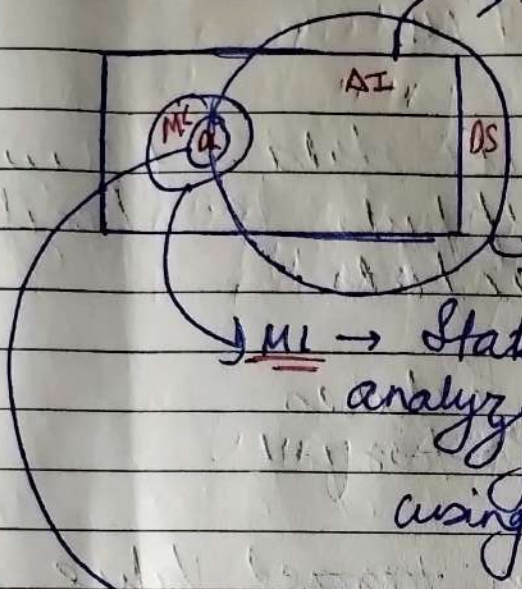
Date : Page :

{AI ML}

- Introduction
- AI VS ML VS DL VS DS



AI → An application that can do it's task without any human intervention.
Ex: (Netflix recommender system)
: (Self driving cars)

ML → Stats tool to analyze, visualize, forecast, predict using data

DL (Deep learning)
↳ trying to make a machine clean like human.
or simply to mimic human brain

Data Scientist DS : a person who analysis data then use any of the above to create a solution for a problem

Date: Page:

Machine Learning

- Supervised ML
- Unsupervised ML

In machine learning we are usually given a set of inputs called "features" and it's labeled outputs called "target values".

~~Supervised ML algo's are those which~~

We use Supervised ML algos ~~where~~ where we have access to both of these values. but, say if we only have input value & no labeled target Values, then we use Unsupervised ML algos.

Ex: → Student pass or Not { Input → Hours studied
Output → Passed / Fail

On the basis of given inputs, we can create and train a model to predict on new data if a student passe or not.

Date : Page :

~~Ex: email classification (Spam or Not)~~

~~how ? Input → Email~~

~~Spam Not spam~~

~~how we need do on relation that makes an email spam or not spam & then predict on a new email that we might find.~~

ex: recommendation made by Differen app. say Amazon:

Input : Recent purchases items in cart, wishlist

on basis of this input we need to recommend a use-s new products they would like to buy.

(Here we don't have target values as we don't know what a user might buy next).

Date : Page :

(Supervised ML)

This consist of mainly 2 type of problems :-

Regression
(output is a continuous
ex House price

Classification
(Yes/No or 0/1 or 1/2/3/
(output is
(Categorical Value)
(Choices
Ex: Student pass

• Regression : It is a statistical method w attempts to find a relationship between a dependent variable & one more dependent variable.

• Classification : It is simply the allocatio of objects to certain pre-exis classes or categories. Specifically in M it use to categorise a given set o inputs.

Algorithms generally used :-

Regression : ① Linear Regression
② Lasso & Ridge
③ Elastic Net
④ Decision trees

Date: Page

5. Random Forest
6. Xgboost
7. ANN (Artificial Neural Networks)
8. RNN (Recurrent neural Networks)
9. ~~CNN~~

Classification : 1. Logistic
2. Lasso, Ridge
3. Decision Tree Classification
4. Random Forest Classification
5. Xgboost Classification
6. ANN
7. CNN → (Convolutional Neural Networks)

~~Most~~ algos used in Regression can also be used in Classification.
(many more algos exist and are used in Supervised ML)

(Still learning ...)

Date: Page:

Unsupervised ML

↳ Clustering Algos.
- → K-Means
- → DBSCAN
- → Hierarchical

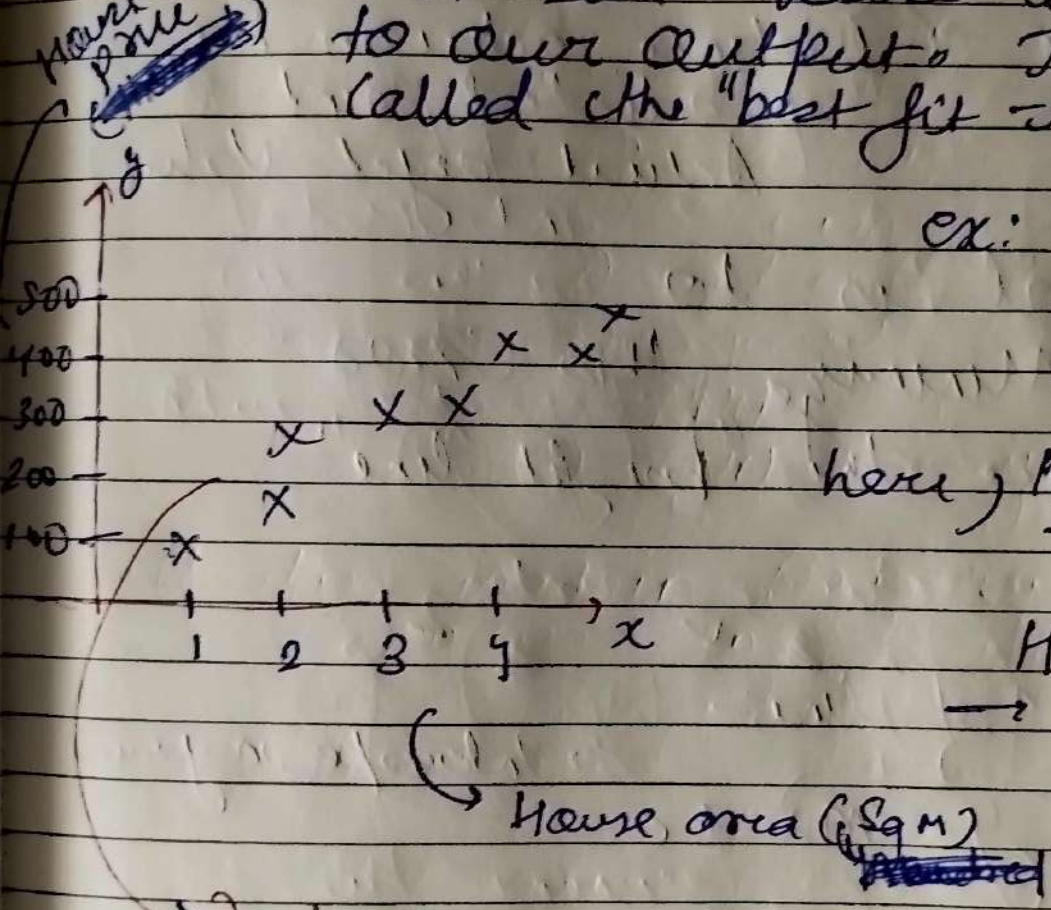
↳ Anomaly Detection
- Isolation Forest
- One Class SVM

(& many more categories & algorithms.)

(Still learning :-) :-)

Supervised ML

- Linear Regression
  - ↳ here we try finding an eqn that best relates our inputs to our output. This line is called the "best fit line".



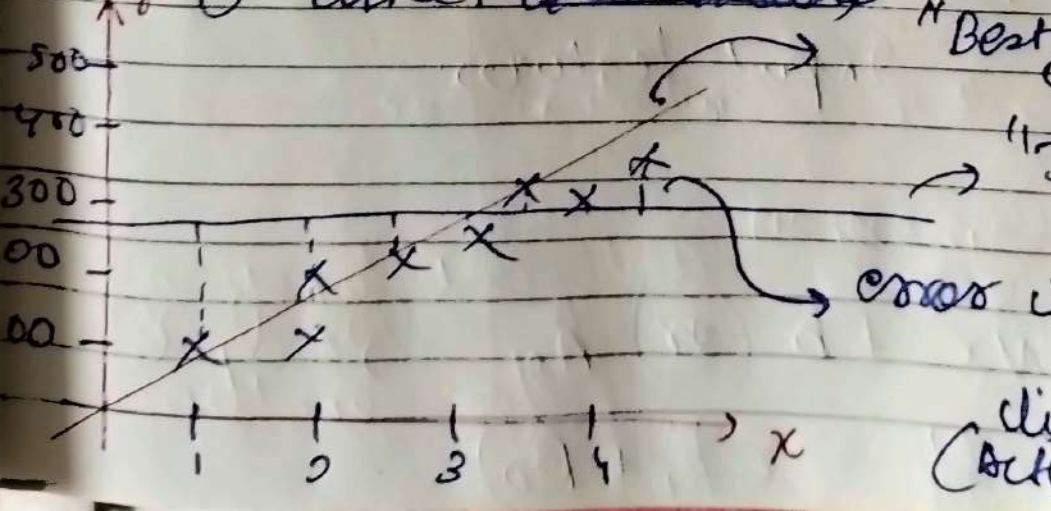
ex: House price
vs
House area.

here, House price → dependent Var.

House area → Independent

→ Data points

through this data we need to fit an ~~best~~ line:- ~~the best line~~



Date: Page:

"Just a line": Doesn't accurately predict a relation between input (x) and output (y). So, it's prone to have a ton of error.

error → (Distance between the predicted point & Actual target value)

Our goal is to "Minimize this error" to produce the "best fit line"

General eq^n of line:-

$y = mx + c$ → Intercept (or y axis)

→ slope

→ prediction

→ Input features

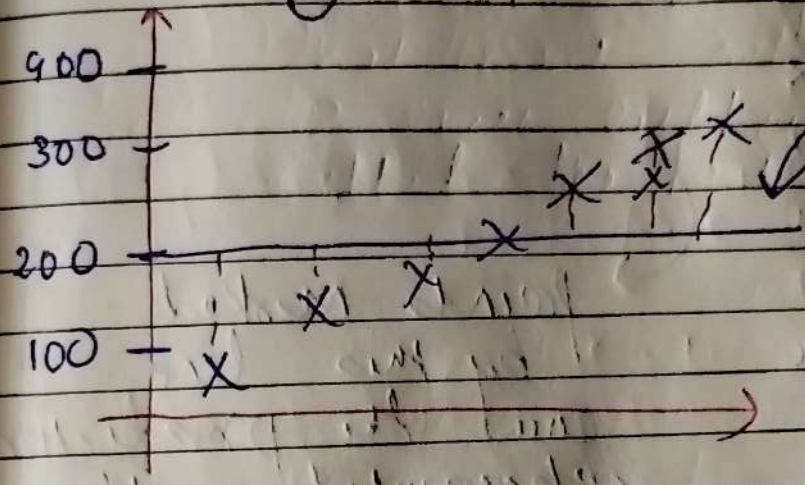
In linear Regression:-

$y_{pred} = wx + b$ (or) $h_\theta = \theta_0 + \theta_1 x$

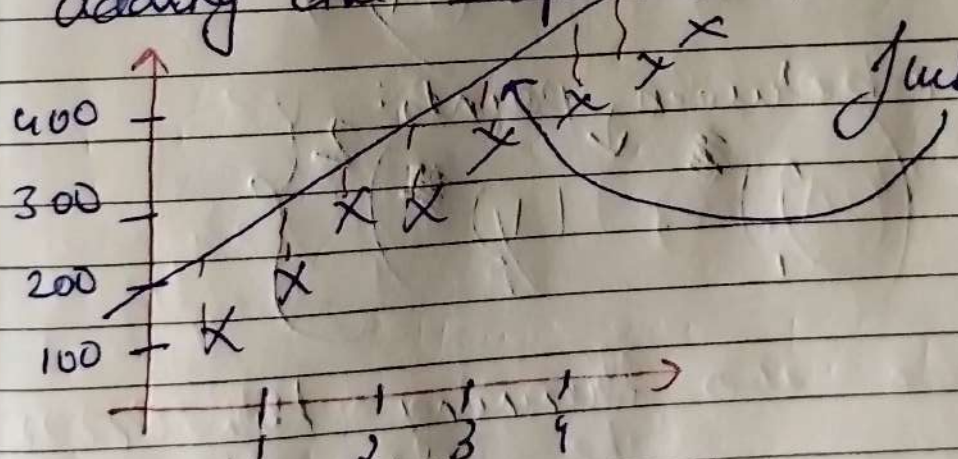
(or) $(y_i = x\beta_1 + \beta_0)$

9 use : $(f_{wb} = Wx + b)$

So, say $b = 200$, $W = 0$ $\{f_{wb} = 200\}$



adding the slope term :- $b = 200$, $W = 5$



So, to minimize this error we need to find correct values of W & b.

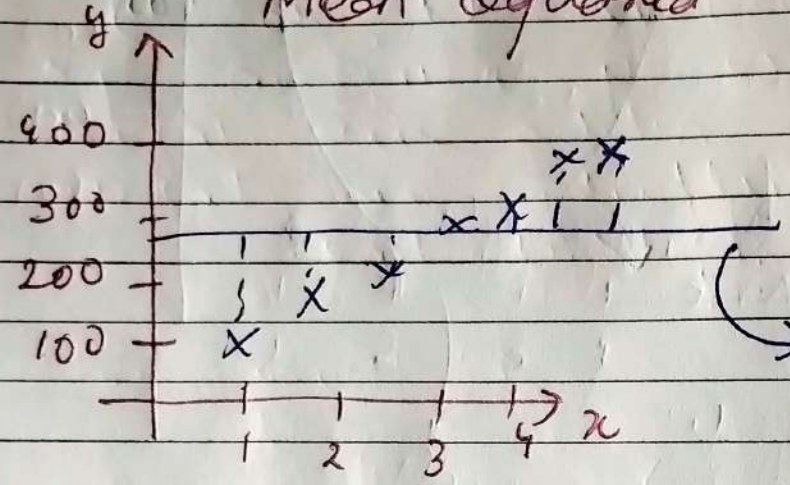
• More general eqn :-

$$\left(f_{Wb} = b + W_1 x_1 + W_2 x_2 + \cdots + W_n x_n \right)$$

Date : Page :

Computing the error :

Mean Squared Error :



points marked by this line and the prediction represented with $(\hat{y})$

Distance between points :

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

- $\frac{1}{2m}$ → here for calculation ease
- $\frac{1}{m}$ → taking mean
- $\sum_{i=1}^{m}$ → summing this error over all the data points

The above eqn is called "Cost function" or "Loss function"

Date: Page:

Represented as :-

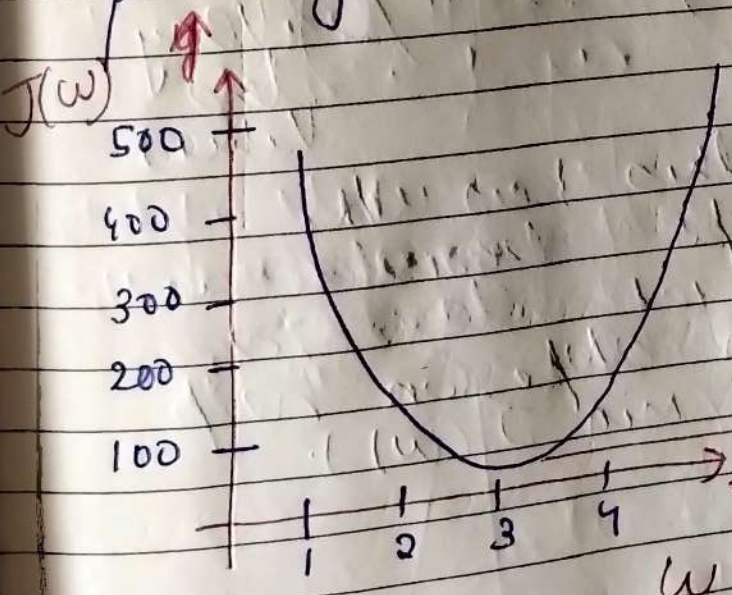
$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \left((\hat{y}_i) - (y_i) \right)^2$$

→ we need to minimize this cost.

Optimization Algorithm:

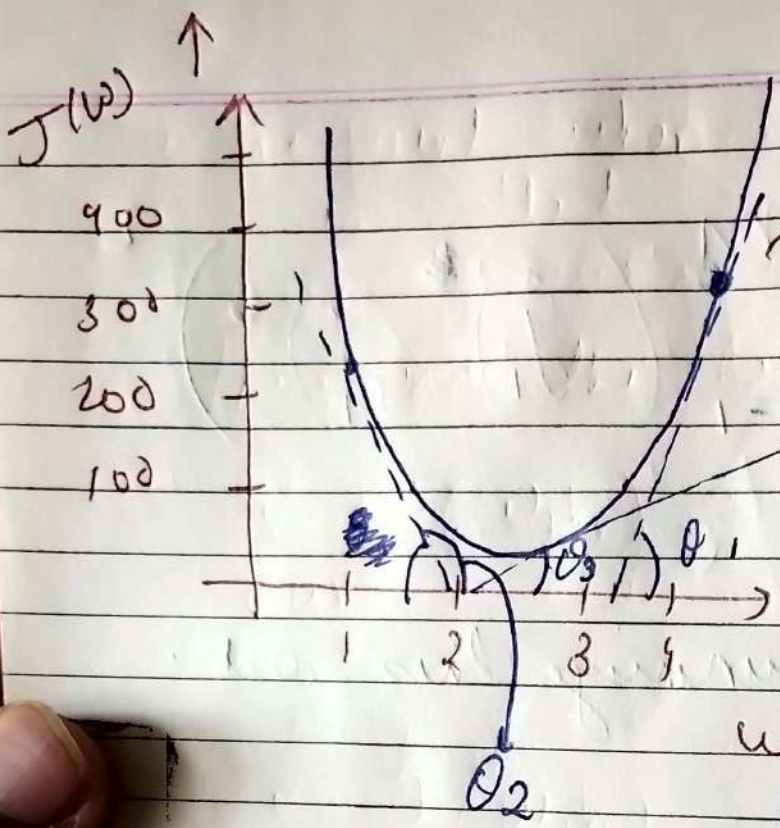
① "Gradient Descent Algo"

Plotting this $J(w,b)$ for different w values & $b=0$.



→ parabolic curve with a global minimum

→ at this global minimum we get the least value $J(w)$. we need to come how reach this point.



→ taking a random point & calculating it's slope

$\theta_0, \theta_2 \rightarrow \theta_1 \rightarrow \theta_3$

(slope)

now, $\tan(\theta_1) \rightarrow$ +ve

$\tan(\theta_2) \rightarrow$ -ve

$\tan(\theta_3) \rightarrow (+/-)$ (very less value)

We can use this, as the ~~slope decreases we get closer to~~ variation in slope can help get to ($\min J(w)$).

We will simply :—

$$W_{new} = \left(W - \frac{2}{2W} J(W)\right)$$

to min J(w),

if slope is +ve → W↓ → getting closer to min J(w).

(if we are already near min J(w), ~~we make the update~~ the value of derivative will be small, so no significant ↑ or ↓.)

going from right to left → W↓

we encounter min J(w)

similarly left to right → W↑

to Quantify how much to move we use the ∂ value of slope.

Repeated updates to the parameters result in optimal W value

This algorithm is "gradient descent algo."

representation :-

$$W_{new} = W_{old} - \alpha \frac{\partial}{\partial w} J(w,b)$$

"learning rate"

$$b_{new} = b_{old} - \alpha \frac{\partial}{\partial b} J(w,b)$$

The point when it reaches min J(w,b) is called "Convergence" & we repeate our algo till then.

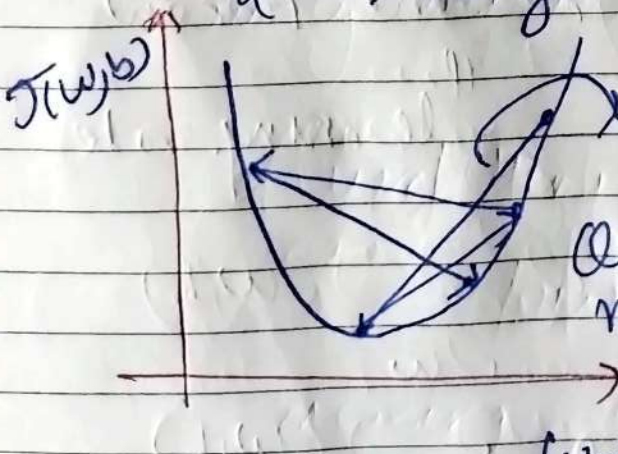
these are the derivative terms:-

$$\frac{\partial}{\partial w} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} ((\hat{y}_i) - y_i) \cdot x_i$$

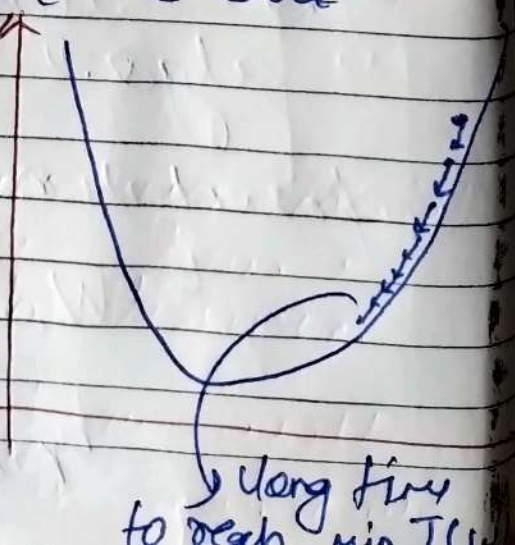
$$\frac{\partial}{\partial b} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} ((\hat{y}_i) - y_i)$$

Learning Rate (α) : It decides how big of a step we take while updating of parameters (w,b)

α >> large



α <<< small



we need the pick alpha Dane where in the middle of this.

Matrix Notation of Linear Regression

$$y = XW + \varepsilon$$ (ε → residuals)

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ & \vdots & & \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}$$

(for bias/intercept term) (no of features in each example)

↳ total no of examples

$$W = \begin{bmatrix} b \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_p \end{bmatrix}$$

b → intercept term

$w_1, w_2, w_3, \ldots, w_p$ → coefficients / slopes

another way to find the parameters :

$$\vec{W} = (X^T X)^{-1} X^T Y$$

{Still learning as to How this works ?}.

→

Problem of Overfitting and Under-fitting.

- Bias : error in the prediction made by model.
- Variance : Specifies the amount our model when a different portion of data (or new unseen data) is used.

We, want our model to have low-Bias, & low Variance (though both simultaneously are not possible)

Overfitting Occurs when, our model has ~~high Bias & low~~ low bias & high Variance.

meaning our model performs well on training data but fails on unseen new data

Date: Page:

It occurs when our model is trained only on training data & the optimization algo end up with over accurate parameters that work only to model real training data.

There are multiple way to fix this ex: (Splitting data to set (into train, test, Cross Valid)

(Using regularization methods)

Underfitting: Occurs when our model has high Bias & low Variance
i.e. it doesn't perform well on training data & there is no change in its performance on Switching data.

Causes are poor training of model, wrong K etc.

(Ridge & Lasso Regression)

Solution to the problem of overfitting.
This is a regularization method.

Overfitting occurs when low bias & high Variance.
So, here we intentionally add some error to our model which results in more error on training data but also reduces Variance.

- Ridge Regression. (also called L2 Norm)
we add this term to our (MSE) or Cost function.

$$\left\{ + \frac{1}{2m} \left(\sum_{i=1}^{n} (W_i)^2 \right) \right\}$$

(n = no of features) ← → Regularization parameter.

& it's derivative wrt to W becomes

$$\left\{ + \frac{1}{m} \left(\sum_{i=1}^{m} W_i \right) \right\}$$

Date: Page:

- In addition to adding an bit of error, to more generalize the model.

Lasso Regression also helps in Reducing the dimensionality of our data.

i.e

it eliminates irrelevant features.

We add this term in Lasso: (also called L1 norm)

$$+ \frac{1}{p} \lambda \sum_{i=1}^{p} |w_i|$$

and it's Derivative cost w.

(now as mod doesn't have a proper Derivative we use *Subderivative*.

$$+ \lambda \sum_{j=1}^{p} \text{sign}(w_j)$$

- $\text{sign}(w_j) = \begin{cases} 1, & \text{if } w_j > 0 \\ -1, & \text{if } w_j < 0 \\ 0, & \text{if } w_j = 0 \end{cases}$

Date : Page :

Difference between the 2 is
→ Ridge introduce more errors,
→ Lasso introduce less error than Ridge but helps in Reducing dimension.

Now, The Combination of the above 2 becomes :-

(Elastic net regularization
↳ having good point of both)

this is the term added to cost function :-

$$+ \frac{1}{2p} \lambda_1 \sum_{j=1}^{p} (W_j)^2 + \frac{1}{p} \lambda_2 \sum_{i=1}^{p} |W_i|$$

and derivative cost to w :-

$$+ \frac{1}{p} \lambda_1 \sum_{j=1}^{p} W_j + \frac{1}{p} \lambda_2 \sum_{i=1}^{p} \text{sign}(W_i)$$

The above eq is also used in this form:-

$$\frac{1-\alpha}{2p}\sum_{j=1}^{p}(w_j)^2 + \frac{\alpha}{p}\sum_{j=1}^{p}|w_j|$$

→ where α is "mixing parameter".

(if $\alpha = 1$, Elastic Net performs Lasso

$\alpha = 0$, Elastic Net performs Ridge

$0 < \alpha < 1$, (combination of both))

Date: Page:

- In Ridge Regression, the penalty reduces the values of large coefficient but it never goes to zero.
- but, in Lasso this goes to zero resulting in zero parameters reducing to 0

(I still am not entirely clear as to why Lasso goes to 0 ...)

→ Linear Regression itself also has many variants. —

(1) Simple Linear Regression → (with 1 feature)

(2) Multivariable Linear Regression
↳ (with multiple feature)

(3) Multivariate
↳ predicting more than one variable on the basis of multiple input features

(ex: (predicting if student passes in English & Hindi, simultaneously)

Date: Page:

④ Polynomial Regression: (fitting a data with higher degree relation)
ex: $(y = x^2)$

(many more still learning - 1L)

→

Date: Page:

{Logistic Regression}

↳ Here we try classifying either a continious set of inputs or a set of binary inputs into one of 2 category usually 1/0 or (True, False)

We used the normal line's eqn to find the value of a prediction but as we have to classify this input in terms of 0 & 1 we use

Sigmoid function
(it a case of logistic function)

(logistic growth)

$$p(x) = \frac{L}{1 + e^{-(x - x_0) \cdot k}}$$

L → Supremum of the values of function.

x_0 → location parameter which means mid-point of curve where $(p(x_0) = 1/2)$

k → logistic growth rate

Date: Page:

when the value of

$L=1, k=1, x_0=0$

$$f(x) = \frac{1}{1+e^{-x}}$$

↳ Sigmoid function

In our Code:

$$f(w,b) = \frac{1}{1+e^{-(wx+b)}}$$

$(z = wx+b)$

↳ this function squashes value between (0 & $L=1$).

Date: Page:

a more accurate way of writing sigmoid function, as we won't always have $(\mu = 0)$

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

→ Scale parameter
↓
Controls the spread of the distribution

↓
(midpoint of curve, where $p(\mu) = 1/2$)

and when written with $f(w,b)$:-

$$f(w,b) = p(x) = \frac{1}{1 + e^{-(b + wx)}}$$

$$\begin{cases} b = -\mu/s \\ \text{Intercept} \end{cases}, \quad \underbrace{w = 1/s}_{\text{Slope}}$$

$$\left(\mu = -b/w, \; s = 1/w\right)$$

while, implementing we mostly care about.

$$f(w,b) = \frac{1}{1+e^{-(wx+b)}}$$

now, here also the initial sigmoid might not "fit" our data well. So we have to reduce the error here as well.

But, Linear Regression's Cost function won't work here. we need to find something else:-

we define.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}\left(f_{(w,b)}, y^{(i)}\right)$$

new one

$$\begin{cases} -\log(f(w,b)) & \text{if } y=1 \\ -\log(1-f(w,b)) & \text{if } y=0 \end{cases}$$

Combining we get this:

$$\Rightarrow -y\log(f(w,b)) - (1-y)\log(1-f(w,b))$$

while implementing we mostly care about.

- $f(w,b) = \frac{1}{1+e^{-(wx+b)}}$

now, here also the initial sigmoid might not "fit" our data well. So we have to reduce the error here as well.

But, Linear Regression's Cost function won't work here. We need to find something else:-

we define.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}\left(f(w,b), y^{(i)}\right)$$

(new one)

$$\begin{cases} -\log(f(w,b)) & \text{if } y=1 \\ -\log(1-f(w,b)) & \text{if } y=0 \end{cases}$$

Combining we get this:

$$\Rightarrow -y\log(f(w,b)) - (1-y)\log(1-f(w,b))$$

also known as binary cross entropy))

finally:-

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\left(-y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)\right)$$

reduces to

$-\log(\hat{y}_i)$ for $y_i = 1$

& to

$-\log(1-\hat{y}_i)$ for $y_i = 0$

and it's total sum is also called (-ve log likelihood)

Now why this & how does the penalizes the input if it has a large error.

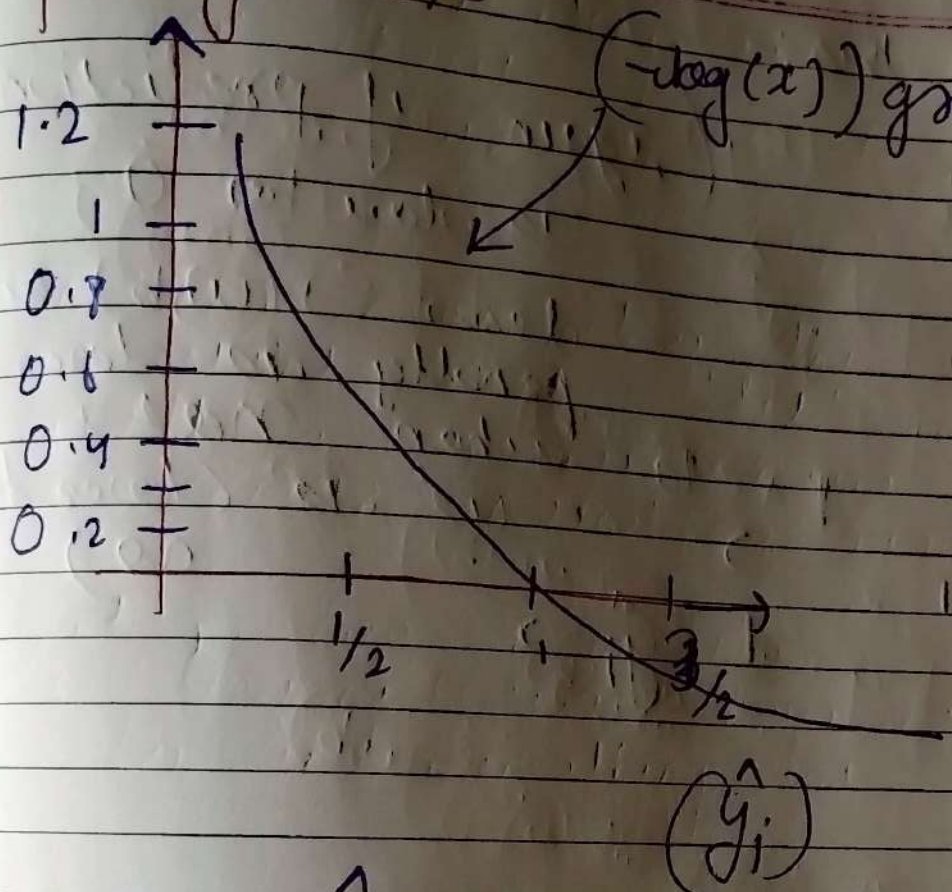
The function is in 2 parts for 4 reason:-

when ($y_i = 1$), we want our value to be as close to true value as possible.

So this function works:-

$-\log(\hat{y}_i)$

plotting this:-



when, $\hat{y}_i$ is close to 1, $J(w,b) \to 0$

& $\hat{y}_i$ is close to 0, $J(w,b) \to \infty$

heavily penalizing the wrong predictions.

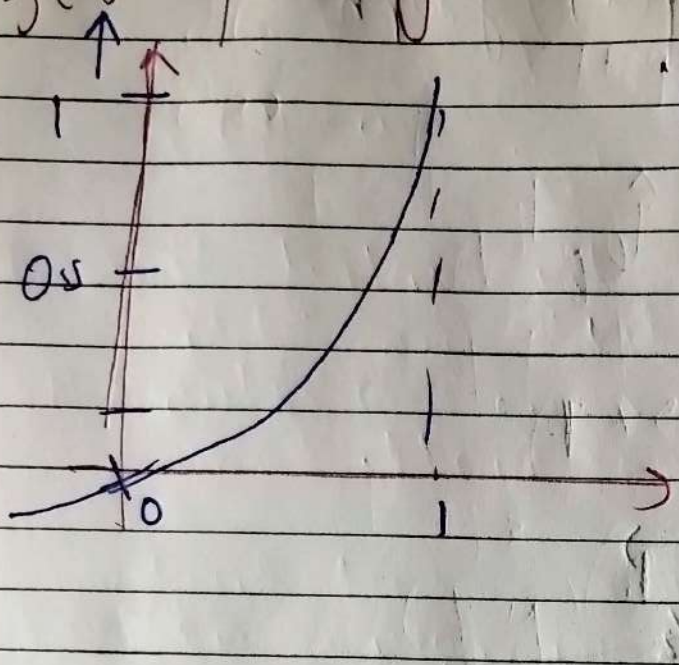
Similarly when $(y_i = 0)$,

this function works

$$-\log(1-\hat{y}_i)$$

Date : Page :

$J(w,b)$: plotting this :-



(again if prediction is close to 0, then loss $J(w,b)$ or penalty is closer to 0 & when not, it is closer to 1)

Gradient Descent remain same as linear Regression.

as when we take derivate of above Cost function we get the same value as linear regression, only now definition of $f(w,b)$ is different

• Let's find the derivative :-

$$A = -y * \log(\hat{y})$$

$$= -y * \log\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$\frac{\partial A}{\partial w} = -y * \frac{1}{1+e^{-(wx+b)}} \underbrace{\frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)}$$

$$= -y * \left(1+e^{-(wx+b)}\right) + \frac{1}{1+e^{-(wx+b)}}\left(1 - \frac{1}{1+e^{-(wx+b)}}\right)$$

$$\frac{-1 \cdot e^{-(wx+b)} \cdot x}{\left(1+e^{-(wx+b)}\right)^2}$$

$$= -y \cdot \frac{e^{-(wx+b)}}{\left(1+e^{-(wx+b)}\right)} \cdot x$$

$$= -y\left(\frac{1+e^{-(wx+b)} - 1}{1+e^{-(wx+b)}}\right) x$$

$$= -yx\left(1 - f(wx+b)\right)$$

or

$$\hat{y}$$

$$\frac{\partial B}{\partial w} = -(1-y) \cdot \log(1-\hat{y})$$

$$\frac{\partial B}{\partial w} = \frac{+(1-y)}{1-\hat{y}} \cdot +\frac{\partial}{\partial w} \frac{1}{1+e^{-(wx+b)}}$$

$$= \frac{(1-y)}{1-\frac{1}{(1+e^{-(wx+b)})}} \cdot \frac{e^{-(wx+b)} \cdot x}{(1+e^{-(wx+b)})^2}$$

$$= (1-y) \cdot \frac{(1+e^{(wx+b)})}{e^{-(wx+b)}} \cdot \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})^2} \cdot x$$

$$= (1-y) \cdot x \cdot f(wx+b)$$

now for $\frac{\partial (\log L)}{\partial w} = \frac{\partial A}{\partial w} + \frac{\partial B}{\partial w}$

$$= -yx(1 - f(wx+b)) + (1-y) \cdot x \cdot f(wx+b)$$

$$= -yx + yx f(w,b) + x f(w,b)$$

$$- yx - f(w,b)$$

$$= (f(w,b) - y)\, x$$

$$= (\hat{y} - y) \cdot x$$

Equal to the term in Linear regression.

Similarly we can find b.

$$\frac{\partial A}{\partial b} = -y \cdot \log(\hat{y})$$

$$= -y \cdot \frac{1}{\hat{y}} \cdot \frac{\partial}{\partial b} \hat{y}$$

$$= -y \cdot \frac{1}{\hat{y}} \cdot \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})^2}$$

$$\frac{\partial A}{\partial b} = -y \cdot \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})^2}$$

Date : Page :

$$\frac{\partial B}{\partial b} = -(1-y)\log(1-\hat{y})$$

$$= +\frac{(1-y)}{1-\hat{y}} \cdot \frac{\partial}{\partial b}(\hat{y})$$

$$= \frac{(1-y)\,(1+e^{-(wx+b)})}{e^{-(wx+b)}} \cdot \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})^2}$$

$$\frac{\partial Cost}{\partial b} = \frac{\partial A}{\partial b} + \frac{\partial B}{\partial b}$$

$$= -y\,\frac{e^{-(wx+b)}}{1+e^{-(wx+b)}} + \frac{(1-y)}{(1+e^{-(wx+b)})}$$

$$= \frac{1-y(1+e^{-(wx+b)})}{(1+e^{-(wx+b)})}$$

$$= (f(w,b) - y)$$

or $(\hat{y} - y)$

Date: Page:

another way of minimizing loss is to maximize the inverse of (-ve log likelihood)

meaning (+ve log likelihood)

Still learning this—

Finally to convert the final squashed value to a probability 0 or 1 we use a "Decision Boundary"

or a threshold line, above which model predicts 1, below which 0. usually of 0.5

usually we take ($y = 0.5$)

Date : Page :

Naive Bayes Classifier

"Based on Bayes' Theorem:

It describes the probability of an event, based on prior knowledge of condition that might be related to the event.

→ "likelihood"

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

(PRIOR)

→ Conditional probability → probability of A happening if B happened.

aka posterior probability.

+ (Similarly for $P(B/A)$)

$P(A)$ & $P(B)$ → probability of A & B respectively without any condition.

For Machine learning :-

Say we have to predict (y) given that $(x_1, x_2 \cdots x_n)$ happended.

So, $P\left(\frac{y}{x_1, \cdots x_n}\right) = \frac{P(x_1 \cdots x_j / y)\, P(y)}{P(x_1 \cdots x_j)}$

to understand How we use this, let's take the example of :—

"whether to play or not, on the basis of "Outlook", "Humidity", "Temp", "Windy".

Date : Page :

Outlook	Temp	Humidity	Windy	play
Rainy	H	Hi	F	no
R	H	Hi	T	no
OC	H	Hi	F	yes
S	M	Hi	F	yes
S	C	N	F	yes
S	C	N	T	no
OC	C	N	T	yes
R	M	Hi	F	no
R	C	N	F	yes
S	M	N	F	yes
R	M	N	T	yes
OC	M	Hi	T	yes
OC	H	N	F	yes
S	M	Hi	T	no

R → Rainy, OC → Ques Cast, S → Sunny

H → hot, M → Mild, C → Cool

Hj → High, N → Noorml.

now we can further simplify the eqn.

$$P(y/x_1, x_2, x_3 \cdots x_n) \Rightarrow$$

$$P(y) * \frac{P(x_1 x_2 x_3 x_4 \cdots x_n / y)}{P(x_1, x_2, x_3, x_4 \cdots x_n)}$$

$$\Rightarrow P(y) * \frac{P(x_1/y) * P(x_2/y) * \cdots P(x_n/y)}{P(x_1) P(x_2) \cdots P(x_n)}$$

Say $y \rightarrow$ (T or F)

then we can write :-

$$P(T/x_1, x_2, \cdots x_n) \Rightarrow P(T) * \frac{P(x_1/T) \cdot P(x_2/T) \cdots P(x_n/T)}{P(x_1) P(x_2) \cdots P(x_n)}$$

Similarly for $P(N/x_1, x_2, \cdots x_n)$

(* We can ignore the denominator as it is constant for all)

So, now we compare these values:

(PLAY)		P(Yes)	P(NO)
Yes →	9	9/14	5/14
NO →	5		

(Outlook)

	Yes	NO	P(Yes)	P(NO)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	3/5
	9	5		

→ P(Sunny / Yes) or P(Sunny / NO)

(Temperature)

	Yes	NO	P(Yes)	P(NO)
Hot	2	2	2/9	2/5
mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
	9	5		

(Humidity)

	(Yes)	(No)	P(Yes)	P(No)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
	9	5		

(Wind)

	Yes	No	P(yes)	P(No)
Weak (F)	6	2	6/9	2/5
Strong (T)	3	3	3/9	3/5
	9	5		

(OH) $P(\text{Weak}/\text{Yes}) * P(\text{Strong}/\text{Yes})$

Now, Say we need to predict if to play on a day with:-

$X = (\text{Sunny}, \text{Hot}, \text{High}, \text{Strong})$

$P(\text{Yes}/X) = 9/14 * 3/9 * 2/9 * 3/9 * 3/9$

$\Rightarrow 0.00529$

$P(\text{No}/X) = 5/14 * 3/5 * 2/5 * 4/5 * 3/5$

0.02742

$\Rightarrow$

Date: Page:

We can't really understand anything from this value so,

$$P(Yes/\overline{X}) = \frac{0.00589}{0.00589 + 0.02744}$$

$$= 0.1729 \times 100$$

$$= 17.29\ \%$$

$$P(No/\overline{X}) = \frac{0.02744}{0.02744 + 0.00589} \times 100$$

$$= 82.71\ \%$$

$$= 83.82\ \%$$

So, our prediction will be (No)

Evaluating an Classification Model.

(Other than logistic loss)

- Confusion Matrix.

Say, I have some data on which I have trained a model and made some prediction.

$y \rightarrow$ 0 | 0 | 1 | 0 | 1

$\hat{y} \rightarrow$ 0 | 0 | 0 | 0 | 1

(Actual) \ (Predicted)	NO(0)	YES(1)
NO(0)	2	1
YES(1)	1	4

Confusion Matrix divides our model's prediction into 4 categories, which helps us understand how confused our model is.

- True Positive (TP): predicted → 1
  Actual → 1

- False Positive (FP): prediction → 1
  Actual → 0

- False Negative (FN): prediction → 0
  Actual → 1

- True Negative (TN): prediction → 0
  Actual → 0

Note: Confusion Matrix will change with change in the threshold value.

Now, using this Confusion Matrix we can determine one of these measures of evaluation for our model:-

* Accuracy → $\frac{TP + TN}{TP + TN + FP + FN}$

→ It is simply how accurate our model is predicting (work good for balanced dataset).

Raj
Page:
Date:

In Case of Imbalanced dataset :-

Say we have 900 True
100 False.

if our model predicts True every time it will have a 90% Accuracy, but we know the model is performing poorly.

For, Imbalanced dataset we need to try something else.

First, we need to understand what type of prediction our model needs to make. i.e.

Say, Diagnosing a Cancer Patient:

In this case a False Negative is very dangerous as it means the person has Cancer but we predicted not, than making a False positive.

So, we need our model to predict True Values much more and as accurately as possible.

or

Say, Giving Loan:

Denying a good customer loan (false positive) is not as bad as giving it to a bad customer. (false negative).

Here, precision and Recall comes in.

Precision:

$$\Rightarrow \frac{TP}{TP + FP}$$

Here we ask the question:

("Out of all the prediction, how accurately have it predicted positive class?")

Recall:

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

Precision measure the accuracy of the positive prediction made by the model.

* High Precision means our model is likely to predict and actual positive as 'positive'.

Ex: 10 True predicted → 7 actual True → 70% precision
3 actual False

Recall $\Rightarrow \frac{TP}{TP+FN}$

Here we ask the question:

"Out of all positive Cases in Sample, how many of those are Correctly predicted by our model?"

Recall measures the ability of model to detect positive Sample.

High Recall model means our model accurately predicting the positive Samples.

~~Ex: 10 prediction → 7 TP~~
~~3 FN (actual Positive)~~

Raj
Page:
Date:

ex 10 Prediction made by our model.

5 True Positive

5 False Negative (meaning actual value is Positive)

So, our model has 50% Recall.
meaning it is predicting 50% of time an Positive Value as actually positive.

where in precision, out of all the "True" prediction made, we were finding How many of them are actually True).

(Trade off) → when we ↑ threshold we make stricter prediction about what is to be considered as True or False.
ie (TP ↑, FN ↑)
(FP ↓)
ie (precision ↑, Recall ↓)

{F1-Score}

~~This is a bit of middle ground between Recall & Precision~~

This creates a bit of middle ground between a high Recall & high precision model.

$$\Rightarrow 2 * \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

("Harmonic Mean")

F1 itself is actually a special case of F_β Score.

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

for, $\beta > 1 \rightarrow$ recall ↑

$\beta = 1 \rightarrow (F1)$

$\beta < 1 \rightarrow$ Precision ↑

ROC & AUC → (Area under ROC)

↳ (Receiver Operating Characteristic)

is created by plotting (TPR)
True Positive Rate

& false positive Rate (FPR)

⭐ → TPR, Recall, Sensitivity = $\frac{TP}{TP+NN}$

⭐ Specificity = $\frac{TN}{TN+FP}$

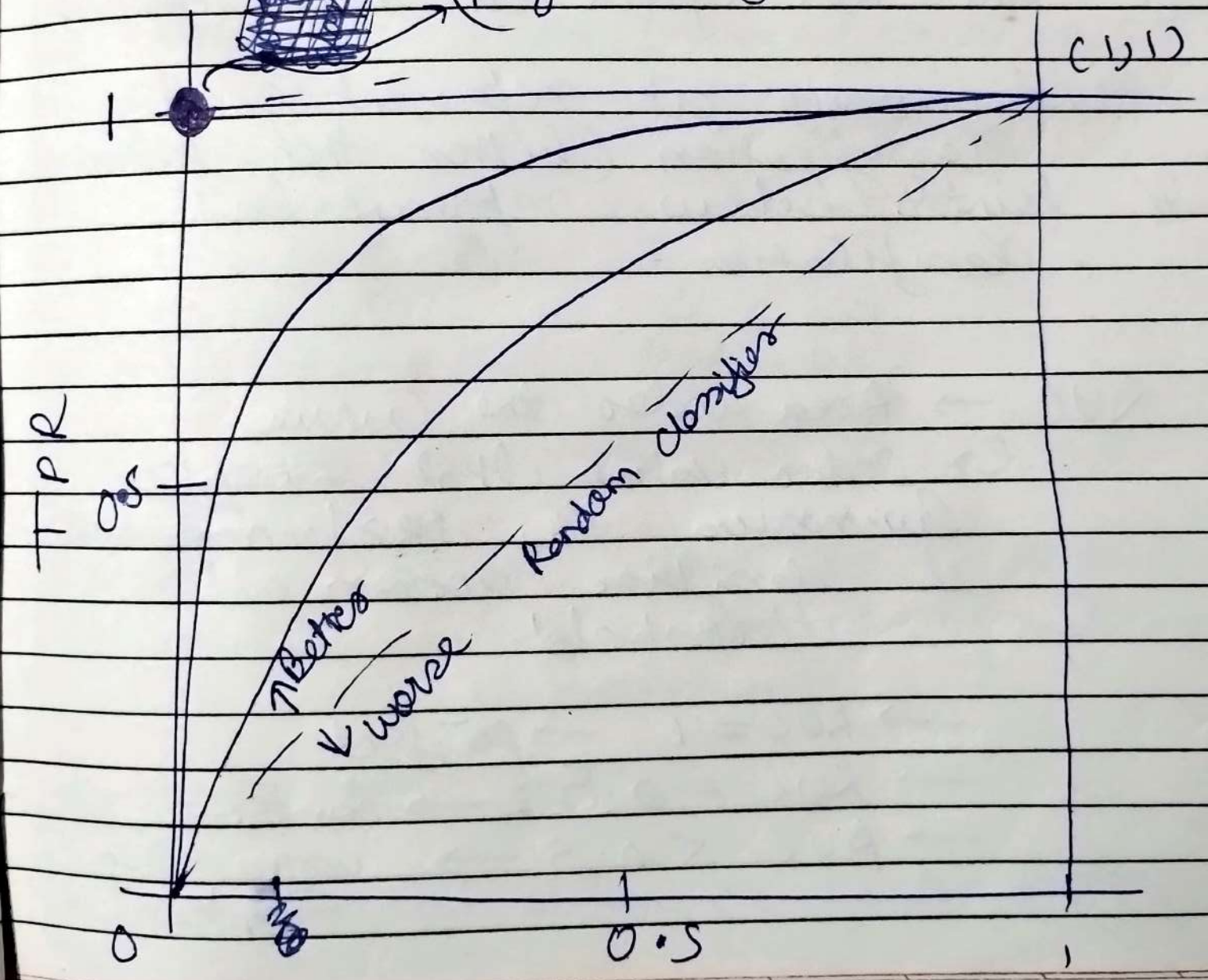
⭐ → FPR, (1 − Specificity) = $\frac{FP}{FP+TN}$

ROC → helps evaluating performance of classification model by plotting TRR Vs FPR at various threshold settings.

First we Calculate TPR & FPR for different threshold (0 ... 0.9).

then, we plot the TPR & FPR of (Y) (X) a corresponding threshold value.

ROC Space → (Perfect classifier)



In the graph best possible prediction method would be point the point "perfect classifier"

Representing
100 % sensitivity (NO FN)
100 % specificity (NO FP)

The diagonal (Random classifier), divide the ROC space in two regions.

Points above it represent good classification (better than Random)
& Points below represent bad classifications.

AUC → Area under the curve
↳ Scalar value that ~~classifies~~ summarizes the performance of classifier across all threshold.

→ AUC = 1 → perfect
→ AUC = 0.5 → Random
→ AUC < 0.5 → worse than Random

{ Phi Coefficient or }

Mathews Correlation Coefficient

- ~~(geometric mean of regression coff of problem and its dual)~~.

- $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

→ Regarded as the balanced measure as it takes into count true & false positive and negatives. and can be used even if the classes are of very different size.

It Returns Value between :—

−1	0	+1
~~perfect~~ Total Disagreement	no better than random prediction	perfect prediction

~~for comparison~~

{ • The Closer the value to 1, the better. }

Cohen's Kappa

Probability of observed agreement

$$K = \frac{P_o - P_e}{1 - P_e}$$

P_e → Probability of agreement by chance.

(Binary classification)

$$\frac{2 \times (TP \times TN - FN \times FP)}{(TP + FP) \times (FP + TN) + (TP + FN) \times (FN + TN)}$$

It is used to measure "inter-rater reliability" for a categorical item.

* inter-rater reliability is the degree of agreement among independent observers who rate, code, or asses the same phenomenon.
  → (In our case agreement between our model & Actual Values)

for example,

Say we hire 2 Judges to segregate 3 colored balls in different bags. Say Red, green, blue.

If both Judges Judge without bias, they would agree on color of all balls i.e. they will have zero disagreement.

→ but it's not possible, so we want to measure the agreement between them.

→ {Some they will agree actually}
→ Some they will agree by chance

by chance for our context means, if we just randomly guessing we still might get an agreement.

We don't want our ML model doing this.

- We need to measure only the actual agreement.

$P_{observed}$ → Probability of observed agreement.

$P_{bychance}$ → Probability of ~~Observ~~ agreement by chance.

Example this is the Confusion Matrix :-

(II)

	R	G	B	Total
R	44	5	1	50
G	7	20	3	30
B	9	5	6	20
Total	60	30	10	100

(Observed)

The Diagonal Values are the Observed Values: ~~Total~~

(1) $P_{Ob} = \frac{44 + 20 + 6}{100} = 0.70$

Or

$.44 + .2 + .6 = 0.70$

now, for probability by chance,

out of 100 ball how many did Judge 1 classify as red → 60

$P_{J_1R} = 60/100 = 0.6$

Similarly, $P_{J_2R} = 50/100 = 0.5$

→ both classifying Red

$\rightarrow P_{J_1J_2R} = 0.6 \times 0.5 = (0.3)$

(P(by chance for Red))

Similarly, ~~$P_{J_1J_2R} = 0$~~

$J_1 \rightarrow$ Red, $J_2 \rightarrow$ green: $0.6 \times 0.3 = 0.18$

$J_1 \rightarrow$ Red, $J_2 \rightarrow$ Blue: $0.6 \times 0.2 = 0.12$

Similarly we Construct a new Confusion Matrix:

J2 \ J1	Red	green	blue	Total
Red	0.30	0.15	0.05	0.5
green	0.18	0.09	0.03	0.3
blue	0.12	0.06	0.02	0.2
Total	0.6	0.6	0.1	1

('by Chance)

Now $P_{byChance}$ is the diagonal values

$$= 0.30 + 0.09 + 0.02 = 0.41$$

So, $K = \frac{P_{obs} - P_{byChance}}{1 - P_{byChance}} = \frac{0.70 - 0.41}{1 - 0.41}$

$$= 0.4915$$

A Value of $K > 0.75$ is considered good.

(The one we just did is the Un-weighted)

→

Weighted Kappa :-

Currently our data had distinct value but say if we had ("Shades of red") → (ordinal categories)

In this case we calculate "weight". Say we have 5 shades of Red and we need to assign value from 1 to 0

In order to do this we define a ("Distance" Variable.

(linear) ↓

	R_1	Dist	R_2 → $1 - \frac{dist}{maxdist}$
R_1	1	0	
R_2		1	→ 0.75
R_3		2	
R_4		3	(Quadratic)
R_5	0	4	$R_2 \rightarrow 1 - \left(\frac{dist}{maxdist}\right)$

→ 0.94

(linear → weight are linearly spread between 0 to 1)

(Linearly)

	R_1	dist
R_1	1	0
R_2	0.75	1
R_3	0.50	2
R_4	0.25	3
R_5	0	4

(Quadratic)

	R_1	dist
R_1	1	0
R_2	0.94	1
R_3	0.75	2
R_4	0.44	3
R_5	0.00	4

(Similarly, we can calculate for others)

* Similarly for our case say, we need to classify, (Dark Red, Red, light Red)

(linear)

	Dark red	red	light red
DR	1	0.5	0
R	0.5	1	0.5
LR	0	0.5	1

→ (now we can calculate Kappa like in Un-weighted) ←

Evaluating a regression Model:-

(we know MSE)

{Mean Absolute Error}

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

Penalizes the large errors less than MSE, but gives a better idea of average error magnitude.

{Root Mean Square Error}

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

{Mean Absolute Percentage Error}

$$\left\{ MAPE \rightarrow \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \right.$$

- R-Squared (R^2) → {Coefficient of Determination}

↳ It is the proportion of Variation in the depent Variable that is predictable from the Indepent Variables.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$SS_{res} \rightarrow \left(\sum_i (y_i - \hat{y}_i)^2 \right)$$

↳ residual sum of squares.

(error)

$$SS_{total} \rightarrow \left(\sum_i (y_i - \bar{y})^2 \right)$$

→ Variance

mean(y_i) actual values

↳ (total sum of squares)

best case $R^2 = 1$, worst case $R^2 = 0$ (in general)

R^2 value usually range between 0 to 1 but can go < 0 as well. In this case model is performing very bad, than just taking the mean value.

There are limitations with R^2 :-

- A high R^2 does not imply causation between independent and dependent variable.
  - R^2 measure how well the indepent variables are associated with the dependent variable, not whether they cause the change.
    - → ex: If you have a high R^2 between ice-cream sales and temprature, it shows they are related but it doesn't mean temprature directly causes people to buy ice cream.

Date: / /
Page:

- Omitted Variable Bias:
  - It doesn't Reveal if imp Variable have been left out of the Model.

- Correct Regression:
  high R^2 doesn't mean you have chosen best model types for your data.

- Appropriate Variable:
  - R^2 does not indicate if the most appropriate set of independent Variable has been chosen

- ~~Collinearity~~

- Transformation:
  - R^2 does not indicate if model might be improved by using transformed Version of Variable.

- Sample size
  → R^2 doesn't indicate if there are enough data points to make solid conclusions.

  we can get a high R^2 with very less data points.

- Collinearity → R^2 does not reveal if collinearity is present
  → (when independent variables are highly correlated)

  It is bad because:
  - ↳ (It makes it difficult to understand effect of each variable,
  - ↳ It adds redundancy, increasing our model complexity)

(To Address some of these we use ~~R^2 Adjusted or $(\bar{R})^2$~~)

{Adjusted R^2 , $(\bar{R})^2$}

↳ address the problem of overfitting to better for model comparision.

$$\bar{R}^2 = 1 - \left(\frac{(1-R^2)(n-1)}{n-p-1} \right)$$

n → number of observation

p → is the number of predictors (features)

Train Test (Cross Validation) Split.

In machine learning we split our data in 3 categories → (Train, Test, CV)

Out of this we train our model on "Train" set, then perform "Hyperparameter tuning" on "CV" set.

After finding the best set of parameters, we then used "test" data to evaluate how our model is performing.

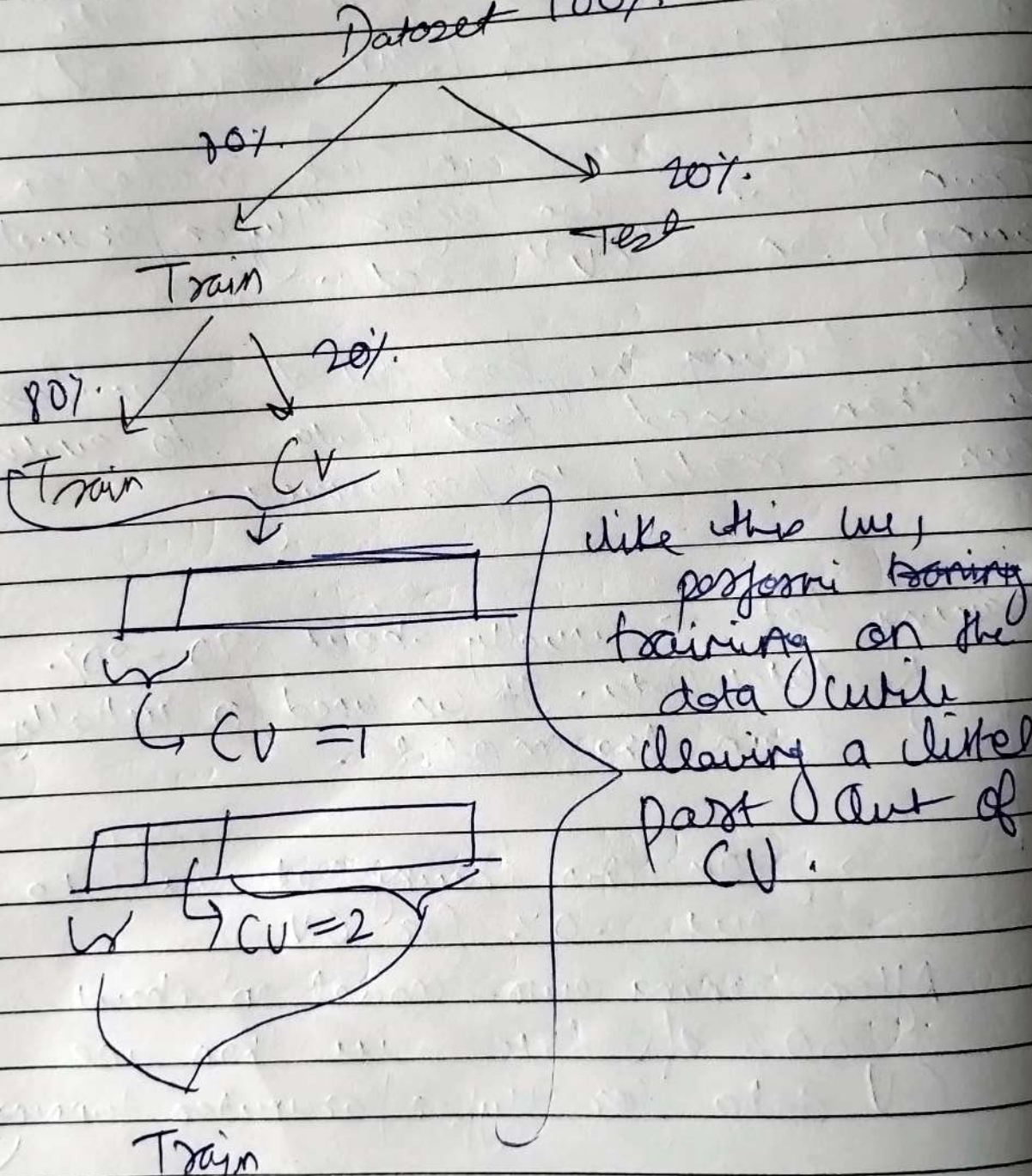
- Training Set → usually about 70-80% of total Data. It is used to initially train the model.
  - → From this training set we take out 20% for CV

  After training our model on about 60% of data, we then use CV data for Hyperparameter tuning.

- Test: After the above step is completed we then use the choosen set of hyperparameters & evaluate our model on test data.
  (generally about 20-30%)

Types of taking Cross Validation.

① LOOCV (Leave one out CV)



generally not used due to being "Time Consuming" also this is prone to "Overfitting".

② Hold out CV

Training

- this time we split the data but first by creating a Random-State (Shuffle).

{Train} {CV}

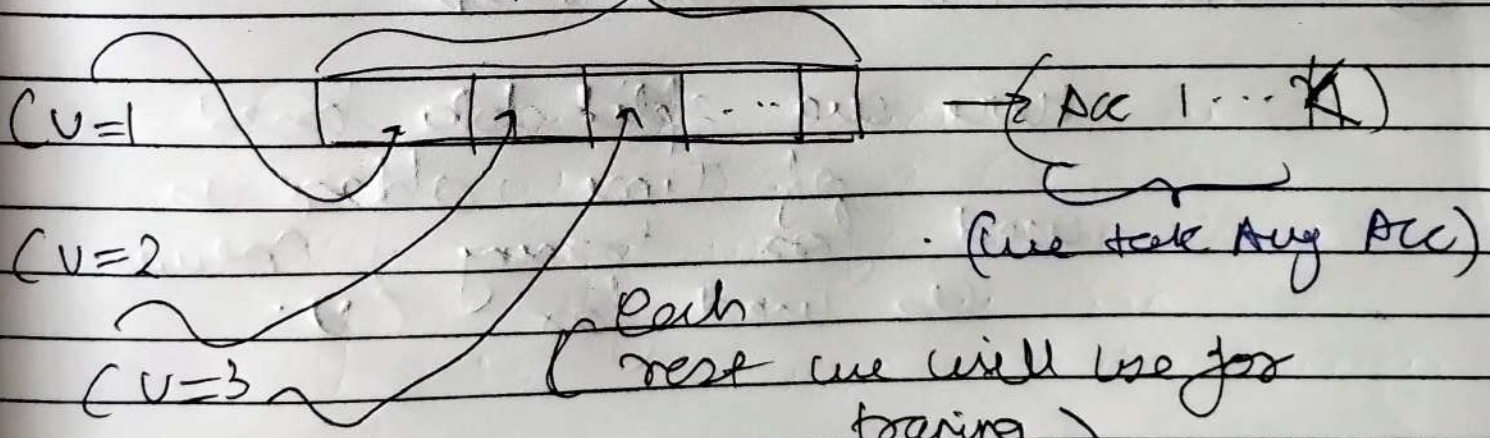
③ K Fold Cross Validation.

(Train) → 1000 points

Out K = 10

then (CV = 1000/10 = 100)

10 section Rest 900 in Train.



(Both of these have a problem of Imbalanced dataset)

④ (Stratified K fold CV)

(Here we take equal proportion of data from different categories)

Say

900 → A
100 → B

Train ← CV (out of 900 say 50%, out 100 say 50%)
Test ← rest

(will work with Imbalanced dataset up to some extent)

⑤ Time Series Cross Valid.

Here we split data based of Time, older is used for training, newer is used for CV.