Python may be used to easily perform complex ML tasks & Build ML models.

→ Python helps to rapidly build and test software prototypes.

Python packages:-

→ Python packages are folders & modules that form the building blocks in python-based programming.

Python ~~libs~~ libraries:-

→ Python libraries are a collection of packages or specific files containing prewritten code.

1. Numpy
2. Pandas
3. Tensorflow
4. Aesara based on Theano
5. matplotlib
6. Scipy
7. Scikit-learn
8. keras
9. pyTorch

1. Numpy & Pandas helps to manage preparation, loading, & manipulation of data.

2. Tensorflow & Aesara can be used for fast numerical computing.

3. Matplotlib is used to plot the data.

4. Scipy helps solve the mathematical Equations & algorithms.

5. Scikit-learn provids Efficient versions of common algorithms to develop ML models.

6. keras makes the implementation of neural networks Easy.

7. PyTorch Specialises in deeplearning applications & accelarates the path from prototyping to deployment.

Supervised learning:-

* Machines are trained with labelled data as input.
* Machine learning model identify:

Patterns & methods

↓

Learn from them

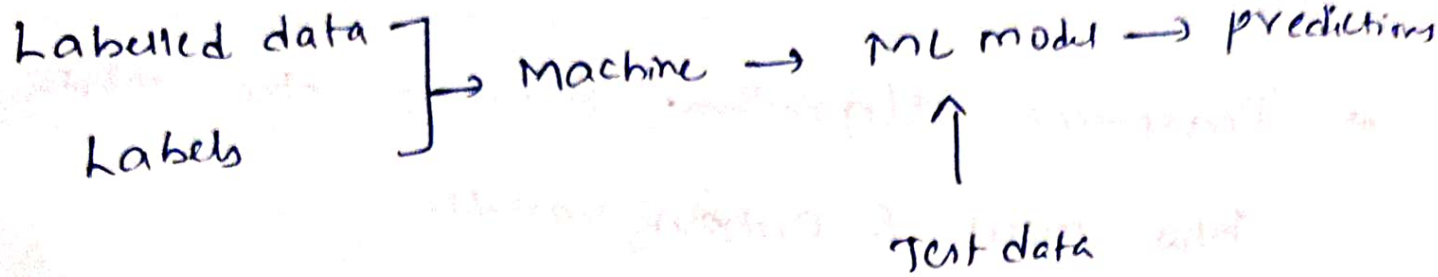
↓

predict output

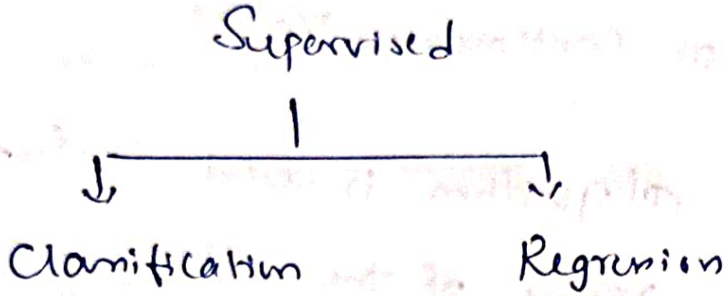
* If any prediction is incorrect, an Operator knows to correct it
* Process continuous until the algorithm achives the highest accuracy

It includes:-

* Linear & logistic regression
* Multi-class classification
* Decision trees
* Support vector machines (SVM)



Types:-



Classification:-

* Classification algorithm segregates data into two or more categories. with one or more inputs, a classification model predicts the value of one or more outcomes.

Ex:- Classification model used to segregate the emails into Spam or Ham.

Regression:-

* Regression algorithm, Estabishes the relationships b/w input & output variables.
* Suitable for situations where the output variable is a real or continuous value.
* Regression algorithm is used to forecast or predict the value of the stock market.

Applications of Supervised learning:-

→ Supervise learning can optimise & automate process

⇒ * Used in HR operations to find the right candidate

* It used in finance to segregate loans are:
  * good loans
  * Bad loans.
* Seafarers identifies sea level trends using Supervised learning.

Supervised learning models are trained to Detect & prevent large scale fraud.

The model learns from:

* Financial transaction with fraud patterns.
* Cyber attacks on servers
* Fake social media profiles.
* Waste mangement:

→ Overfitting & Underfitting:-

* Defines how machine learning model are learning & applying what they learned.
* Let's understand the terms bias & Variance.

Bias:-

- Bias is an Error introduced in a model. (How Error)
- High bias means difference blw the actual & Predicted values. This is not good for model.
- low bias indicate that the difference blw the actual & predicted value is low.

Variance:- It indicate how scattered data is.

High Variance :- More scattered data

Low variance :- less scattered data.

	bias	Variance
overfit	low	high
underfit	High	High.

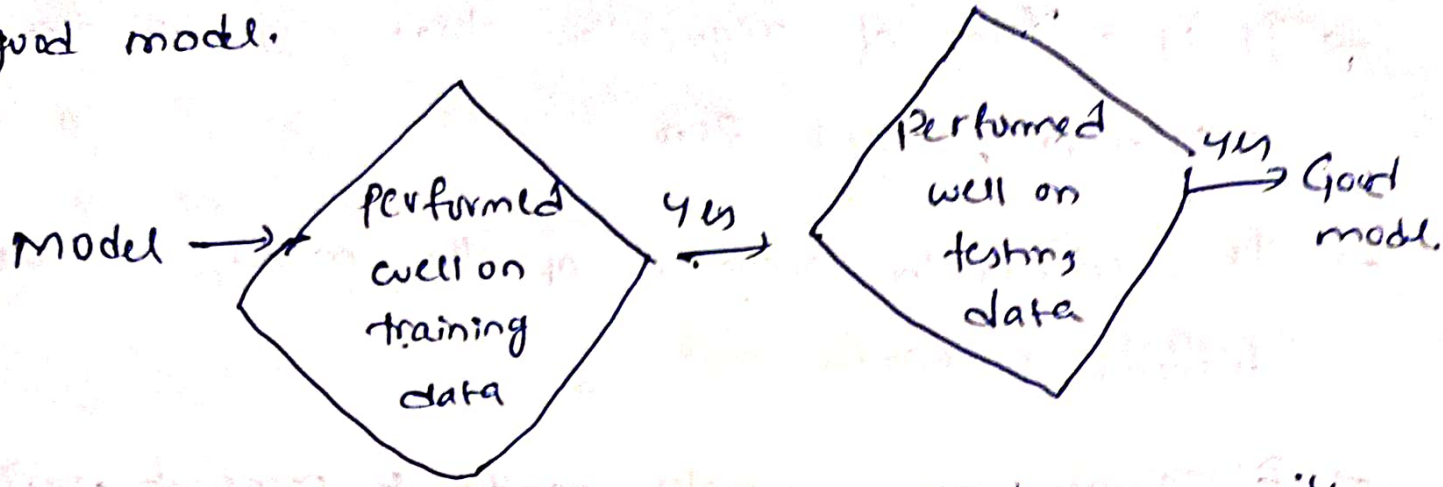
→ Overfit:- It indicates low bias & high variance in the data.

* It happens when a model focuses on too many details in the training dataset.
* has a negative impact on the performance of the model on a new dataset.

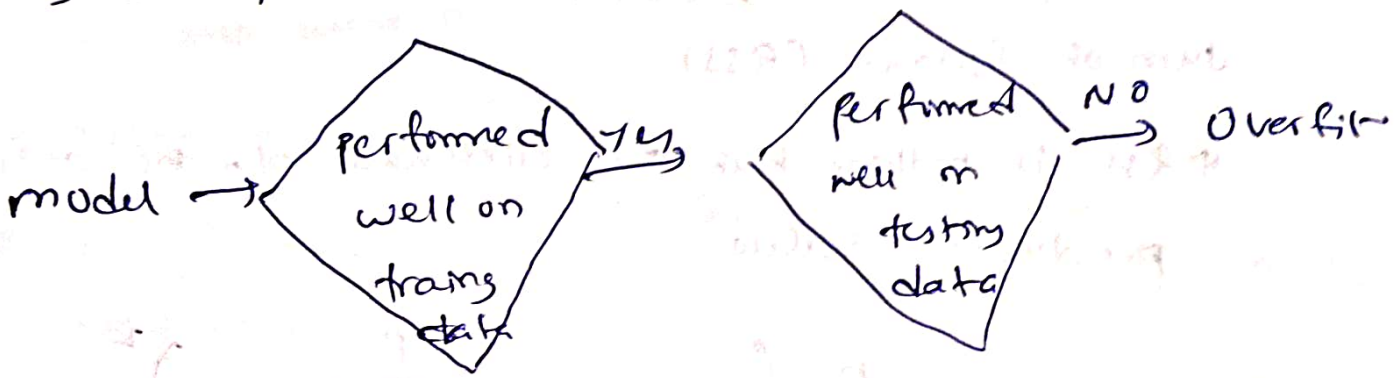
→ Underfit:- It is high Bias & high variance in the data.

* Underfitting is detectable as a it Exhibit poor performance on the training dataset.
* A model is underfit if it is trained with limited features like temperature & wind speed.

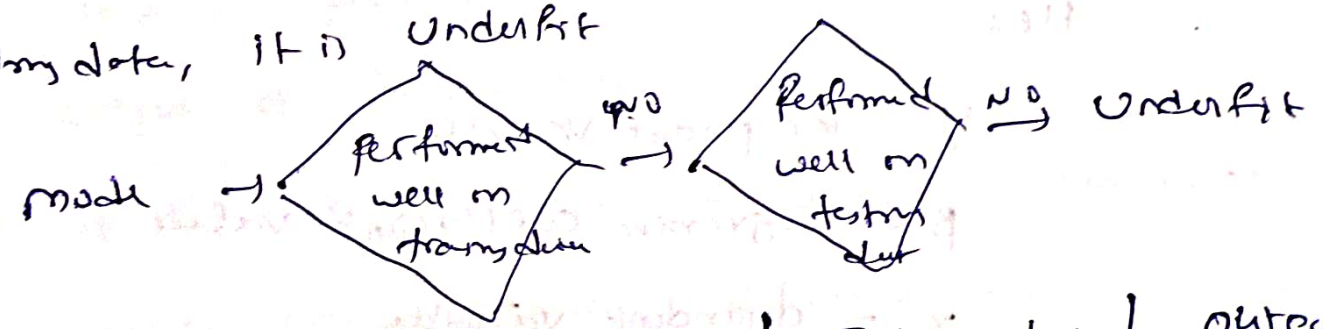
If model performs well on traing data & testing data, it good model.



If model performs well on traing data but not with testing data, it is overfit.



If model doesn't performs well on both training data & testing data, it is Underfit



	Traing data	Testing data	Output
model	✓	✓	Good
model	✓	x	over fit
model	x	x	under fit

Regularization:-

→ It is a form of regression that shrinks the coefficient towards zero.

→ It reduces the variance of the model, without increase in bias.

→ Discourages more complex models & prevents overfitting.

→ Fitting involves a loss function and known as residual sum of squares (RSS).

* RSS is nothing but the difference b/w the actual & predicted value.

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Here,

y - independent variable

β - regression coefficient value

x - dependent variable

Types of Regularitation:-

* Dropout Regularization:-
  * It works by removing a random selection.
  * The more units dropout, the stronger it regularization.
  * This is good for training nerual Networks.

* Early Stopping:-
  * It uses large number of Epochs & plot the validation loss graph.
  * Stop training and save the model when the validation loss moves from decreasing to increasing.

* Co-adaptation:-
  * Neurons predict patterns in the training data using output of specific neurons.
  * If the validation data doesn't have patterns that causes co-adaptation, it could cause overfitting.
  * Dropout regularitation reduces co-adaption.
  * It Ensures neurons cannot rely solely on any neurons.

* Losses Regression - (L1 Regularization)

  * It penalizes weights in proportion to the sum of the absolute values of the weights.

  * Drives the weights of irrelevant or barely relevant features to Exactly zero.

* Ridge Regression : (L2 Regularization)

  * Ridge Regression shrinks coefficients close to zero for unimportant predictors.

  * never make them Exactly Zero.

  * In other words, the final model will include all predictors

Linear Regression Equation:-

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

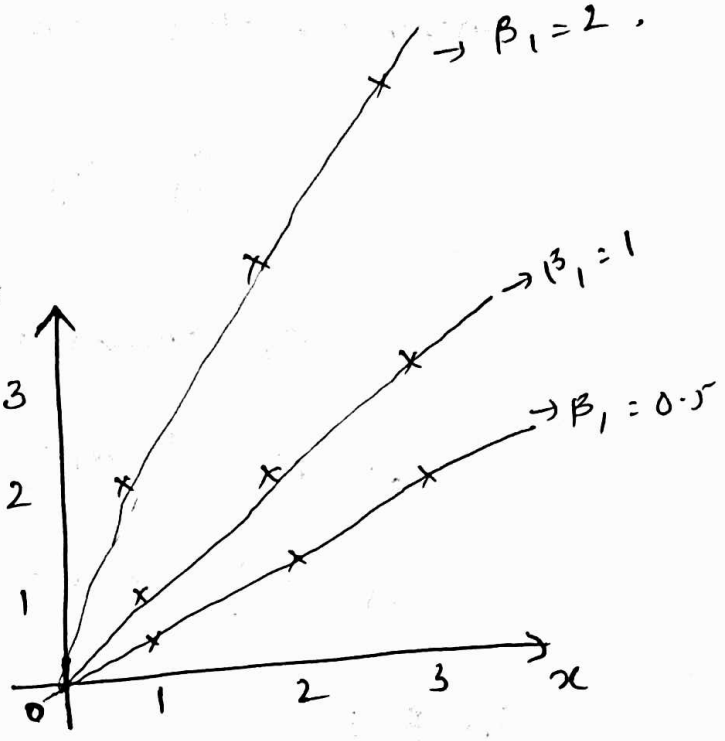
Here,

→ $\beta_0 = 0$, $\beta_1 = 1$, $x = \{1, 2, 3\}$

⇒ $x = 1$, $\hat{y} = 0 + 1(1) = 1$

⇒ $x = 2$, $\hat{y} = 0 + 1(2) = 2$

⇒ $x = 3$, $\hat{y} = 0 + 1(3) = 3$



→ $\beta_0 = 0$, $\beta_1 = 0.5$, $x = \{1, 2, 3\}$

⇒ $x = 1$, $\hat{y} = 0 + 0.5(1) = 0.5$

⇒ $x = 2$, $\hat{y} = 0 + 0.5(2) = 1$

⇒ $x = 3$, $\hat{y} = 0 + 0.5(3) = 1.5$

→ $\beta_0 = 0$, $\beta_1 = 2$, $x = \{1, 2, 3\}$

⇒ $x = 1$, $\hat{y} = 0 + 2(1) = 2$

⇒ $x = 2$, $\hat{y} = 0 + 2(2) = 4$

⇒ $x = 3$, $\hat{y} = 0 + 2(3) = 6$

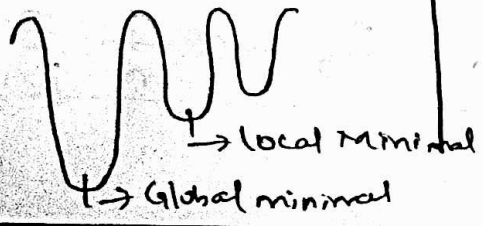
Mean Square Error (or) Cost function:-

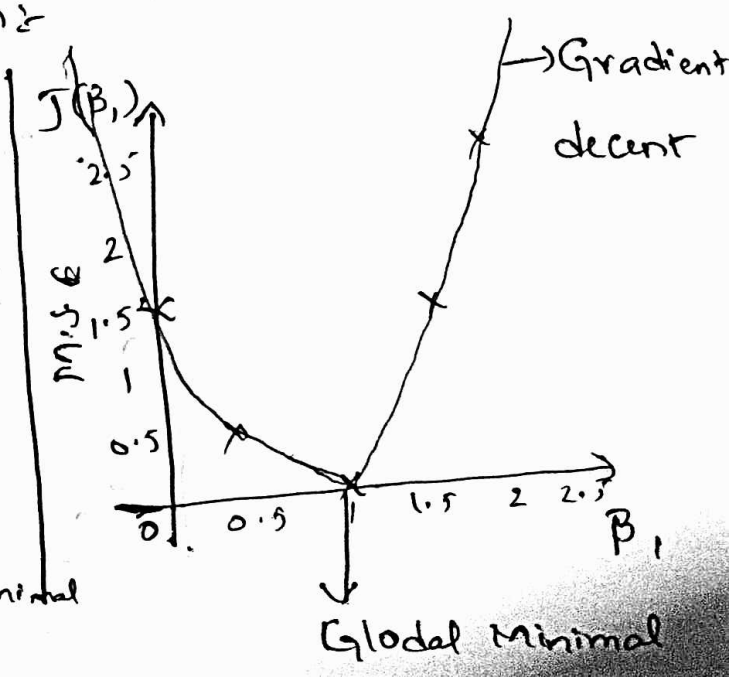
$$J(\beta_1) = \frac{1}{2m} (\hat{y}_i - y_i)^2$$

$$J(1) = \frac{1}{2(3)} \left[(1-1)^2 + (2-2)^2 + (3-3)^2 \right] = \frac{0}{6} = 0$$

$J(0.5) = 0.58$

$J(2) = 2.5$





Gradient Descent Optimization:-

Minimize the cost function $J(\beta)$ (MSE).

$$\theta_j : \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

→ θ_j = Model parameters (β_0, β_1)

→ α = Learning rate

→ $\frac{\partial J}{\partial \theta_j}$ = partial derivative (Gradient).

in β_j's,

$$\beta_j = \beta_j - \alpha \frac{\partial J}{\partial \beta_j} \quad \left[\because J = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2\right]$$

$j = 0$

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

$$\left[\because \hat{y}_i = \beta_0 + \beta_1 x_i\right]$$

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

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

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

$$\boxed{\beta_0 = \beta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left[\beta_0 \hat{y}_i - y_i\right]}$$

$j=1$,

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

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

$$= \beta_1 - \alpha \frac{1}{2m} 2 \sum_{i=1}^{m} \left[(\beta_0 + \beta_1 x_i) - y_i \right] x_i$$

$$\beta_1 = \beta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[(\beta_0 + \beta_1 x_i) - y_i \right] x_i$$

$$\boxed{\beta_1 = \beta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_i}$$

$$\beta_j = \beta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) \cdot x_j^{(i)} \quad \text{for all } j \in \{1, \ldots n\}.$$

Logistic Regression :-

Cost function :-

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

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

- True class label

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

Here, m = number of training Examples or observations

y_i = Actual class label (0 or 1)

$\hat{y}_i$ = predicted probability (from Sigmoid function)

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:

$$L(\beta) = \prod_{i=1}^{n} P(y_i/\hat{y}_i) \qquad [\because \log(a^b) = b\log a]$$

$[\because \log(ab) = \log a + \log b]$

Taking log on Both sides,

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

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

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

Negative log-liked,

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

Gradient Descent Optimization :-

Minimize the cost function $J(\beta)$ - M.S.E.

$$\theta_j : \theta_j - \alpha \frac{\partial J}{\partial \theta_j} \quad \left[\because \hat{y}_i = h_\beta(x_i) = \frac{1}{1+e^{-x_i \beta}} \right]$$

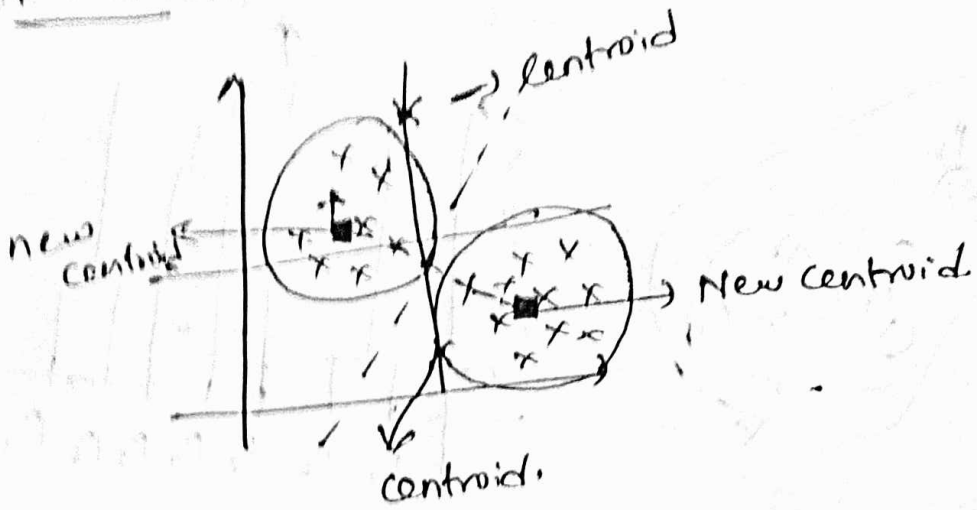
$$\beta_j = \beta_j - \alpha \frac{\partial J}{\partial \beta_j}$$

$j = 0$,

$\beta_j = \beta_0 \Rightarrow \beta_0 = \beta_0 - \alpha \frac{\partial}{\partial \beta_0} \left[-\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i) \right]$

$$\beta_0 = \beta_0 - \alpha \frac{\partial}{\partial \beta_0} \left[-\frac{1}{m} \sum_{i=1}^{m} \left[y_i \log(h_{\beta_0} x_i) + (1-y_i) \log(1-h_{\beta_0} x_i) \right] \right]$$

K-means (Large Dataset)



Euclidean Distance

$$d(p,q) = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

→ centroids

1. We this K values → Suitable K = 2
2. Initialize K - number of centroids
3. Compare the AVG to update centroids

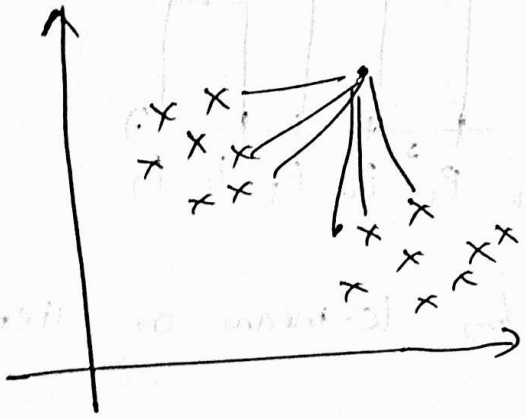
Deciding K-Value :-

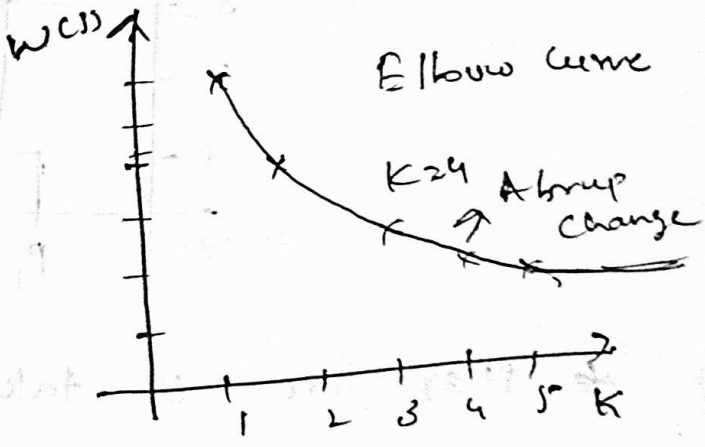
→ validating purpose

Elbow method (K value)

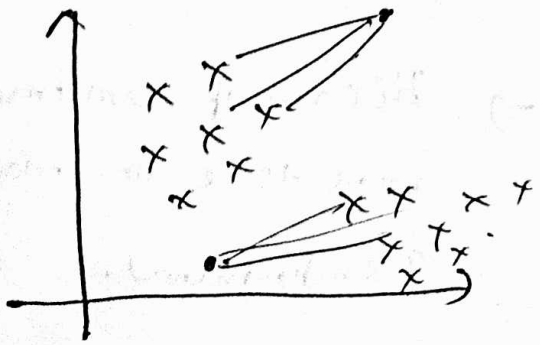
WCSS - within Cluster Sum of Square

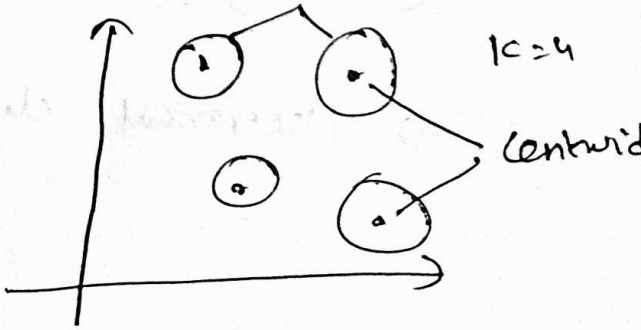
K = 1





K = 2





K-means :-

Point	X	Y	Nearest Centroid	Cluster	Centroid
A	1	2	centroid 1	1	1.5
B	2	1	centroid 1	1	1.5
C	4	5	centroid 2	2	4.5
D	5	4	centroid 2	2	4.5

Step-1 :- K = 2, (Assume)

Centroid-1 → (1, 2) (Same as point A)

centroid-2 → (5, 4) (Same as point B)

Finally

Cluster 1, (1, 2)
(2, 1)

Cluster 2, (5, 5)
(5, 4)

Centroids 1 (1.5, 1.5)
2 (4.5, 4.5)

Step-2 :- Calculate Euclidean Distance from Each point to Each centroid.

$$d(P, Q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Point A(1, 2) : x_1 y_1

* Centroid 1 (1, 2) : x_2 y_2 $d = \sqrt{(1-1)^2 + (2-2)^2} = 0$

* Centroid 2 (5, 4) : $d = \sqrt{(1-5)^2 + (2-4)^2} = \sqrt{20} = 4.47$

∴ point A closer to centroid-1

Point B(2, 1)

* Centroid 1 - (1, 2): $d = \sqrt{(2-1)^2 + (1-2)^2} = \sqrt{1+1} = \sqrt{2} = 1.41$
* Centroid 2 - (5, 4): $d = \sqrt{(2-5)^2 + (1-4)^2} = \sqrt{9+9} = \sqrt{18} = 4.24$

∴ point B is closer to Centroid - 1

point C(4, 5)

* Centroid 1 - (1, 2): $d = \sqrt{(4-1)^2 + (5-2)^2} = \sqrt{9+9} = \sqrt{18} = 4.24$
* Centroid 2 - (5, 4): $d = \sqrt{(4-5)^2 + (5-4)^2} = \sqrt{1+1} = \sqrt{2} = 1.41$

∴ point C is closer to centroid - 2

point D(5, 4)

* Centroid 1 - (1, 2): $d = \sqrt{(5-1)^2 + (4-2)^2} = \sqrt{16+4} = \sqrt{20} = 4.47$
* Centroid 2 - (5, 4): $d = \sqrt{(5-5)^2 + (4-4)^2} = 0$

∴ point D is closer to Centroid - 2

Step-3:

Point	Nearest centroid	Cluster
A	Centroid 1	Cluster-1
B	Centroid 1	Cluster-1
C	Centroid 2	Cluster-2
D	Centroid 2	Cluster-2

Step-4 : Update Centroids:

Now, we calculate the new position of each Centroid by finding the mean (average) of points in its cluster.

→ Cluster 1 : points A(1,2) & B(2,1)

New centroid 1 :

* $X = (1+2)/2 = 1.5$

$Y = (2+1)/2 = 1.5$

New centroid 1 = (1.5, 1.5)

→ Cluster 2 : points C(4,5) & D(5,4)

New centroid 2 :

$$X = \frac{(4+5)}{2} = 4.5$$

$$Y = \frac{(5+4)}{2} = 4.5$$

New centroid 2 = (4.5, 4.5)

Step-5 - Recalculate Distance (Repeat step-2)

⇒ • Distance from A(1,2) & new centroid-1 (1.5, 1.5)

$$d = \sqrt{(1-1.5)^2 + (2-1.5)^2} = \sqrt{0.25+0.25} = \sqrt{0.5}$$

$$\approx 0.71$$

Centroid 2 (4.5, 4.5)

$$d = \sqrt{(1-4.5)^2 + (2-4.5)^2} = \sqrt{12.25 + 6.25} = 4.30116.$$

∴ Point A is closer to new centroid-1

⇒ Point B(2,1) $c_1 = \sqrt{(2-1.5)^2 + (1-1.5)^2} = \sqrt{0.5} = 0.71$

$$c_2 = \sqrt{(2-4.5)^2 + (1-4.5)^2} = 4.30$$

∴ point B is closer to new centroid-1

⇒ point C(4,5) $= \sqrt{(4-1.5)^2 + (5-1.5)^2} = \sqrt{6.25 + 12.25} = 4.3016$

$$= \sqrt{(4-4.5)^2 + (5-4.5)^2} = \sqrt{0.5} = 0.71$$

∴ point C is closer to new centroid-2

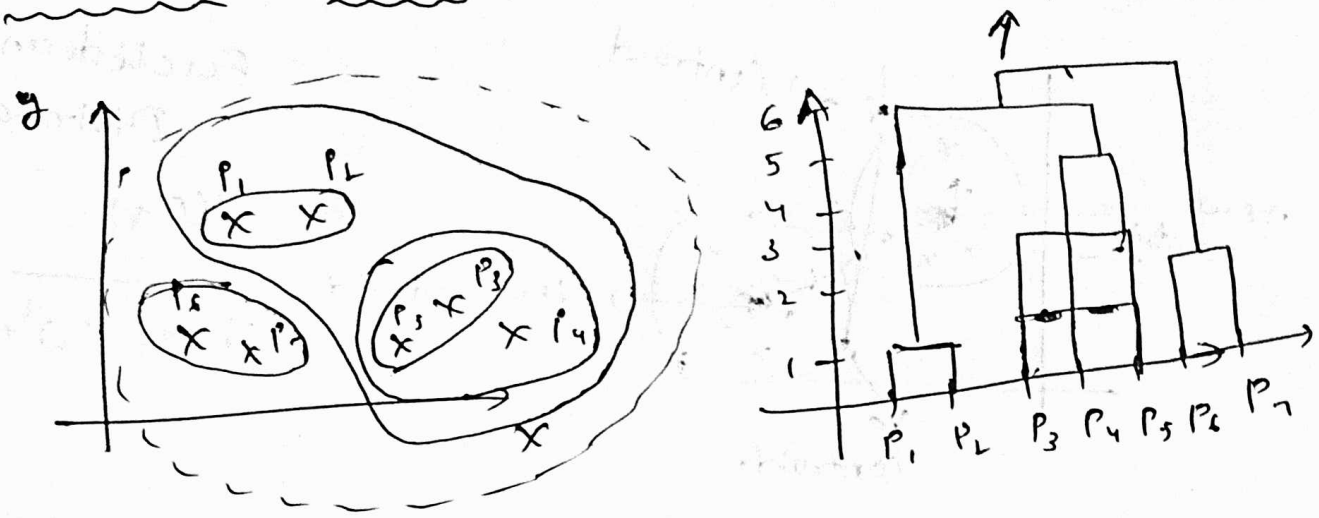
⇒ point D(5,4) $= \sqrt{(5-1.5)^2 + (4-1.5)^2} = 4.3016$

$$= \sqrt{(5-4.5)^2 + (4-4.5)^2} = 0.71$$

∴ point D is close to new centroid-2.

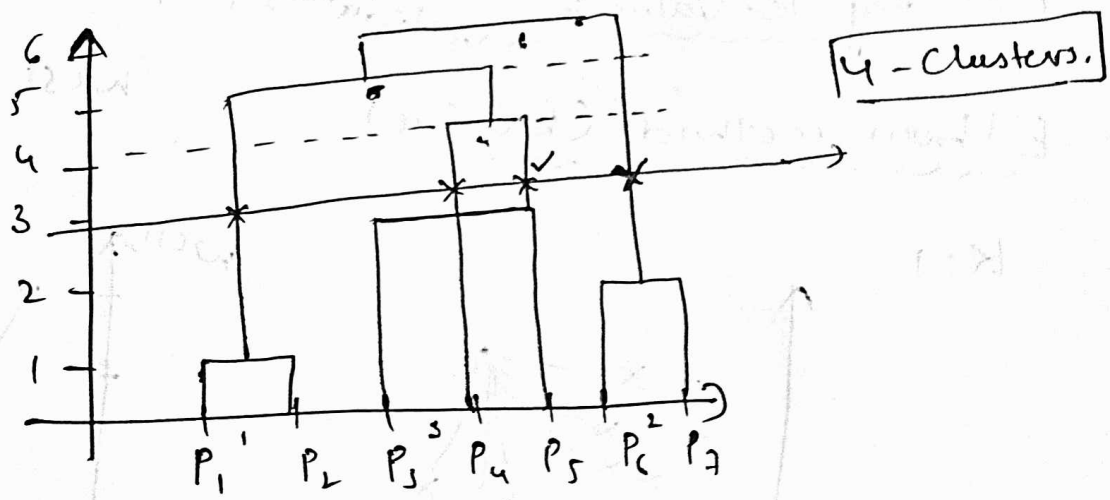
∴ Cluster anignement remain the same. So the algorithm stop.

2. Hierarical Clustering:- (Small dataset)



Dendogram - Bottoum root to top.

* You need to find out the longest vertical line that has no horizontal line passed through it.



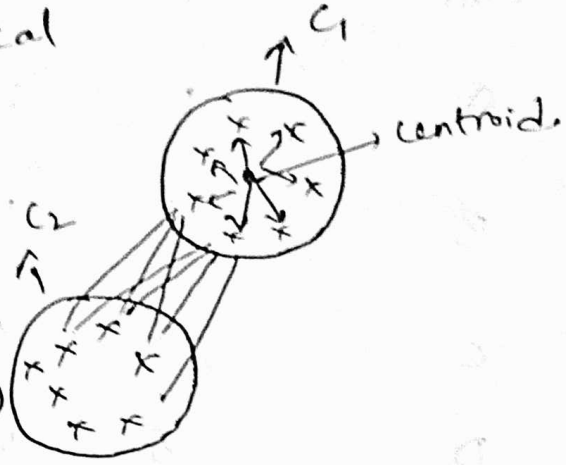
* Max time is taken by K-means or Hierarichal Clustering !!

→ Hierarical clustering → Because of constructing max time to make Dendogram.

Validate Clustering models:

* Silhouette Score → K-means
  → Hierarchical

→ $$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i,j)$$



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

Good clustering model:
$a(i) >> b(i)$ or $b(i) >> a(i)$

→ value b/w −1 to +1

+1 → good model

−1 → bad model

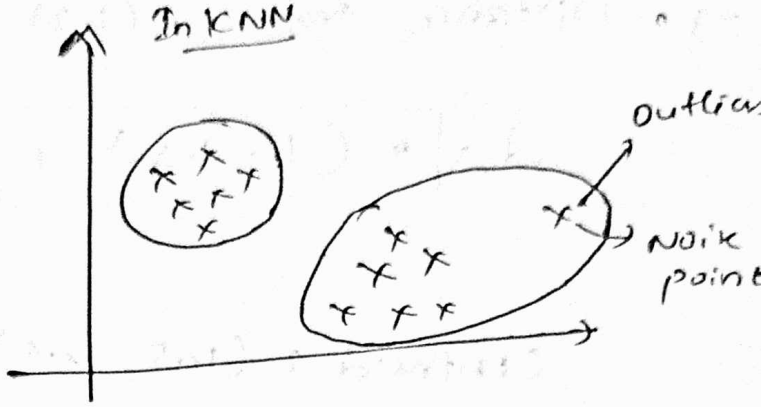
0 → Clusking need to be improve

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

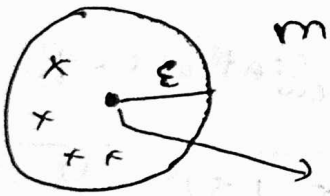
⇒ −1 to +1

DBScan Clustering is (Density Base Spatial Clustering Application with Noise (DBSCAN))

1. Min point
2. Core point
3. Boarder point
4. Noise point



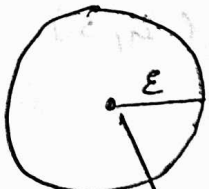
* Min points → kind of hyperparameter
  - ↳ called as Epsilon. → radius of circle.



min point = 4

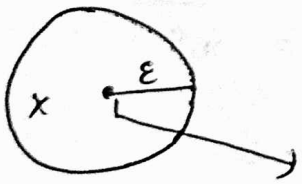
core point.

(Assume min points are present in cluster is called core point)



→ Negated (outliner)

Noise point



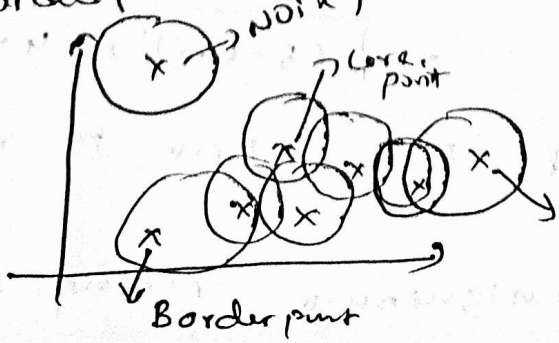
Boarded point

↓

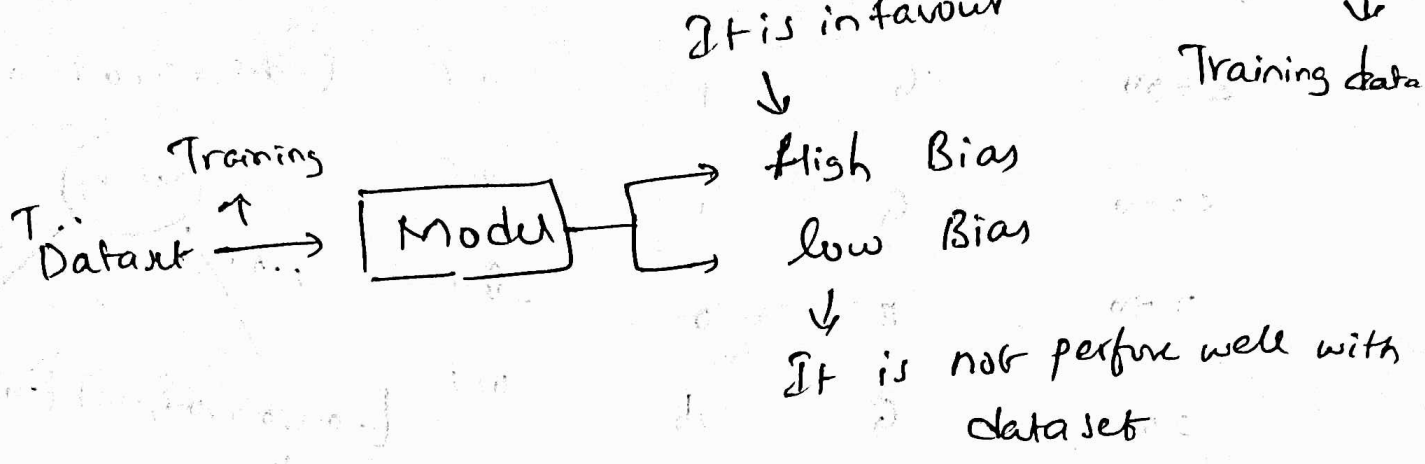
Cluster with one ~~point~~ core point (atleast) is called Boarded point

↓

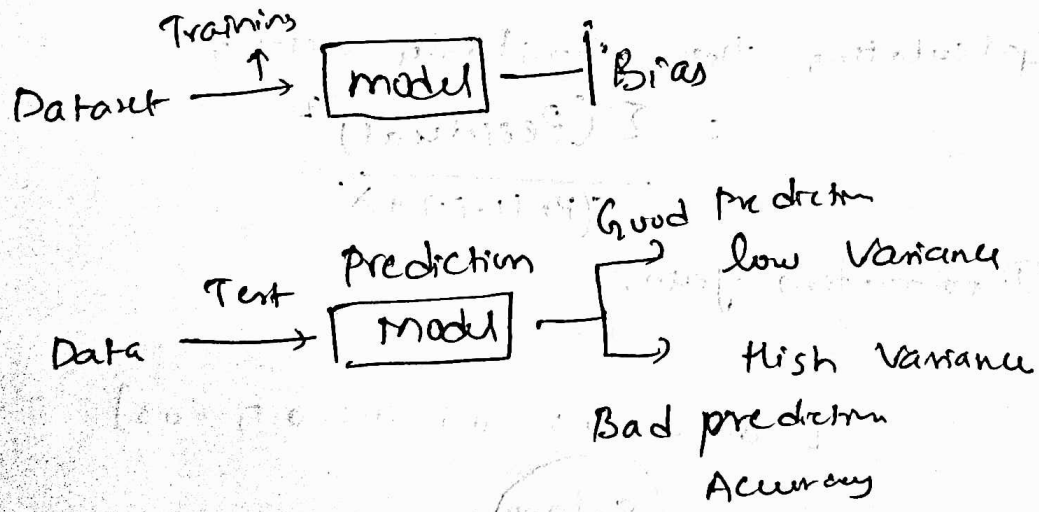
Cluster with No points is called Noise point



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



Variance:- Variance refer to 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 AC = 60%

Test AC = 55%

↓

{ high Bias, high variance }

Model 3

Train ACC = 90%

Test ACC = 92%

↓

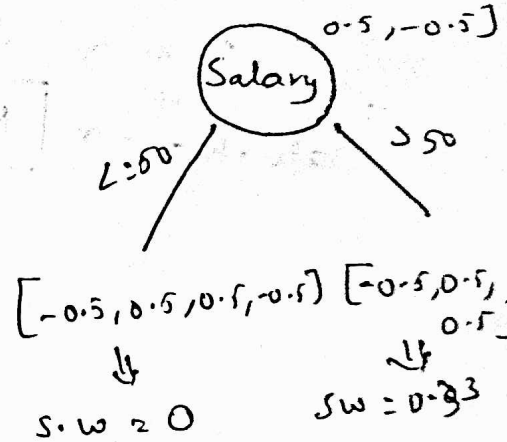
{ low Bias, low variance }

↓

Generalized model

XGBoost Classifier:- Extreme Gradient Boosting

Salary	Credit	Approval	Residual (Pr - App)	Base model → Pr = 0.5
<250	B	0	-0.5	
<250	G	1	0.5	[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]
<250	G	1	0.5	
>50	B	0	-0.5	
>50	G	1	0.5	
>50	N	1	0.5	
<250	N	0	-0.5	



Step-1 :- Create a Binary Decision Tree using the feature.

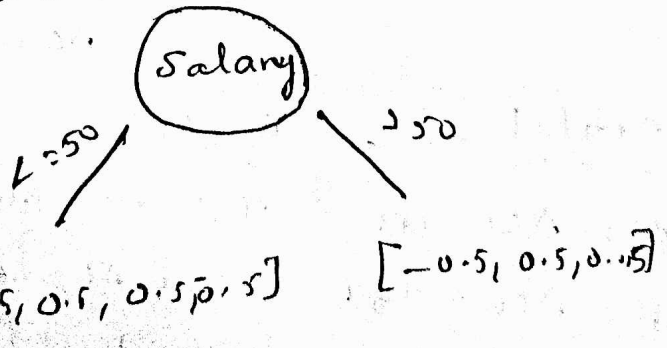
Step-2 : Calculating the similarity weight

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

Step-3 : Information gain.

Step-1 :-

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



Similarly weight $\lambda = 0$ → kind of Hyperparameter

Step-2 :- Residuals

$$\text{Similarity weight} = \frac{[-0.5 + 0.5 + 0.5 - 0.5]^2}{0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5) + 0}$$

$$= 0$$

$$S.W\ 2 = \frac{[-0.5 + 0.5 + 0.5]^2}{0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5)}$$

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

$$S.W.\ \text{root Class} = \frac{[-0.5 + 0.5 + 0.5 + 0.5 + 0.5 - 0.5]^2}{0.25 + 0.25 + 0.25 + 0.25 + 0.25 + 0.25}$$

$$= \frac{0.25}{1.75} = 0.142$$

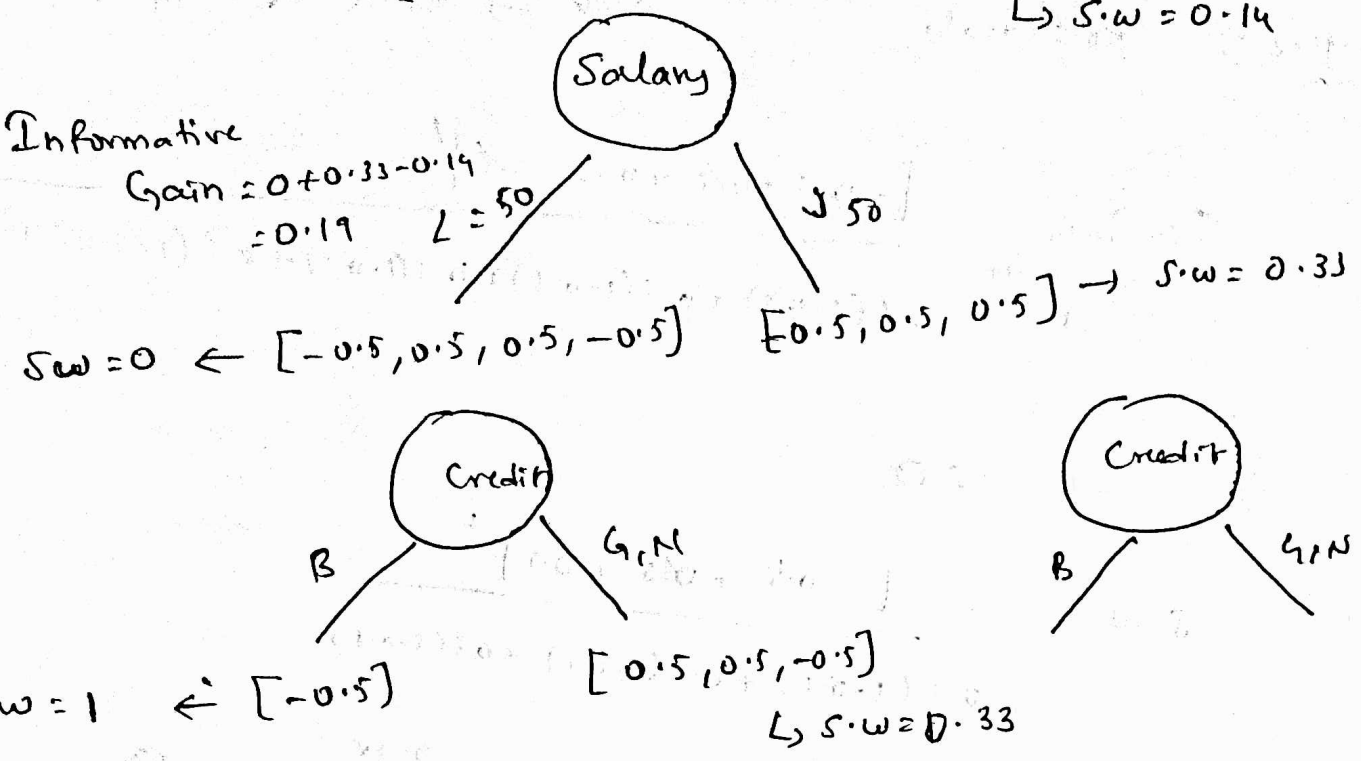
Step-3 :- Informative Gain = S.W-1 + S.W-2 − S.W root

$$= 0 + 0.33 - 0.14$$

$$= 0.19$$

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

↳ S.W = 0.14



Sw = 0

Sw = 1

Informative Gain = 0 + 0.33 − 0.14 = 0.19

↳ S.W = 0.33

Information gain = 1 + 0.33 − 0 = 1.33

for first value in the table

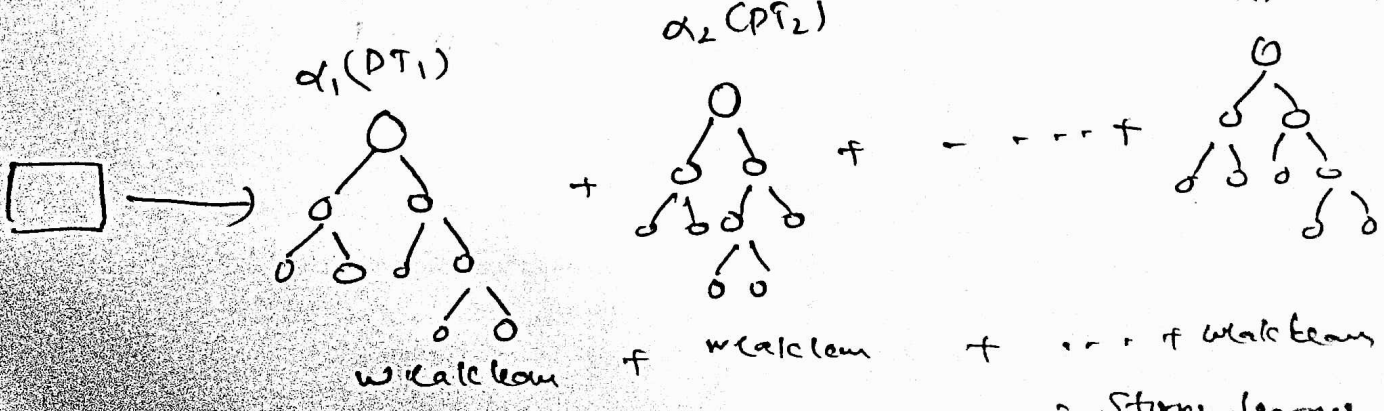
$\sigma[0 + \alpha(1)]$

α – Learning rate

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

XG Boost – Blacle Box Model

→ pre purning



= Strong learner.

XG Boost Regressor :-

Exp	Gap	Salary	R_1
2	Yes	40K	-11K
2.5	Yes	42K	-9
3	NO	52K	1
4	NO	60K	9
4.5	Yes	62K	11

Avg of Salary.

① Base model → 51 lc

[-11k, -9, 1, 9, 11] → s.w = 4/6

Exp

<2 : [-11] s.w = 60.5 + 2

>2 : [-9, 1, 9, 11] s.w = 28.5

2.4 = 89.13

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

Here, $\lambda = 0$ → Hyperparameter Tuning

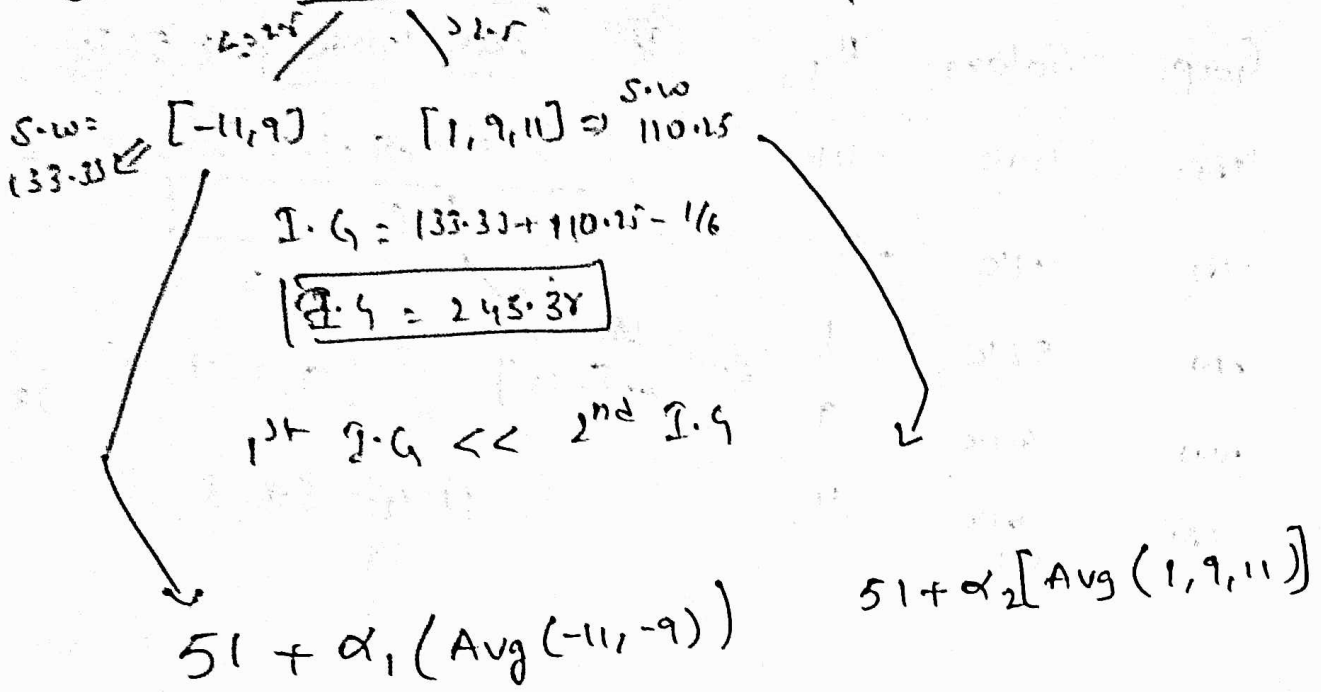
$= \frac{[-11]^2}{1+0} = \frac{121}{2} = 60.5$ | $\frac{[-9+1+9+11]^2}{4+1} = \frac{144}{5} = 28.5$

root = $\frac{[-11-9+1+9+11]^2}{5+1} = \frac{1}{6}$ =

Information Gain = 60.5 + 28.8 − 1/6 = 89.13

$[-11, -9, 1, 9, 11]$: S.w 1/6

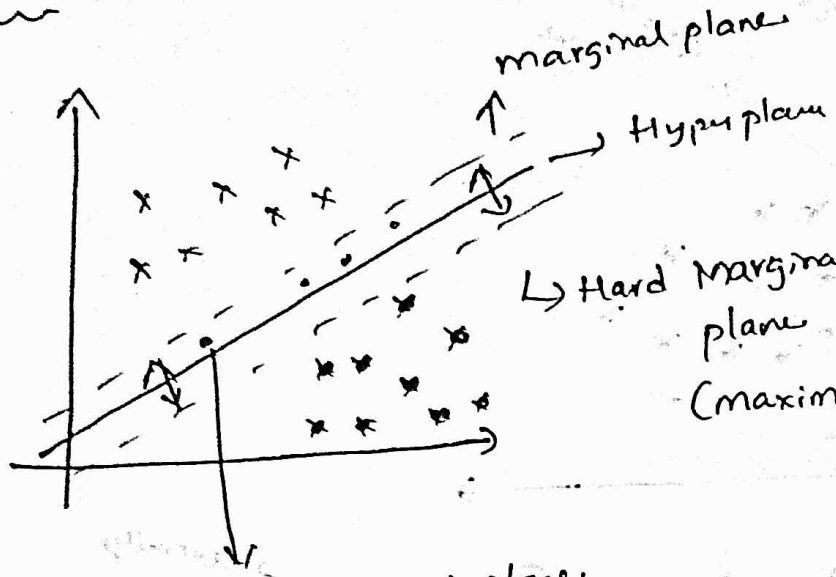
② Exp

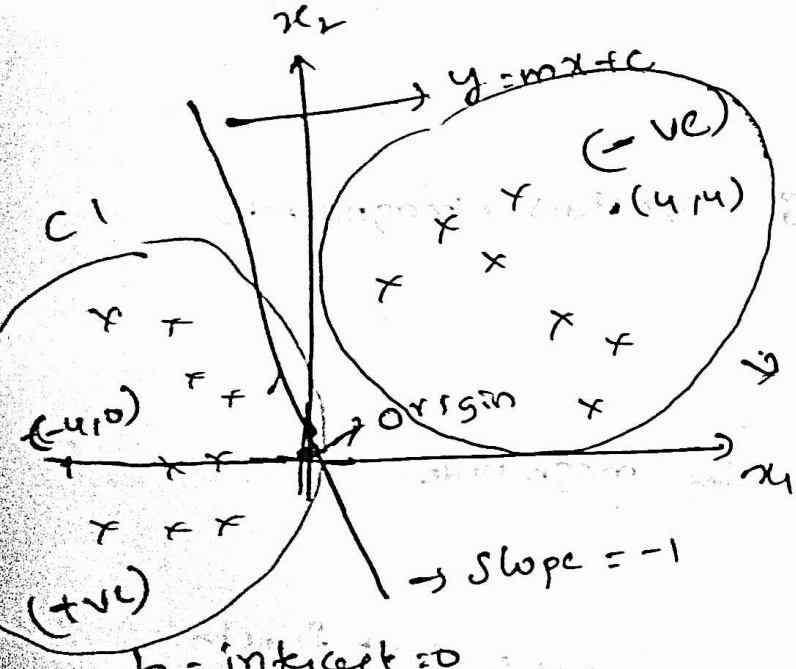


$O/P = \text{Base model Avg.} + \alpha_1(DT_1) + \alpha_2(DT_2) + \cdots + \alpha_n(DT_n)$

→ It is also a Black Box Model.

SVM





b = intercept = 0

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

$$= 4 \quad +ve \text{ Value}$$

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

$$= -4 + 0 = -4 \quad -ve \text{ Value}$$

Slope

$y = mx + c$ → Intercept

$ax + by + c = 0$ → Eq of Straight Line

$by = -ax - c$

$y = -\frac{a}{b}x - \frac{c}{b}$

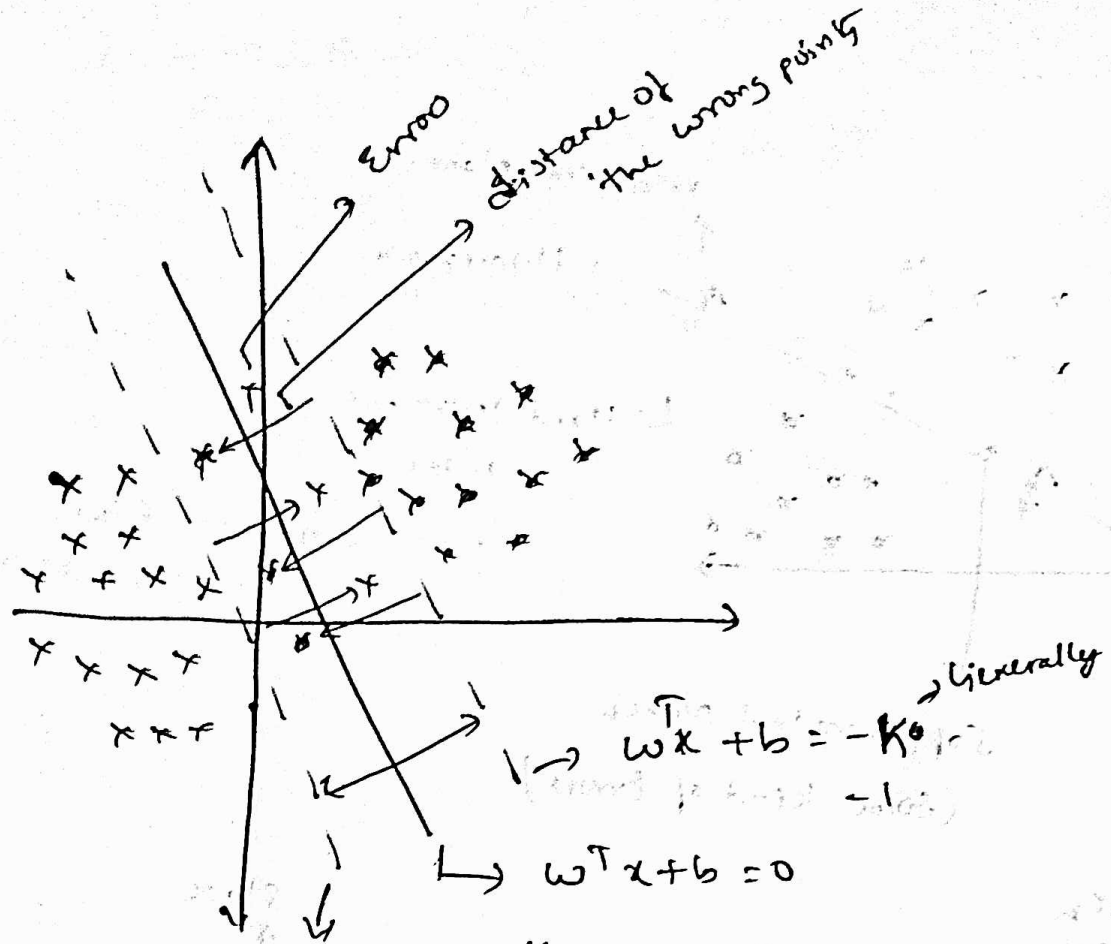
↓ Slope ↓ Intercept

$m = -a/b \qquad c = -\frac{c}{b}$

$y = mx + c$

$y = w_1 x + w_2 x + \cdots + b$

$\therefore y = w^T x + b$



$\vec{w}$ Vectors + magnitude

$w^T x_1 + b = 1$

$w^T x_2 + b = -1$

$(-)$ $(-)$ $(+)$

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

$\frac{w^T}{|w|}(x_1 - x_2) = \frac{2}{|w|}$ → dividing magnitude

$\Rightarrow$ maximize $\frac{2}{\|w\|}$ $\Rightarrow$ maximize cost function

(w, b)

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

major Aim $y_i * (w^T x_i + b) \geq 1$

for correct point

$$\underset{(w,b)}{\text{maximum}} \ \frac{2}{\|w\|} \Rightarrow \underset{(w,b)}{\min} \ \frac{\|w\|}{2}$$

Regularization

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

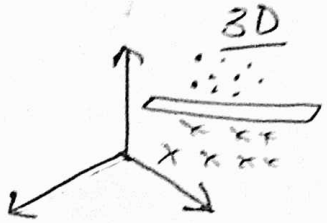
ξ_i → Extra value of the Error

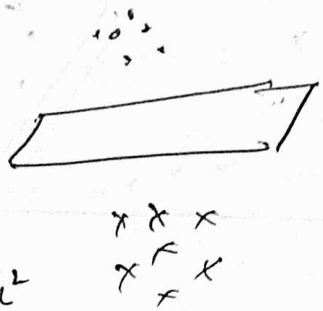
$\sum_{i=1}^{n}$ → Sum of distance of the wrong data points

C_i → How many Errors we can have

SVM kernel

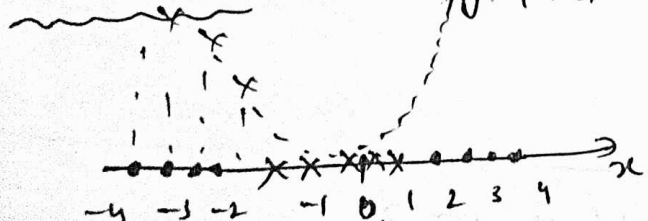
Kernel converts low dimension to high dimensional



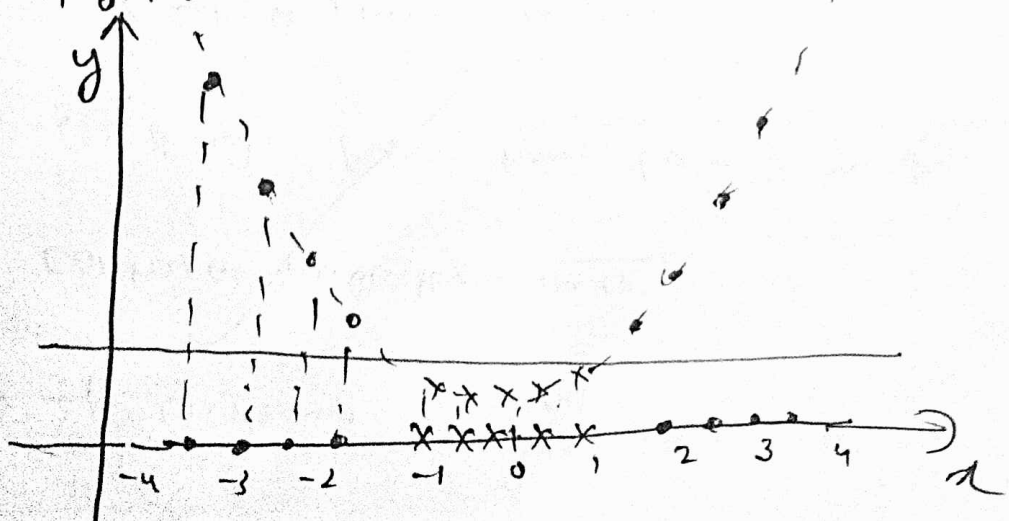


1 - Dimension :-

$y = f(x) = x^2$



1D → 2D



Polynomial kernel:

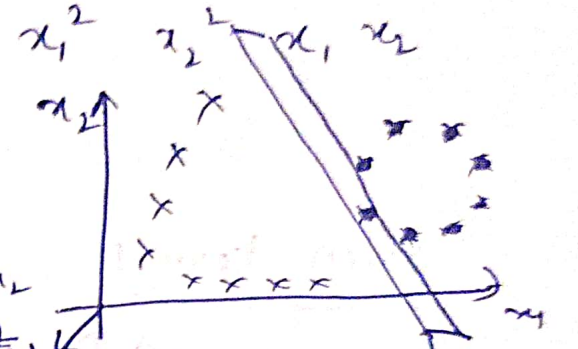
data x_1 x_2 y_0 $\quad y = f(x_1, x_2)$

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

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

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

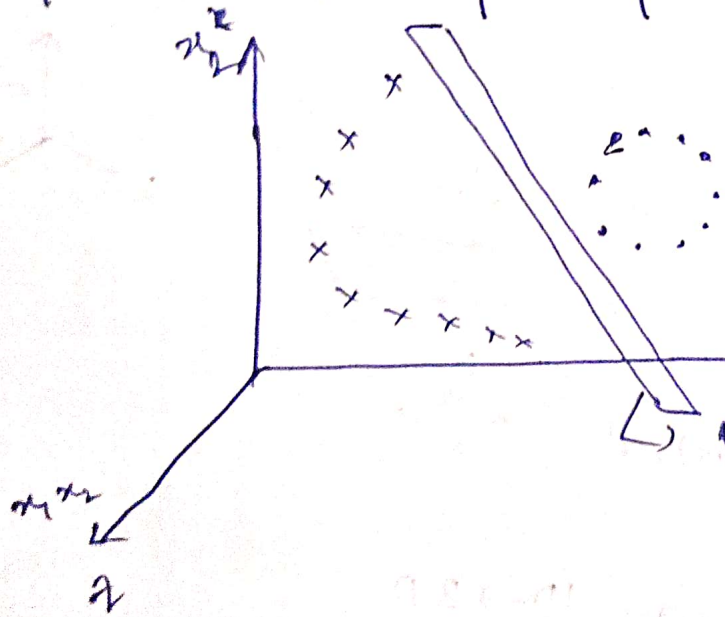
Find out



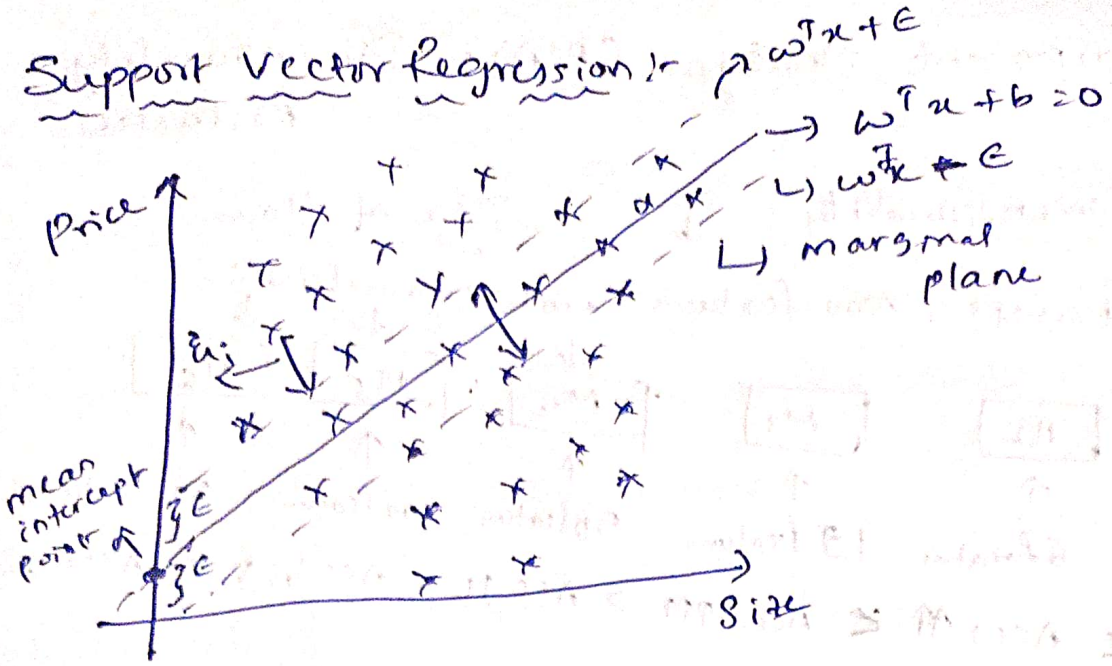
→ x_1 | x_2 | y_0 | x_1^2 | x_2^2 | $x_1 x_2$ → features

↓

Using this feature to convert low dimensional to high



Support Vector Regression :-



Cost function :-

$$\min_{(w,b)} \frac{\|w\|}{2}$$

Constraint $|y_i - w^T x_i| \leq \epsilon$ → It is very good.

$\downarrow$

$\hat{y}$

Cost function : $\min_{(w,b)} \frac{\|w\|}{2} + C_i \sum_{i=1}^{w} |\xi_i|$ → Hinge loss.

↳ Hyperparameter

Constraint :-

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

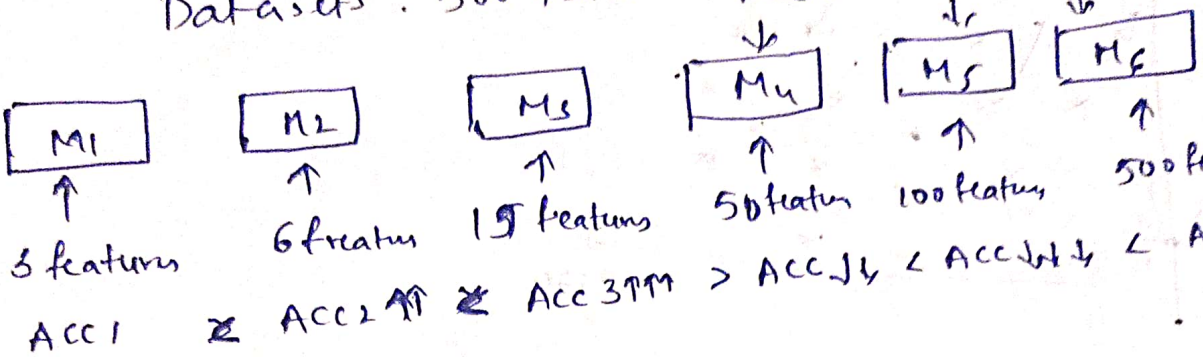
Principal Component Analysis (PCA) :- Dimensionality Reduction

1. Curse of Dimensionality :- ↓

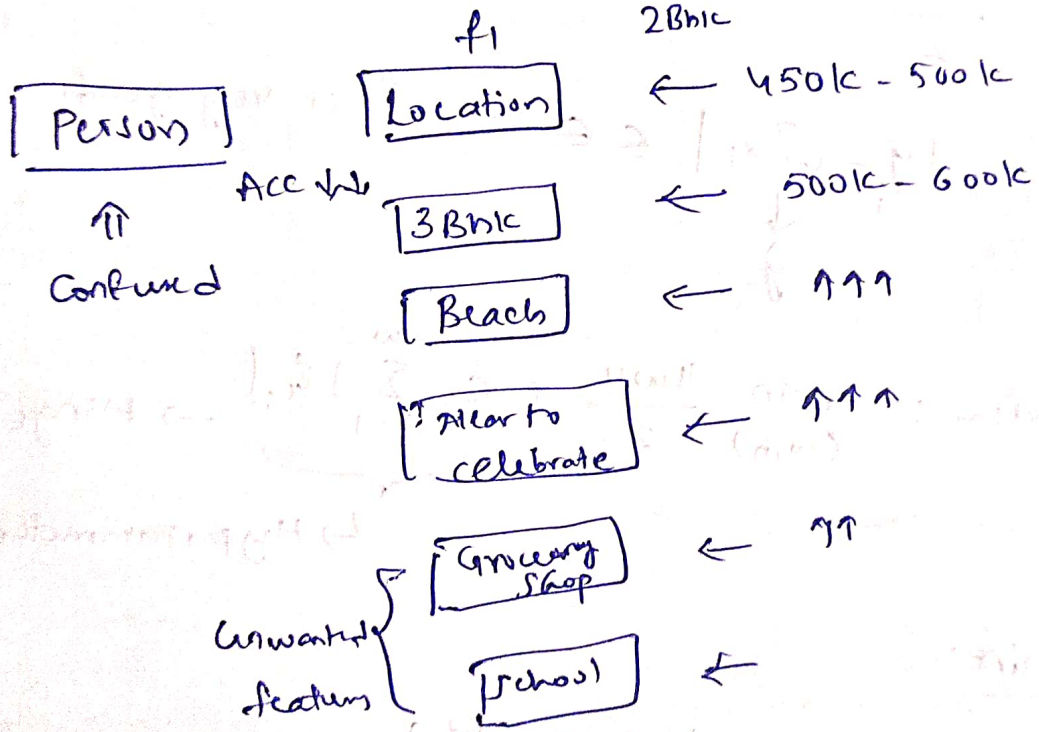
Datasets : 500 features

Price of House

model is overfitted



2. Model performance Degrade - when we use more features (Dimensions)



Two different ways to remove curse of Dimensionality

1. Feature selection
   ↓
   takes imp features

2. Dimensionality Reduction
   ↓
   Feature Extraction

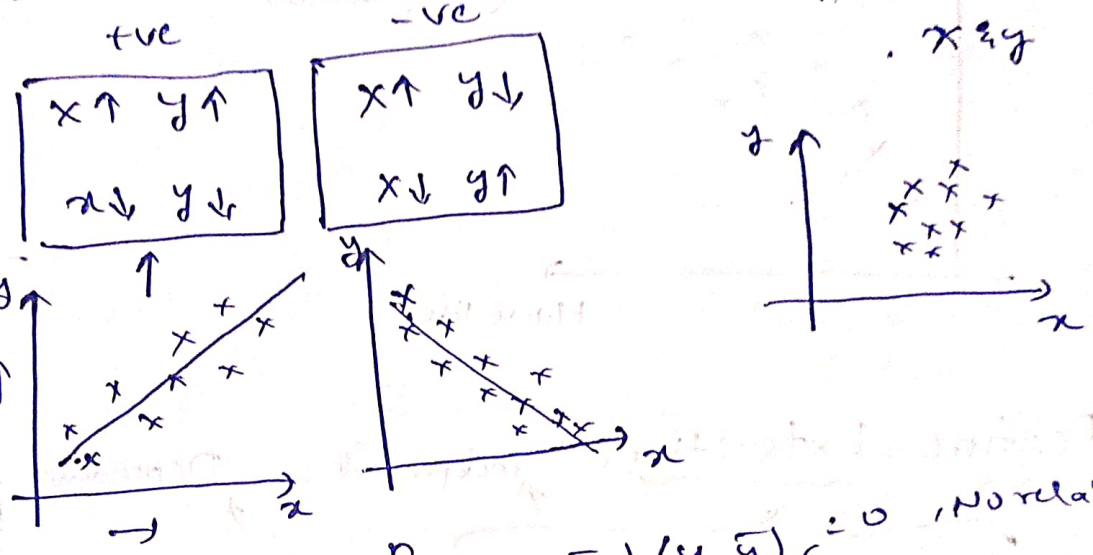
Feature Selection Vs Feature Extraction:- → Dimensionality Reduction.

1. Why Dimensionality Reduction?

* Prevent → Curse of Dimensionality
* Improve the performance of the model
* Visualize the data → Understand the data.

Feature Selection:-

I/p	O/p
X	Y
-	-
-	-
-	-
-	-



* Covariance $(x, y) = cov(x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1}$
  - = 0, No relation
  - = +ve, Positive cov
  - = -ve, Negative cov

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

↓

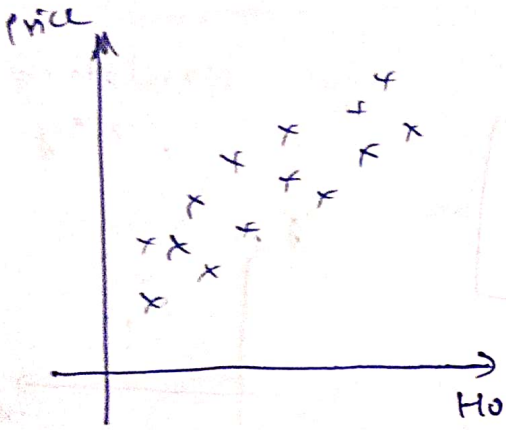
* The more toward the value of +1 then more +ve correlated
* The more toward the value of -ve then more -ve correlated b/w x & y
* The more toward the value of is zero then NO correlation.

Ex:-

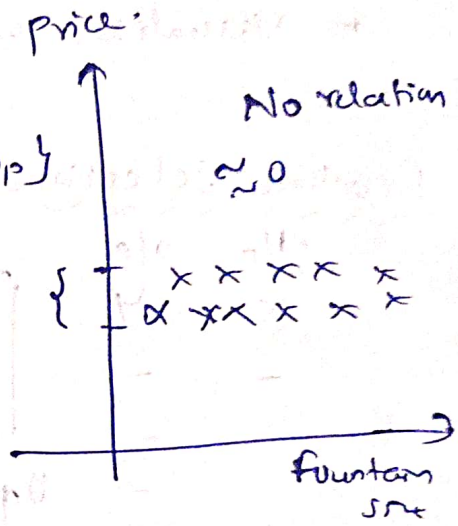
Data set Housing:- Independent feature

House size | Fountain size | → o/p feature Price

Drop this feature



← {Linear Relationship}



Feature Extraction:- independent | Dependent

Room size | No. of Rooms | Price

2 feature → 1 feature

Dimensionality Reduction

⇓ Transformation to Extract New feature

House size | Price.

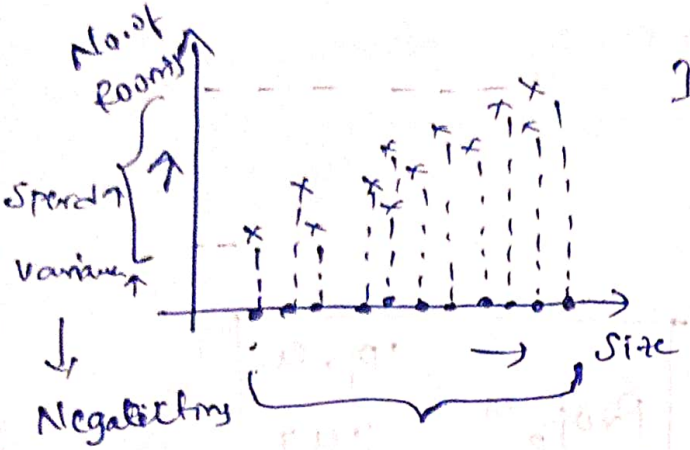
PCA Geometric Intuition:

Housing Dataset

Size House	No of Rooms	price

PCA

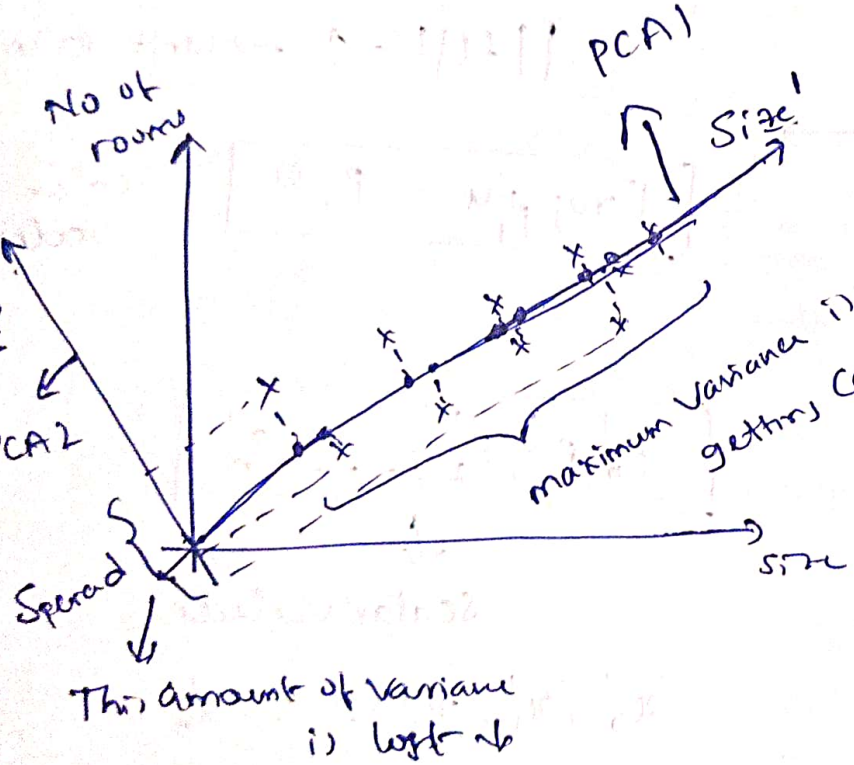
2 Dimension → 1 Dimension



In Feature Extraction

* Loss of information (No. of Rooms)

Spread ↑, → Variance ↑



Eigen Decomposition on matrix

↓

Transformation

2D → 1D

much information is not lost

* 2D axis are PC1, PC2

Var(PC1) > Var(PC2)

To get the best principal component which capture maximum Variance.

* 3D PC1, PC2, PC3

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

3D → 1D → 3D - 1D.

3D → 2D

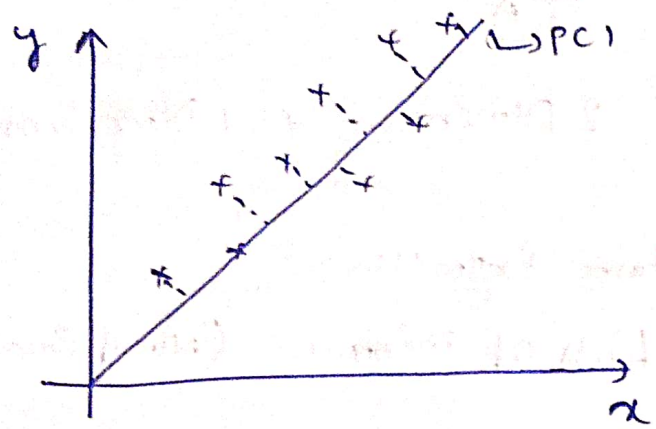
PC1, PC2, PC3 → 2D

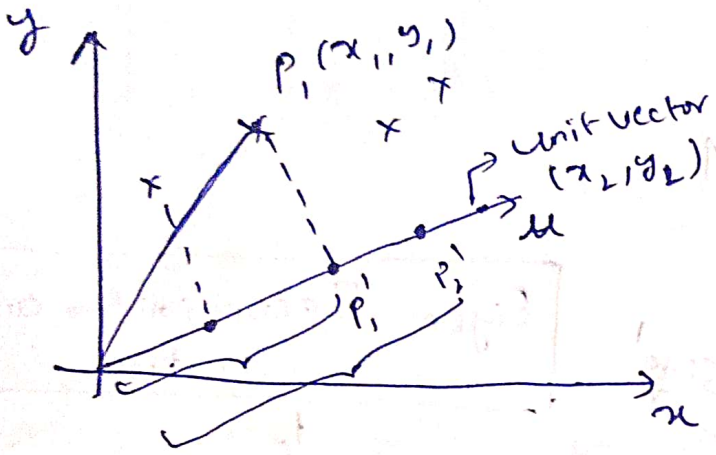
Var(PC1) > Var(PC2) > Var(PC3)

Maths intuition behind PCA Algorithm:-

- 2D → 1D

1. Projection ✓
2. cost function → Variance ✓

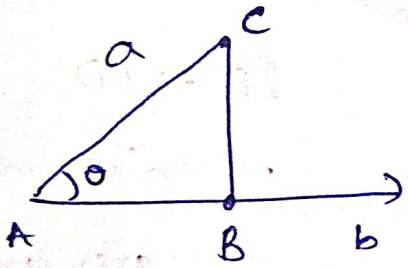




$$Proj_{P_1} u = \frac{P_1 \cdot u}{\|u\|}$$

$\|u\| = 1$ → unit vector

$$Proj_{P_1'} u = P_1 u$$ → Scalar vector



$$P_0', P_1', P_2', \ldots P_n'$$

↓

Scalar values.

$x_0', x_1', x_2', \ldots x_n'$

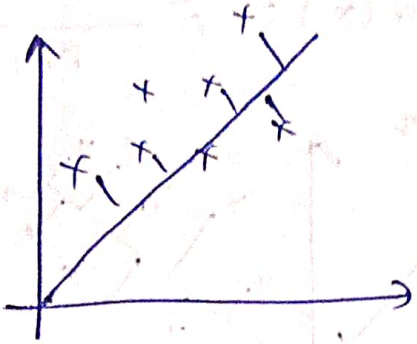
$$\therefore \text{Cost function} \rightarrow \text{Variance} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}$$

Goal – find the best unit vector which capatures the maximum variance.

Eigen Vector & Eigen Values:- $[A \cdot V = \lambda \cdot V]$

↳ Linear Transformation of matrix.

1. Covariance matrix b/w features
2. Eigen Vectors & Eigen Values will be found out from this covariance matrix.
3. Eigen Vector → larger Eigen Value → magnitud of the Eigen vector
   ↓
   Capature the maximum Valiance



[Eigen Decomposition of Covariance matrix]

⇓

Eigen vector & Eigen Values.

$$[\quad] * [v] = \lambda \cdot [v]$$

matrix.

λ → Eigen Value

$[A * V = \lambda * V]$

↓

Eigen Vector → Maximum Magnitude

↓

use as principal Compart

↓

maximum Variance

Steps to Calculate Eigen Value & Vectors:-

1. Covariance of feature

[x, y] z

↓

x'

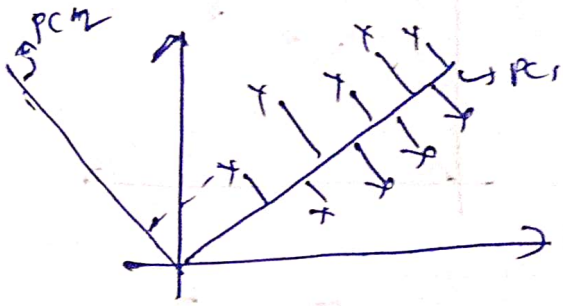
$$Cov(x,y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

2×2

	x	y
x	Var(x)	Cov(x,y)
y	Cov(y,x)	Var(y)

$$Cov(x,x) = Var(x) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{N-1}$$

	x	y	z
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)



f_1 & f_2

f_1 f_2 & f_3

$A \cdot v = \lambda \cdot v$

λ_1 λ_2

↓ ↓

PC_1 PC_2

λ_1 λ_2 λ_3

↓ ↓ ↓

PC_1 PC_2 PC_3

2D 1D

↓

3D → 2D

λ_1 λ_2 → Eigen Values

↓

PC1, PC2

3D → 1D

λ_1 λ_2 λ_3

↓ ↓ ↓

PC_1 PC_2 PC_3

Projection → 1D