

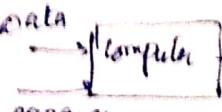
Machine Learning (ML) :-

Page NO.: 1

- It is a field of Computer Science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed.

In traditional coding:-

if we want to add Numbers... This program is explicitly coded to perform sum of 2 nos.


Then we write program to add two members output and now if we pass any two numbers it will return me the sum of two nos. but if I pass three Nos then the program will not run.

In ML what we do is we pass an Excel sheet which has 2 nos and their sum output again 3 nos and its sum and so on and we train the Machine Learning algo on this data and now if we pass any no. of input to this algo it will return their sum (since it has been trained like that).
↳ The algo will recognize the pattern of addition.

* Some scenarios where ML is useful

- Spam Email Classifier:

Suppose as a software developer you are asked to write a program to check if an email is spam or not (You already have a lot of emails & you know which are spam and which are not). Now suppose you wrote a code using lots and lots of if-else statements like if there are more than 3 discount or if there are many pictures or if there are more than 7 flags etc. present in the email then mark it as spam. Now suppose the advertisement company knows about this and now instead of using flags; it uses Big in the email, then now you have to come back and make changes in the logic of your code... But this is not the case in ML... Even if the company changes flags to Big the ML algo will recognize it (since it has been trained on a large dataset) and will mark it as spam.

Image classification:-

problems where there are so many cases that we cannot write it down in a code like....

let's say we are asked to write a program such that it can classify the image ~~has~~ a dog or not. Now there are so many breeds of dogs & each dog has so many features that we cannot write cases for each breed and then for each feature in such cases we write ML Algo and train this Algo or a large data set such that if a new picture is given to it it can classify if it is a picture ~~has~~ a dog or not.

We'll have to train the ML algo just like humans are trained.

Data mining

Kahi kahi kuch info in the data
bahot hi jyada hidden hota hai; jo sifir visualization se dikha nahi pata.

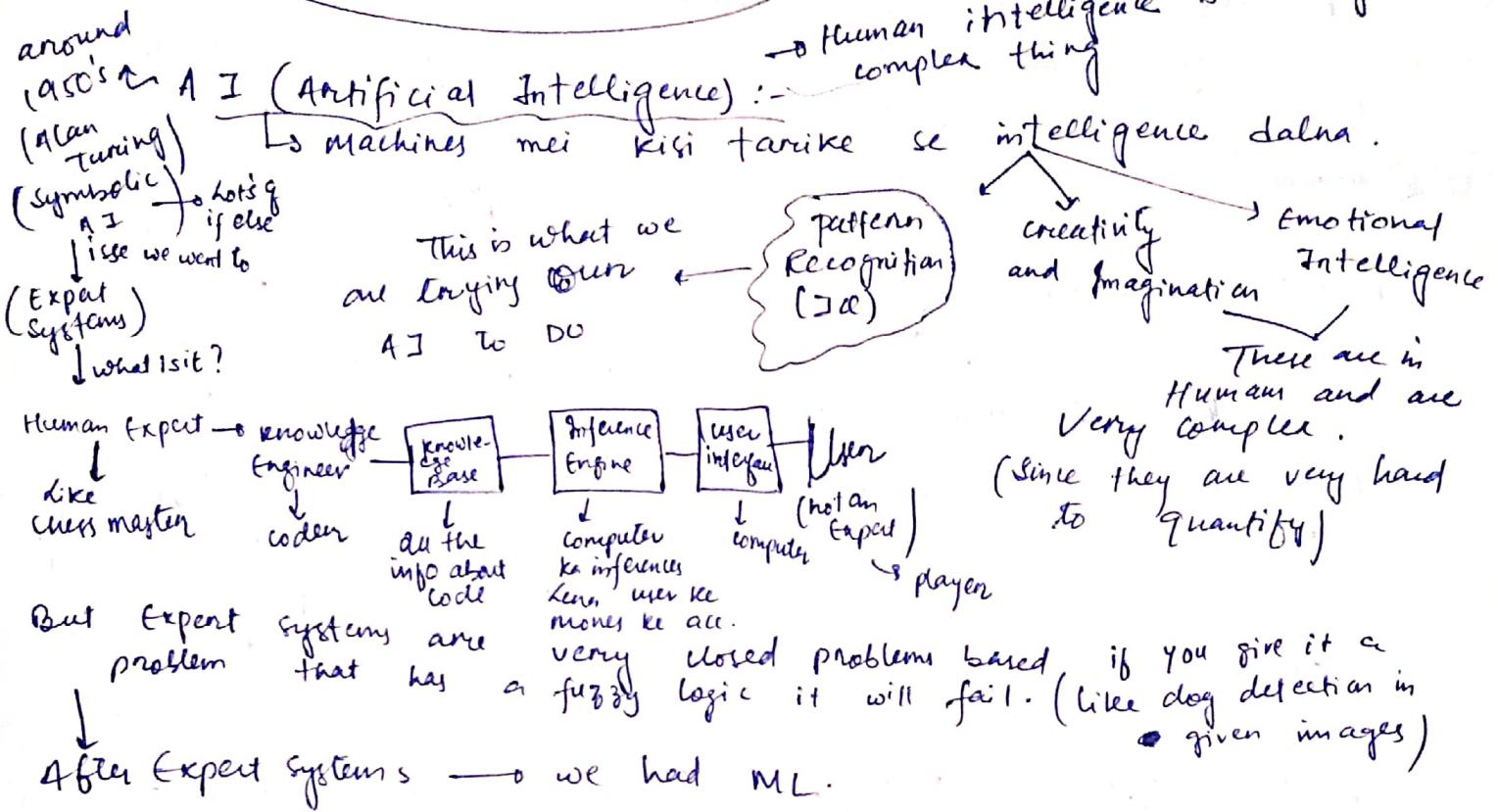
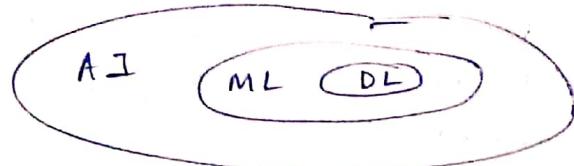
Bahot jyada hidden jo patterns hote hain
unko dhundne ke liye we use data mining
Like: email spam classifier ke liye humne Ek ML Algo likha
and usko train kiya and then usko deploy kiya
Now after some time of the Algo doing its job we
look for what patterns it has built, what are the keywords
that it has predicted on its own for spam email
check... these are the info we have and in me \exists agar
hum kuch important cheez nikal pa rahi hain apne
data ke baare mei jo Data Analysis \exists pata nahi chel
paya tha then we call this Data mining.

* ML Bahot pehle hi aaya tha but isko recognition Bahot late mila sinle one of the major factor on which ML runs is Data & tab Data jyada tha nahi ya use stone
nahi kar pate the and agar use stone kar bhi le then
utne strong hardware/machine nahi tha jo un algos ko ithe
same data pe run karwa paye **But** with the evolution
of internet - These problems are solved.

Data Analysis: It is
a part of Data Science
where you search
for hidden pattern
or hidden meanings/info
in the data.

AI V/s ML V/s DL

page 2



ML (Machine Learning) → Already discussed on page 1

→ Diff b/w ML & symbolic AI is in ML we don't do explicit programming (Hence we write corresponding code likha)

In ML → we have data (input + output). → we feed it to our system (algo) and its job is to find the rules / patterns such that when it gets a new input using those patterns / rules it can give an output or prediction.

In symbolic AI → we were writing the rules for the recognition of a dog in the picture.

But in ML → the system is giving the rules to us.

DL (Deep Learning) → The process is same like ML

Data feed karne se system ko - system rules / pattern form karne and based on these patterns / rules system new data pe predictions karne.

→ It has Algorithms and these Algorithms are inspired from Biology (Neurons) (but DL doesn't work exactly like our Brain)

→ DL is a mathematical model, it's just that the core unit of it neuron (perceptron) is based on Biological neuron.

Buzz of DL.

There are certain things that ML cannot do properly if we are working on some fuzzy logic based problem where the features are not obvious → then DL may help

In ML we provide the features

like say Resume dekhkar Batana hai ki placement hogya ya nahi then in ML the features ^(created by us) has to be given to the ML model when training the model

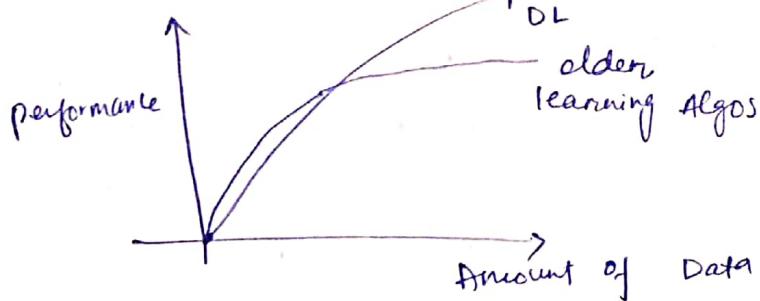
^{Batana hai ki placement}
And this features will be created by the expert in the field.

But in the DL Space, we just have to provide raw text and the system will automatically generate/extract the features.

More the layers of neurons in DL the system's efficiency / power improves. (Since har layer ke saath aar deeper in the data jo chups hua hai wo apan nikal pate hui)

More data you give to DL models → their accuracy improves *utna hi.*

In ML after a certain point their accuracy stabilizes — feeding more data will not improve the Accuracy.



- Image related Tasks
- Text related Tasks
- DL works better.

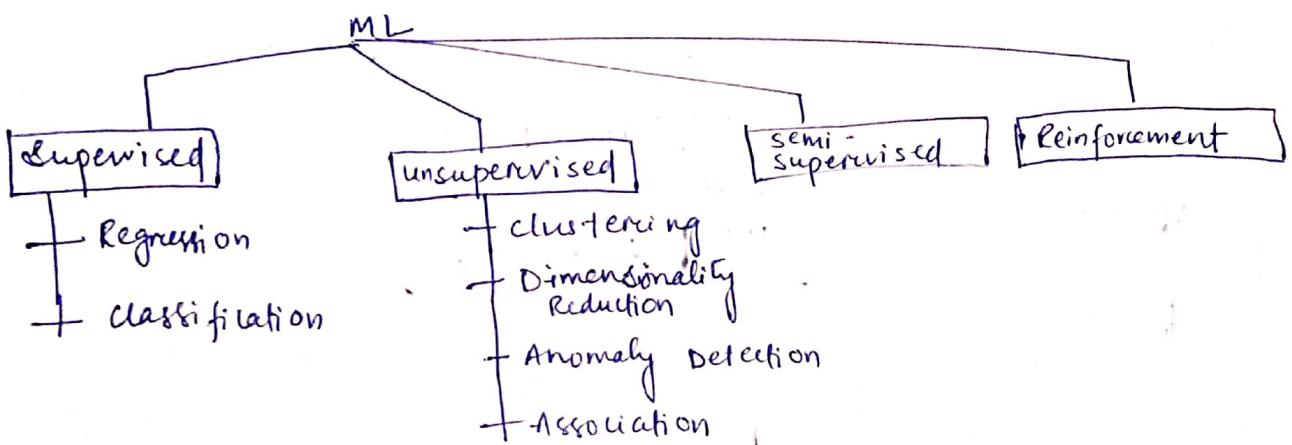
- * Small data pe ML hi better accuracy de dega
- * on Large set of Data we use DL

TYPES OF ML

—ML ke types alag alag cheezo pe depend karne hain

1st. The amount of Supervision Needed for a ML Algo to
Get Trained

Based on amount of Supervision Needed ML
can be of following types



Supervised ML

ML means learning from data or getting Trained on your data.
i.e. in Data we have (input + output) and we want to find a
relationship b/w input & output so that ER naye input ke liye
aap output bata paao then this is called Supervised ML

Suppose we have ⁵⁰⁰⁰ students data (Input + Output)
For each we have: IQ | CGPA | Placement →

<u>IQ</u>	<u>CGPA</u>	<u>Placement</u>
87	7.1	Y
111	8.9	Y
75	6.3	N

The input col's are IQ & CGPA &
output is placement col" so
placement col" depends upon the IQ & CGPA data.

Now we want to draw some mathematical Relationship b/w
the input & output such that if we give the model the inputs
of a new student say IQ = 73, CGPA = 7.3 then what will
be the output (Placement: Y/N)

→ Agar aapke paas aisa dataset (input + output) hai & aapko input
& output ke beech mathematical Relationship nikalna hai such that
new input data ke liye output predict kar paye → supervised
ML

parts of supervised ML

+ Regression
+ classification

Data

Numerical

Ex:- Age, weight, IQ, CGPA

Categorical Data

+ Gender

+ Nationality

Regression

are working on a ML problem which if we have a data set (input + output) & if the output is numerical then that ML problem is called Regression.

Ex:- IQ + CGPA is INPUT and output is package kitne ka laga

IQ	CGPA	package
8.7	8.9	6 L
111	7.9	4 L

Classification

if we are working on a supervised ML problem and the output is categorical then the problem is called Classification.

Ex:-

IQ	CGPA	Placement
79	7.9	Y
110	8.2	Y
87	6.1	N

More Examples: - (i) Given all the details of a house you have to predict the house price (Regression)

(ii) mails ke baane mei bolot kuch bataya hua hai you have to predict mail spam hai ya nahi (Classification)

(iii) Saare weather conditions ko dekhan predict karne ki aaj barish hogi ya nahi (Classification)

(iv) Alag alag images ko dekhan predict karne hi image mei kitne dogs hai - (Regression)

Hence:

Supervised ML Input + output data → output Numerical → Regression
→ output Categorical → Classification

Unsupervised ML

Loge me basically bas input data hoti hai output data nahi hoti.

Since output hai hi nahi toh yaha task prediction toh ho hi nahi sakte since we don't know ki predict kya karna hai.

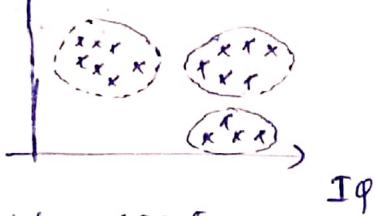
Ex: 8000 students data (only input (IQ + CGPA) output :- placement ka data hi nahi hai)

To in such case we do

- + clustering
- + Dimensionality Red.
- + anomaly Detection
- + Association Rule Learning.

Clustering:-

We can plot the input data (IQ + CGPA) on the 2D coord system.



Fact Now we can ourselves create a Label Δ data ^(new col)
 $0 \rightarrow$ more IQ + more CGPA category
 $1 \rightarrow$ more IQ + less CGPA category
 $2 \rightarrow$ less IQ + more CGPA category.

if we have an E-commerce website know the types or the categories of our customers then it is important to

and Note:-

if we have n-dimensional input then clustering can categorize it into an n-dimensional hyperspace which is not even visible to the naked eyes.

Hierarchical Clustering :- categories ke andar bhi clustering lagaken clustering karuna.

clustering algorithm ye detect kar sakte hai ki konse students same group mei hone chahiye jise mei apne students ko different categories mei divide kar sakte hoon. Toh isse hum kisi bhi incoming student ko categories mei dhal sakte hai accordingly.

Scanned with CamScanner

Dimensionality Reduction:-

Sometimes kisi problem mei input data mei bahot saree columns ho jainge (like 1000 input col's) — in such case the algo will run slow and 2ndly expt. ke baad jyada input col dhalne ki result mei kuch khas jyada improvement hota nahi hai (in ML) hence kuch col's ki jarurat hi nahi hoti then Dimensionality Reduction will Remove/ group those extra col's.

Ex:- Suppose we have a large input Dimension/ col's of which two are:-

No. of rooms	No. of washrooms
.	.

then Dimensionality Reduction will combine this colⁿ to a new colⁿ sq. feet area for room + bathroom (This is called feature extraction) (Latent)

Here we reduced the Dimension/ col of input.

Algorithm used is PCA

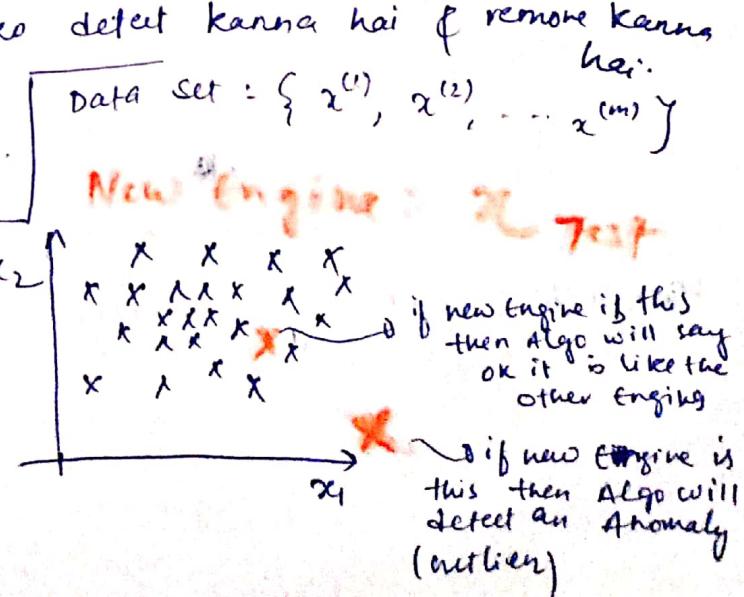
Another use of Dimensionality Red is visualization :- when we have a lot of input Dimension/ col then we cannot plot them on 2-D or 3-D to visualize then we use Dimensionality Reduction to bring those input col to 2 or 3 and then plot them and see the same relationship that we had in higher Dimensions.

ANOMALY DETECTION.

Kuch gadbad (outliers) hei usko detect karna hai & remove karne hei.
 Manufacturing mei defect → Credit card processing defect → Loan approval defect → etc.

Ex:- Aircraft engine features:

- x_1 : heat generated
- x_2 : vibration intensity



Association Rule Based Learning:-

products have to be kept in organized way.

Suppose we have to setup a supermarket and now we have 5000 - 6000 products, now how do we decide ki kiske bagal mei konse product rakhega.
(kis product)

For that we collect the bills of last 1 to 2 years and find patterns among the products bought like
like out of 100 bills jab tahi ^{80 times} milk kharida gya 70 ^{60 times} 1 eggs shi liye gaye then hum milk and eggs ko ek saath rakh bagal rakh denge (since there is a strong association between milk and egg)

Interesting Casestudy in US: Beer + Baby Diaper

SEMI SUPERVISED

Partially Supervised & Partially unsupervised

The label colⁿ (output colⁿ) create karne mei for the human effort legla hai data set on which Algo will be trained, (like et the end kon batayga is dog hai ki nahi picture mei, and kon batayga ki placement hea ya nahi)

Hence label create karna difficult kaam hai but input data can be fetched from anywhere (web scraping)

Ex:- To train an Algo we need dataset so to get input data (say images of dog) we can webscrap but now to get the output data (col) ki images mei dog hai ki nahi we have to do it manually and that will take human effort & cost money

So idea is ki hum kuch Data ko manually label kare and baaki ke dataset khud label ho jai is the key idea behind Semi Supervised ML

Ex:- Google photos:- jab hum photos click karte hai toh wo ye automatically identify karne legla hai ki iss photo wale aadmi ko mei aur 4,5 photos mei dekha hai toh wo iss photo ko vnsab ke saath clustering/Group and agar hum EK photo ko label (^{mummy} "mummy") kare toh automatically uss Group ke baaki saane photos par shi same label (^{mummy} mummy) kandega.

REINFORCEMENT ML

Hum Data de le hi nahi hai - system ke paas data hota hi nahi hai.

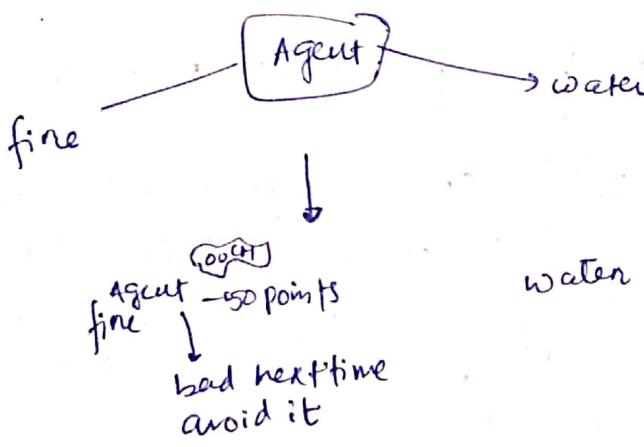
Humara Algo exdum scratch \Rightarrow start karta hai
Seekhna & dheere dheere improve karta jata hai
(Like us / Human Beings)

Ex:- Self Driving car :-

pehle \Rightarrow koi data nahi hai,
wo road pe utar kar gradually
Learn Ranta hai ki Kaise gradi chalni
hai

Like hum college life
mei humara paas
pche \Rightarrow data nahi hai
ki kaisa behan kaha
hai but hum dheere
dheere jattiya karke
seekhte hai.

Reinf. ML mei jo Algo hota hai use hum Agent bolte
hai:-



① observe

② select action using
policy
Agent ka rule book like if may have go to fine

③ Action

④ Get reward or penalty
like fine has penalty

⑤ update policy (learning)
like shot go to fine

⑥ iterate until optimal product is found.

* Company Deepmind (acquired by Google)

made an agent (using Reinf ML)
and that agent v the master

defeated straight 4 out of 5 games.

for Go Game (chinese game more complex than chess)
of the game

* Production is basically the servers on which our code is going to run.

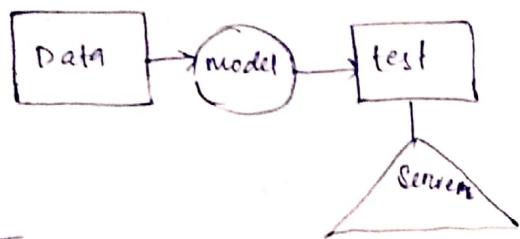
Server ka environment jaha par code run kar raha hota hai use hum production environment bolte hain and jab hum uss code ka run raha hain apne machine pe toh hum uss development environment bolte hain.

ML on servers behave differently & based on that ML is divided into two categories:

BATCH (OFFLINE) ML VS ONLINE ML

BATCH ML

Here we take up our entire dataset and train our ML model on an offline system (our computer) and then once the model is trained then we deploy it on the server (we don't directly train on the server since it is costly & time-taking)
(there is no incremental learning i.e. chale chale pustekh pe train nahi kar raha)
& since the date set is entire (large).



The problem here is the model is static:- Ek baar jo wo sikhi gya wo sikhi gya, model kuch naya nahi sikhi raha hain, wo purane training ke dam pe hi kaam karo raha hain but the business evolves over time and so the model also has to.

Ex: Suppose Netflix ke liye humne Movie Recommendation model banaya on the Netflix Database, ^{Movie} Data & then Netflix ke server pe deploy kar diya and ab ye model movies Recommend kar raha hain but Agar Netflix New movies dhal de DB mei then ye model in naye movies ko recommend nahi kar payega.

Hence, in a way alpha ML model ko constantly evolve karte rehte padega Data ke Saath.

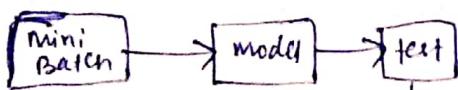
So in Batch ML after New Data is added (in a frequency of 24 hrs or 1 week or 1 month or 6 months or etc) we pull down the model Retrain it on the updated Data (old + new data) then test it & then deploy it again on the server periodically...

Disad. of Batch ML

- Lots of Data (May be data bolat fast se (update) ho raha hai in Batch ML Limitation is ki purne data pe hi train hoga then on such Big Data there is high chance ki code break hojai)
- Hardware Limitation (may be our ML model kisi aisi jagah pe chal raha hai jaha se apka instant connectivity nahi hai toh updated data pe Periodically retrain Karna difficult hogi)
- Availability (Let's say hum ek Social networking website par kaam kar raha hai & humne ek model banaya & we have used Batch ML on 24 hr period ab agar achanak koi baat fail gji then hamara model use to newsfeed generate karta hai on basis of user's interest & user's interest nahi karega, instantly since wo 24 hrs baq news data pe train hone ke baad hi ye newsfeed us breaking news important bhi karega & yaun hamara news feed important bhi nahi hai. ab agar achanak koi breaking news aagaya ya show hi nahi karega, since wo 24 hrs baq news data pe train hone ke baad hi ye newsfeed important bhi jayga joki ab itna jisme hum interested hai & earth kar raha hai hence new data feed kar raha hai model ko)

ONLINE ML

Online ML is done incrementally matlab jo training hui with incremental data hai and hum chale chale batches mei apne model ko data feed karta hai (mini batches) sequentially & har training/batch ke baad mod of improve karta jata hai and since the mini batches have small chunk of data online, ye training hum production server pe bhi kawa sake hai.



continuous inflow of new data
how is doing prediction on old data & new data & math hi train & learn shi kar raha hai on new data and hence with new data model ka performance improve karta jata hai.

Ex:- - Chatbots (Google Now, Alexa, Siri) are examples of Online Learning.

- youtube feeds personalize kota rehta hai according to our searches & videos.

When To use - ONLINE LEARNING

1/ When there is a concept drift :-

Suppose we made a ML model for a problem but the nature of the problem is volatile / changing over time then we should use online Learning.

2/ cost effective (online Learning is cost effective in comparison to Batch Learning)

3/ Faster Solution

→ How to implement

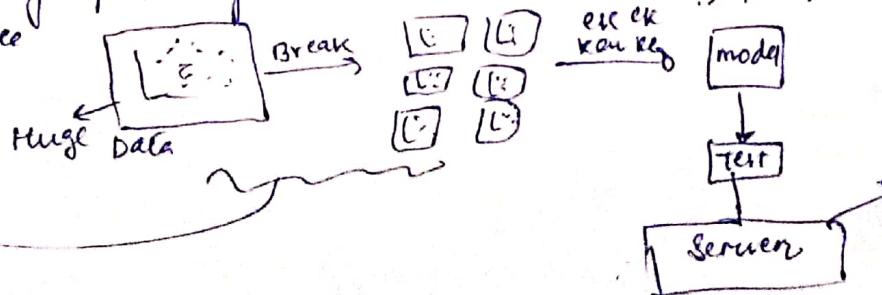
- using scikit Learn
- using River Library

River is a python library for online ML.

It is a result of a merger b/w creme & scikit-multiflow.
River's ambition is to be the go to library for doing ML on streaming data.

Tricky Thing to set
→ Learning Rate :- in online ML, Learning Rate ye decide kanta hai ki kitna frequently hum model ko new data pe train karenge since new data inflow hota consistent hai humne aisa Learning Rate rakhna hoga ki apna model purane cheze ishi yaad rakhe & nauji cheze bhi Likhte jai.

→ out of core Learning The data is so huge that you cannot load it at once on the memory for Batch Learning - in such cases we can use online Learning where we can incrementally train the model by sequentially providing Data in Mini Batches.



server pe model ko Batch ya online rakhna hai that depends.

DISADVANTAGES of Online ML

- Tricky to use / when there is huge inflow of data online
(then setting Learning Rate, controlling the inflow etc. becomes very tricky)
- Risky (because data ke according mere model ko, behavior change hone lagta hai & agar data hi galat (spam) rane lage toh apna model kafi biased hojai ga.)
so :- An active monitoring system using maybe some anomaly detection Algo like kisi bhi anomaly data ke aate hi system ko offline rando ya us data ko reject kande.

Key Differences b/w Batch v/s Online Learning

<u>offline Learning</u>	<u>Features</u>	<u>online Learning</u>
Less complex as the model is constant	Complexity	Dynamic complexity as the model keeps evolving over time
Few-computation, single time batch-based training	Computational power	Continuous data ingestions results in consequent model refinement computations
Easier to implement	use in production	Difficult to implement & manage
Image classification on anything related to ML - where data patterns remains constant without sudden concept drifts	Applications	used in finance, economics, health where new data patterns are constantly emerging
Industry proven tools:- E.g. Sci-kit, TensorFlow, Pytorch, Keras, sparkMlib	Tools	Active research / New project tools - e.g. MOA, SAMOA, scikit-multiflow, StreamDM

Based on how our ML Model Learns ML is of two types

- + Instance Based ML
- + Model Based ML

Learning in ML → memorizing → Instance Based Learning

→ generalizing (concept/underlying principle/pattern, Samajhna)

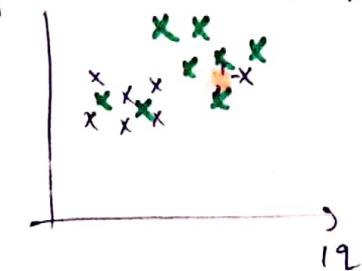
Model Based Learning

* Future mei jab ML Algo padho then try to identify ki wo model instance Based Learning kar raha hai ya Model based learning Lazy LEARNER

Instance Based Learning

Consider we have dataset:-

Based on this data set we need to predict the output of a new student based on his input Now there is an instance based model/Algo Approach this problem:-



input	output
IQ	cgpa
80	8
70	7
?	?

Placement (Y/N)

Y
N

to predict the

→ output is categorical
∴ classification problem.

x - yes
x - no

In Instance based ML when the Algo gets a training Dataset the Algo doesn't learn anything, it stores the Data as it is and instantly answers when a question is asked.

Say we have new data in input/ query point $103, 7.5$ as shown above based on shortest distance then we look at its nearest neighbours (3, 5 etc depends on us) and we find ~~nearest~~ of query point which ~~are all~~ ~~different~~ type of neighbours are more (here green) so we say the output of the query point will be like green Neighbours i.e. yes.

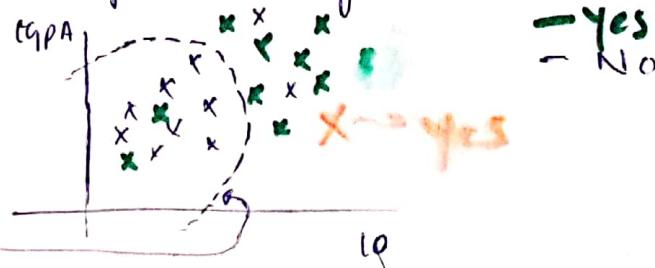
This is the logic of the KNN Algorithm
(K nearest neighbour)

Note in Instance Based Learning, jab tak naya point nahi aaya, tha tab tak apne model ne kuch kiya hi nahi jab unse paas ek query point (input new) aaya tab use same method se output diya. (this has no training or learning concept)

MODEL BASED LEARNING

Now let's discuss the same problem using Model Based Learning.

10	CGPA	placement(Y/N)
80	8	Y
70	7	N
:	:	:



10

Now when we give data set to the algo, the algo tries to understand the behaviour/concept/pattern inside the dataset by drawing a mathematical function ^(b/w input & output) → ye apne model ne data ke upar sun ya train karke sikha. ^{Called model parameter}

Most of the ML algo use this approach → finding a mathematical reln b/w input & output of result is a decision function ^[esp in classification problems]

* Now after getting the decision fn, even if we remove all the training Data, the algo will be able to predict an output for a given input Data. But this is not the case with Instance based Learning where we need the Dataset to find the nearest neighbour.

NOTE:- Always try to identify if an algo is Instance or Model Based.
Examples of Model Based ML

Linear Regression | Logistic Regression | Decision Trees

Key Differences

<u>MODEL BASED</u> (usual/conventional ML)	<u>INSTANCE BASED</u>
- Prepare the data for model training <small>(data preprocessing)</small>	- Prepare the data for model training <small>(no diff here)</small>
- Train Model from training data to estimate model parameters i.e discover patterns	- Don't train model. pattern discovery postponed until <small>scoring query received.</small> <small>new input data / query point</small>

- Store the model in suitable form
- Generalize the rules in form of model even before scoring instance is seen new input data or query point
- Predict for unseen scoring instance using model
- Can throw away input / training data after model training
- Requires a known model form since base kis for ko store karne hai
- Storing model generally requires less storage

- There is no model to store.
- No generalization before scoring. Only generalize for each scoring instance individually as and when seen.
- Predict for unseen scoring instance using training data directly
- Input / training data must be kept since each query uses part of full set of training observations
- May not have explicit model form pure data set ko store karne ke liye
- Storing training data generally requires more storage.

CHALLENGES IN ML

- Data Collection → When we are making a ML project at college level then the data collection is not difficult. But when we are doing actual ML for some company the Data accumulation/gathering is actually quite difficult. In such cases we can either fetch Data from an API or we can do Webscraping.
- Insufficient Data / Labelled Data

Suppose we have two Algos A & B working on the same problem & consider Algo A is better than Algo B and we give 100 rows of data to Algo A & 10⁶ rows to Algo B to work on same problem then Algo B will perform much better than Algo A just because it has more data.

Hence agar humne paas bahot jyada data hai toh farak mali padla ki hum konsa Algo use kar sake hai, this is called the unreasonable effectiveness of Data.

But we have mostly small or medium data \therefore Algo matter

Karta hai:

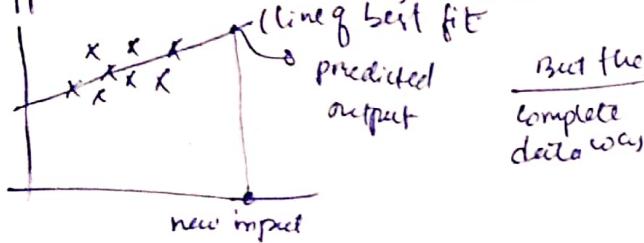
Again: Agar hum data collect kar bhi le, the labelling it (like cat hai ya nahi in picture) is a tedious job.

Data mila \rightarrow kya wo sufficient hai — sufficient hai toh kya wo labelled hai.

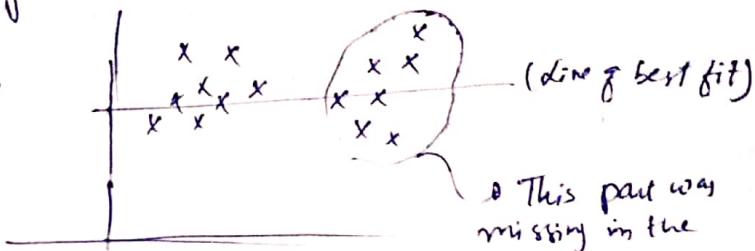
Non Representative Data:

Suppose we are solving a ML problem but hamare pass jo Data hai wo Aadhi Kahani hi bata raha hai Ex:-

Suppose we are solving linear Regression



But the complete data was



Kahi Kahi jab hum Data gather karne hai toh Sabhi jagah ∞ gathers na karke kisi ek jagah ∞ utha leta hai then this data problem ka proper representation nahi hota

Like: we want to Survey ki next T20 worldcup konki team jitegi:-

Agar hum bas Indian logo ∞ hi puche toh wo definitely India bolenge (biased) lekin ye jo data collect hua wo non-Representative data hai. This is called **Sampling Noise**

valid/proper data ke liye humne Survey har us country mei karne chahiye jo team khel sake hai.

Kahi Kahi Agar sab jagah se Data le toh thi dikkat ho jati hai

Like: Agar hum har country mei thi puche toh Sabse in dia sola linee waha bhi mostly Indians hi hai \rightarrow This is called **Sampling bias**

Sahi representation of data ke liye:- 100 Ind, 100 Aust, 100 Eng ... ∞ pucana hoga.

(imp)

Poor Quality Data :-

Dara mei bahot saane outliers hai, missing values hai, abrupt values hai, alog alog format mei values hai and in Sab cheezo ko sahi karna mei bahot jyada mehnat lagti hai.

(imp) Most part of ML Project is improving the quality of Data.

Irrelevant Features (columns)

Apne Dara mei kuch aise columns hai jo kuch contribute hi nahi kar raha apne analysis mei - toh apna model acha perform nahi karega

In ML :- There is a phrase :-

Garbage IN, Garbage OUT → Agar ML model hi kachra dhaloge toh output kachra hi milega.

Suppose we want to analyse which age group are more fit & ^{column} data, we have Age | weight | height | location the location col will not add anything to our analysis hence we remove it.

kabhi kashii hoon do features ko mila ke ek single feature lana dete hai (yaha BMI) jo same analysis dega → And this is called **feature engineering**

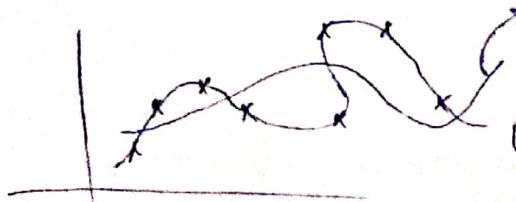
Overfitting: (Big Villain in ML Algos)

Humne ek ML model ko kisi data pe train kiya and ur model ne data ko itni liya (matlab concept) (pattern nahi sikha) toh agar is model ko kisi naya data de toh iska performance acha nahi rhega. Kisi model ko bas purane data pe hi kaam karna chahiye hai.

Ex:- Guwahati gya - see movie - ticket - 500 ₹ - we assume Guwahati is costly but aisa nahi hai hum bas movie ticket ke basin par pure city ke prices predict kar raha hai

slly open ML Algo training data set ko dil hi logoコレ hoi xi jo training data set mei hai wohi sahi hai and baaki sabhi waisa hi hai

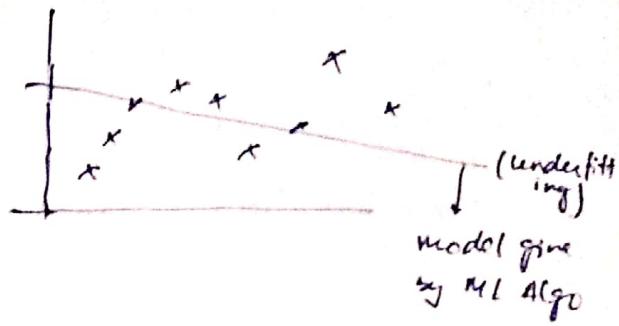
→ Aisa kuch actually fit nahi chahiye tha.



(Overfitting) → jo humne ML model de banayi it is trying to touch every data point in the training Data set

underfitting. (opposite of overfitting)

toyo na training data pe acha result dega & nahi new data pe



Software Integration

End goal of ^{some} ML model is to help the user by becoming a part of ^{the} ^{some} software.

Integrating the ML model with software is not an easy task. Software mei alag alag platforms hote hain - windows, Android, Linux, some old OSs, server OS etc.

Since ML is relatively new so integration of ML model with softwares (like java, javascript) is not very stable.

* Try making end-to-end products like user needs use kar paye
(Try converting ML models to Android, Apps etc.)

offline Learning / Deployment

Lo batch learning wali problem (ek baar model train hoken server po rha isse deployment nahi bhot cost lagta hai & is very difficult)

toh wo update nahi hata - update karna ke liye wapas offline lao - new data pe train & test karne & then phir upload on server)

Today leading providers AWS, Google cloud platform, Azure, they are providing the services that helps deploy ML models to the user but still they are not quite stable (like ^{they are} for softwares)

Cost Involved

If you are working on somewhat bigger scale & deploying it & you are having many users on daily basis then the hidden cost involved is quite big and that the company will probably not allow while deploying on server.

So whenever you make a ML model → convert it to a proper software product and deploy on server and send it to actual users

It just like Devops Now there is something new called MLOps

Applications of ML

B2B Examples. (Business to Business)

B2C examples are easy:
youtube / Insta Recommendations,
Amazon, flipkart etc.)

Retail - Amazon / Big Bazaar

- * Amazon has 6 crores products. And Amazon has an annual Great Indian Festival sale and during this season the sales increase so Amazon has to increase the products in the store but it cannot increase all the 6 crore products in the store.
So it is important to know ki kaise product ka sales increase hoga during the SALE.
And Data of previous 5-6 yrs of Great Indian Festival sale are given to data scientists and they use ML Algo, Data mining to find insights ki kaise product ka stock up karne chahiye & kaise ka nahi.
(Note:- yaha small mistake can lead to losses of crores)
- * After buying product in Big Bazaar, during payment they ask for your phone no. (~~wind~~) — Actually using these phone number they do data profiling, ki ye aadmi konsi cheez jyada khanda hai (js he health consious etc) and then sell it to different companies who do target marketing (same is done by Google, facebook, etc.)
- * positions of products in supermarket is done by ML (Association Rule Based Learning)

Banking & Finance:

- * Loan Baki ko nahi milta — pehle apna profile submit karne padta hai — iss profile ka analysis hota hai at two levels \hookrightarrow ^{ML Algo} _{Loan officer} — analysis by ML algo — this profile is compared by the past defaulters (jinhone loan nahi and agar strong correlation aaya then may be ^{chukaya} get a loan & your profile will be rejected and if correlation is low then the profile is sent to ^{Rejected} _{loan officer} who checks the profile manually & loan is sanctioned if good).
- * Kaha pe branches khole, customers ke liye kaha pe kaise promotion plan start kare etc.
- * Share market / Trading .

Transportation - OLA

• Din ke kisi particular time pe point A \Rightarrow point B
 Jane ka price bahot high kar diya jata hai jiski
 wohi price din bhan low rehta hai.

This is because the driver aap has a red region
 (region where cabs are less & customer are more - say after
 office hours)

Now the customers book cabs of those cabs in the red region
 take the customers but those cabs outside of red region,
 why will they go extra distance to pick customers at
 the same price \Rightarrow OLA says, if they pick customers
 from Red Region, they will be paid double & this
 double (extra) amount is not paid by OLA to drivers but
 it is taken from the customer itself hence the cab prices
 at certain time is higher. (cost per due to peak demand)

* Delivery systems - path optimization

Manufacturing - Tesla

Predictive maintenance:

In Tesla there are robotic arms for setting up a car
 Now if an arm fails then the car production that day will
 stop. So in each arm they use IoT (Internet of Things)
 sensors that monitors the different metrics of the arm (Temp,
 RPM etc.). And if an arm fails - it fails gradually & a
 signal is sent that a particular arm may fail to work
 and they are fixed even before it fails.

Consumer Internet - Twitter (started in 2008)

Twitter was not earning money initially - it's investors were scared -
 for a long time (in NLP)

- then the head of twitter decided to do sentiment analysis of
 the tweets & earn money from those analysis
 Ex: # WB elections 2021 tweets ko ek taraf kar dega and un tweets ke
 sentiments ko evaluate karega (modi vs Mamta) (social media pe
 jab kuch bolta ya likhta hai toh wo apne personality ko hi kind of
 extrapolate karta hai) (so the tweets are actually valuable)

\therefore the tweets are kind of an opinion poll. and these data is sold
 to Stock Brokers (JP Morgan, Morgan Stanley) (jo stock markets mei paisa lagate
 and these stock brokers un companies mei paisa lagaygi jo already

predicted winning party ko support kar rahi hai. Ab agar un mei predicted
 party jeet gaya toh local public bhi uski company mei paisa lagaygi & stock prices goes
 and stock brokers ko profit hoga since they already spent those shares

Machine Learning Development Life Cycle (MLDLC / MLDC)

so far we focused on why & what.
here we focus on how?

MLDLC - is a guideline that you follow whenever you have to make a ML based software product.

- + Frame the problem
- + Gathering Data
- + Data Preprocessing
- + Exploratory Data Analysis (EDA)
- + Feature Engineering and Selection
- + Model Training Evaluation and Selection
- + Model Deployment
- + Testing
- + Optimize

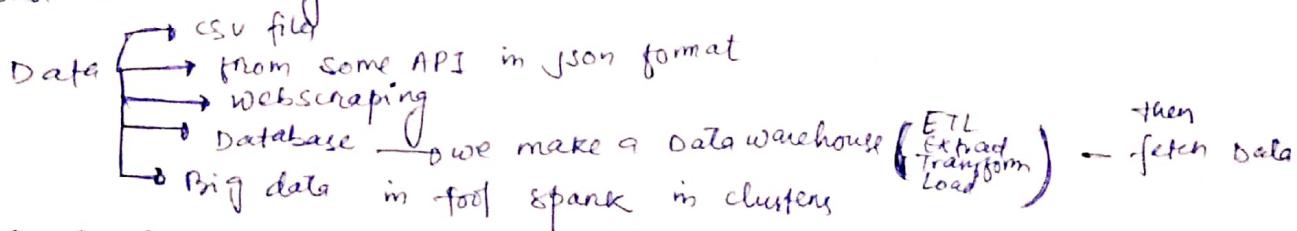
Task: To make a software product that has ML for a company

Frame the problem

Problem kya hai, kya solve karna hai, cost kifha lagoge, Team mei kaun chahiye, how the product should look, who are the users, type of ML model to be used (supervised/unsupervised, offline/online), types of algos to be used, where to get the data

Gathering the data

In case of company projects, the data could be very specific and it's easily available nahi data.



Data Preprocessing

When we bring data by fetching it as above then it is not clean which we cannot directly use (structural issues, missing data, outliers, noisy data, different sources data are not compatible with each other etc.) Hence we need data pre-processing.

- Remove duplicates
- Remove missing values
- Outliers handling
- Scaling (standardization)

Basically, data ko aise format mei leken ana hai jo hamara ML also easily consume kar paye.

EDA (jitha time yake spend kano utna jyada aage ka kaam easy hoga)
Before making the ML model, we should know what we have in our data, the inputs & outputs in the data, their relationships.
Here we make lots of experiments with data, extract the relationships hidden inside the data.

- visualization by plotting graphs
- univariate analysis (blank col ke upar independent analysis)
- Bivariate analysis (mean, std, variance, curve etc)
- multivariate analysis (multiple columns ke beech ka analysis)
- outlier detection (multiple columns ke beech ka analysis)
- Imbalanced data set - Balanced data set
(like cats & dogs ke HQ images same nahi hai for image classification)

Feature Engineering of Selection → existing features mei kuch intelligent changes karne ko that analysis easy hoga.

Ex: Features matlab → input columns

Features are imp since output input cols (features) pe hi depend karta hai
Idea is kashii kashii keeche naye cols (feature) hum create karte hai apne hisab se taaki apna analysis easy hoga
Ex: At qroony + HQ of bathroom → combine eq. features for previous input features rooms & bathrooms (new input feature)

Feature Selection: kashii kashii data set mei bahot hi jyada features hote hain (100 or 200) then hum saare features ko lekar aage nahi badh sake because janurii nahi ki haan input features output ko impact kane so wo remove such input cols/features

Also more the features more the time it will take to train the model. (This we want to reduce)

types

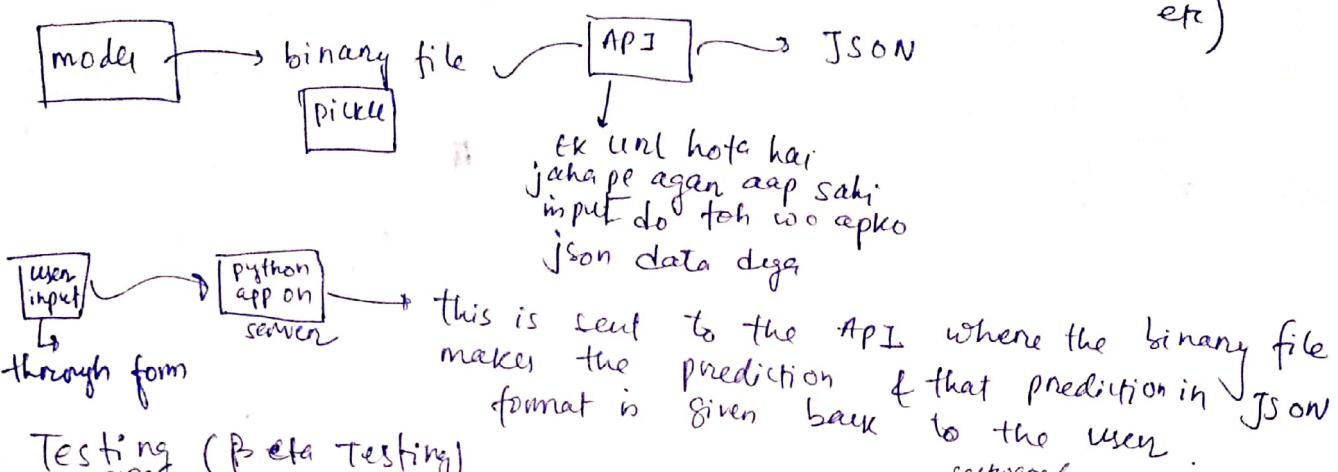
Model Training, Evaluation & Selection
→ Model Training, Evaluation & Selection → bring different ML Algos
After we have the proper Data Set, we bring different Algos (we never actually know which algo will work best)
and feed them the data & train (which will work best for particular data)
(Naive Bayes performs better on test data but hotska hoi ki Koi other Algo to
and hotska evaluate kante hain ki kisika performance kaisa
hai using the performance metric (this tells how good our models
are performing))
And then finally we select one or more algos of the, tune their
best parameters (settings of the algos) to enhance the performance

Ensemble Learning - multiple ml algos ko jod ke ek naya powerful
method/banakte hain

Model Deployment

Now we have a ML model that can make predictions.

Now we need to convert it to a software so that users can use it (software can be: website, mobile apps, Desktop App etc)



Testing (Beta Testing)

To kuch Trusted customers ke paas ya model bhigto hai;

and ask for feedback

A/B Testing

Isse ye decide karne ki kisne jo model banaya wo kitna sahi se kaam kar raha hai agar sahi se kaam nahi kar raha then heem pickle wale steps ko repeat karne hain

Optimize:-

Server pe Launch karne ke pehle hum kuch steps karte hain

- + model Backup
- + Data Backup
- + Load Balancing (bcoz jyada log zarabe hai to unko divide karna same ya karna time mei request serve keren)
- + How frequent we need to Retrain (Rotting prevent karne ke liye this needs to be automated since overtime data evolve karne lag jata hain)
- + Automation setup ki agar model fatiga then use roll back karne waqt ke gane karne

AngelList → site for Job search.

→ ML + DL

Tensors :-

- Tensor is basically a Data structure (to store data)
- Today any ML systems, leading Libraries (Sci-kit Learn, TensorFlow etc.)
use the basic data structure → Tensor. !
by Google
 - Tensor is a container for numbers (rarely strings hum characters & hi store kar leta hai)
Ex: Vectors, matrices are Tensors.

0 dim Tensor is a Scalar (ek no.) }
1 dim Tensor → List of Nos → vector } General form
2 dim Tensor → Matrices }
3 dim Tensor : - - - Tensor

* 0D Tensor / Scalar :- To create 0D Tensor.

(2) (3)

↙
Scalars - 0D Tensor

import numpy as np
a = np.array(4) → To see its dim
print(a) → array(4)
This is a scalar

* array → 1D

2-D arrays → array ke andhar array

3-D arrays → array ke andhar array of like andhar array of so. on.

* 1D Tensors :-

[1, 2, 3, 4] → vector, 1D array, array, 1D Tensor

o. ndim = 1

means the dimension

* in Tensors form:- what is Axis:-

say if we have 2-dimension then there is 2 Axis

" " " " 3- " " " " " 3 Axis

Note:- No. of axis = dim = rank.
of tensor

Creating 1D Tensor using numpy

arr = np.array([1, 2, 3, 4]) → 1D Tensor → isko aap vector
bhi bol sake ho.

↳ arr.ndim = 1

But if asked about
vector [1, 2, 3, 4] → 4 Dim
vector

0 D Tensors \rightarrow scalars (9), (2),

Page 14.

Note:- 1D Tensor - collection of multiple 0D Tensors $\rightarrow [1, 2, 3, 4]$

2 D Tensor - " " " 1D Tensors

3 D Tensors \rightarrow " " "

2 D Tensors

$\rightarrow \begin{bmatrix} [1, 2, 3], & \xrightarrow{\text{matrix}} \\ [3, 4, 5] \end{bmatrix}$

$\begin{bmatrix} [1, 2, 3] \\ [3, 4, 5] \end{bmatrix}$

* 2 D Tensors (Matrices):

1D Tensors:

$[1, 2, 3] \quad [4, 5, 6] \quad [7, 8, 9]$

2-D Tensor
matrix. \downarrow
 $n\text{-dim}=2$
 \uparrow
of axis
 \uparrow
Rank

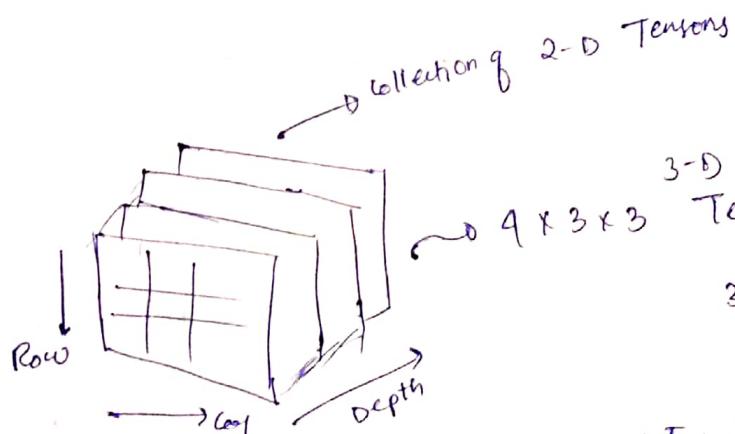
Row \downarrow
 $\begin{bmatrix} [1, 2, 3] \\ [4, 5, 6] \\ [7, 8, 9] \end{bmatrix}$ col

creating 2-D tensor using
numpy:-

mat = np.array ($\begin{bmatrix} [1, 2, 3] \end{bmatrix},$

\downarrow
mat.ndim = 2

* 3-D Tensors:



3-D Tensor
3 axis (Row, Col, Depth)
3 Rank

Sty: 4-D Tensors \rightarrow collection of 3-D Tensors.

(IN ML 0 to 5D tak hi kaam ayega)

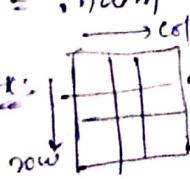
Rank, Axes & Shape, Size of Tensor

Rank = # of Axes = # of dimension = n.dim

Shape \Rightarrow (row ki length, col ki length)

Ex:  \rightarrow 2 dim

shape = (2, 3)

Ex:  \rightarrow 2 dim
dim = 2
shape = (3, 3)

\rightarrow 2 dim
shape = (2, 3)

Size = # of items in Tensor = multiply the shape entries

Note: the concept of row & col's applies when you have a 2D array.

The array:- $a = \text{np.array}([1, 2, 3, 4])$ is 1D

\therefore its shape is a single value iterable
 $a.shape = (4,)$

But say we have it's 2D version:-

$a = \text{np.array}([[1, 2, 3, 4]])$

$shape = (1, 4)$

PRACTICAL EXAMPLES

Example of 1D Tensors

Students data set: (10000)

Classification problem:

CGPA	IQ	STATE	PLACEMENT
8.1	91	WB=0	Y
8.2	97	1	Y

student 1 input:

$[8.1, 91, 0] \rightarrow$ 1D Tensor / 1D Array / Vector
 $\rightarrow \text{ndim} = 1$ But it is a 3-D vector
of Tensor / array

\therefore in terms of vectors it is a 3-D vector

but when we represent them as numbers &

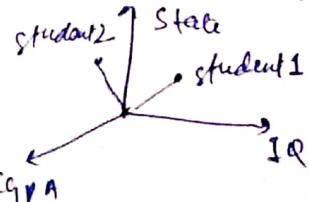
in a Tensor / array \rightarrow it is a 1-D Tensor / array

Say if we have 50 input ^{data}, then as vector
it will be a 50 Dim but as Tensor / array
it will still be 1-Dim

The placement col: $[1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ \dots]$ \rightarrow 1-D Tensor
 \uparrow 10000 entries

QUESTION

Since informing vector it has three elements. In terms of vector there are three axis



Examples of 2-D Tensors

Students Data (10000 students)

Cgpa (9)	state	placement
----------	-------	-----------

input data: student input data

$$\begin{bmatrix} 9.2 & 101 & 0 \\ 8.7 & 97 & 1 \\ \vdots & \vdots & \vdots \\ [-] & [-] & [-] \end{bmatrix} \xrightarrow{\text{10000 rows}} \text{2 dim Tensor.}$$

$$\xrightarrow{\text{10000 rows}} \text{10000th student input data.}$$

In ML jab thi data mile toh uska jo input hoga (collection of input col) uske saare data ko we can store it in a matrix.

placement: $[1 \ 0 \ 0 \ 1 \ 11 \ 0 \ 1 \ \dots]$ $\xrightarrow{\text{1-D Tensor}}$
 or 10000 dim vector

* Examples of 3-D Tensors

IN NLP (When we are working with Textual Data then we see 3-D Tensors) ~~data~~

Ex:- Hi Gautam
 Hi Neeraj
 Hi Jatin \rightarrow This is the text we provide for training to the model & we know ML Algo is mathematic (does not understand text) \therefore we convert this text to a vector (called vectorization in NLP)

set of all unique words

Hi	Gautam	Neeraj	Jatin
1	0	0	0 \rightarrow Hi
0	1	0	0 \rightarrow Gautam
0	0	1	0 \rightarrow Neeraj
0	0	0	1 \rightarrow Jatin

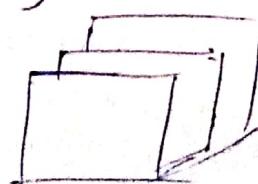
Now:- Hi Gautam $\rightarrow \left[[1, 0, 0, 0], [0, 1, 0, 0] \right]$ $\xrightarrow{\text{2-D Tensor}}$
 o: collection of 1-D Tensors

Similarly, Hi Neeraj $\rightarrow \left[[1, 0, 0, 0], [0, 0, 1, 0] \right]$ $\left[[1, 0, 0, 0], [0, 1, 0, 0] \right]$

Hi Jatin $\rightarrow \left[[1, 0, 0, 0], [0, 0, 0, 1] \right]$

\rightarrow ndim = 3 = # of Axis.

∴ Hi Gautam
 Hi Neeraj
 Hi Jatin \rightarrow



\rightarrow 3-D Tensor

$3 \times 2 \times 4 \rightarrow$ shape

Another example of 3-D Tensor is Time-Series Data : (financial + medical)

Kisi stock ka

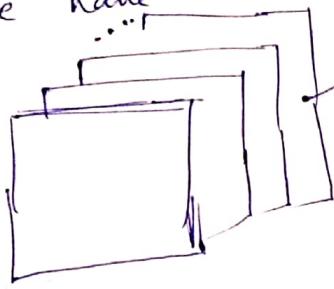
Highest | Lowest price
Every day for 365 days.

Aisa data jo ek frequent time period pe collect karte ho
Ex: Sharemarket Data

Days [- -] $\xrightarrow{\text{2-D Tensor, shape } (365, 2)}$
Days [- -] $\xrightarrow{\text{1-year}}$
Day 365 [- -]

Now:

say we have same of 10-years Data



3-D Tensor
shape = (10, 365, 2)

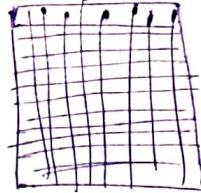
In Time series data
time axis is this is
Data is in 3-D Tensor

Examples of 4-D Tensors

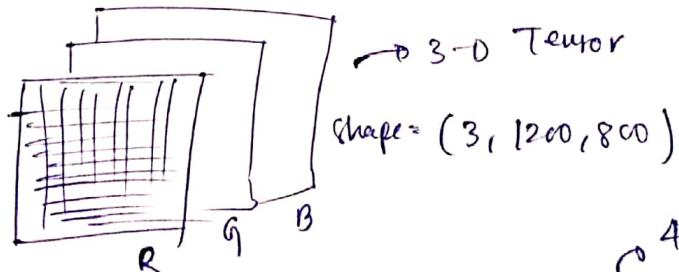
(Collection of image) data \rightarrow in computer vision Domain
Image Processing.

Any image is a collection of pixels:-
mei ek numerical value hota hai jiske
koi cheez dikhai deti hai

1 Colour Image:-



of han pi rof
wajah ki hume



what if we have 50 colour image:- shape = (50, 3, 1200, 800)
(Batches of images)

Examples of 50 Tensors:- Collection of Videos \rightarrow Collection

of images moving fast (moving faster than our periphery of vision)
& in videos we generally use 60fps (frames per sec)

1 video \rightarrow 60sec at 30fps, 980p (480x720)
1800 images \rightarrow (1800, 3, 480, 720)

60sec \rightarrow diff images ko hi
Resolution \rightarrow karsakta
hai \rightarrow each image has 3 shades
of each shade

Say we have 4 video clips \rightarrow (4, 1800, 3, 480, 720) \rightarrow 5-D Tensor

Framing the problem:-

1. Business problem to ML problem

How to increase the revenue of Netflix

→ Bring more new customers
→ charge more from existing customers
→ jo chor raha hai Netflix unko
✓ 80K le.

Churn rate :- jitne thi apne platform ke given period (monthly/yearly) ke jaa raha hai.

Means churn rate of users has come ↓ if it is a platform chor

(Mostly New customers ka aane ka rate is more than churn rate & then means grow)

Say Churn rate of Netflix 4%.

Hence Increase Revenue \Rightarrow Churn rate 4% \rightarrow 3.75%

2. Type of Problem (Supervised / unsupervised / ...)

Our end goal is to use the churn rate - and for that we need to know who are the users who will be leaving the platform and then we need to stop them.
Problem is user will leave the platform Y/N → supervised classification problem.
(by offering instant discount on next month subscription)

Again:- Kuch log aise honge jisko platform chome ka man jyada hogा & kuch log aise honge jisko lag raha hogा ki chor dete hai (kam pissed off honge) → Hence Sabhi ko banaban discount kyu dena

∴ Kam man hai chor ke jaane ka \rightarrow kam discount do
jyada " " " " " " " " → jyada " " "

∴ Now we want to find the amount with which they want to leave the platform (i.e. prob of guy leaving the platform)

Now it becomes a supervised-
Regression problem.

3. Current Solution :-

Say there is already a model developed by some team that predicts the churn rate of next month and this model can help us → we can go & talk to that team regarding the factors they considered for predicting the churn rate.

~~for~~
9.

Getting Data:-

if we want to know prob of a guy leaving the platform then we need data & we should think clearly what type of data we need?

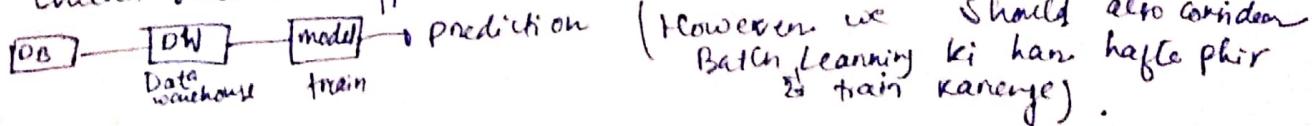
- 1/ Watch time → Agar wo browsing mei jyada time lagao & acha hai ~~apne man parand ke cheese nahi mil rahi~~
- 2/ Search but did not find (machine mei kitne searches kiye & nahi mil)
- 3/ Content left in the middle → means content is not much tempting
- 4/ clicked on recommendation (order of recommendation)

Now Data Engineer Netflix ke Database mei $\frac{1}{2}$
ek Datawarehouse banaya jisme ye saane col's honge

5/ Metrics to Measure: → jo hum kar rahe hai wo se kahi hai ki nahi
kaise paile karoge ki heenue jo model banaya hai wo successful
hai ki nahi → chunne jitna churn rate predict kiya &
jithe log chor kar gaye isse diff behot kam hai
then model se kahi hai.
→ jisko predict kiya chor kar jaiga wohi gyse
mean model se kahi hai.

6/ Online vs Batch :-

Online will be better sinle se hot saane factors
churn rate ko affect karle hai & wo upar niche hota rehta hai.

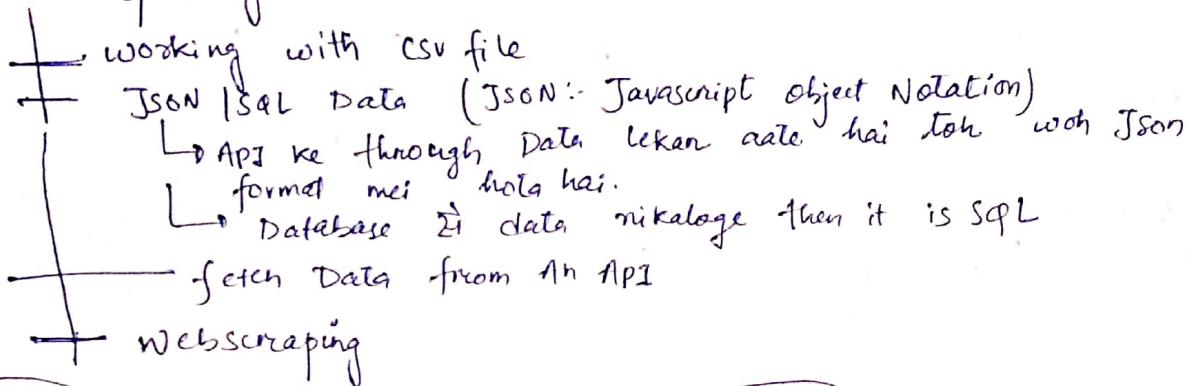


7/ Check Assumptions:-

cols features decided → are they available
will the model work same for us people and Indian people (Geographic based changes in model)

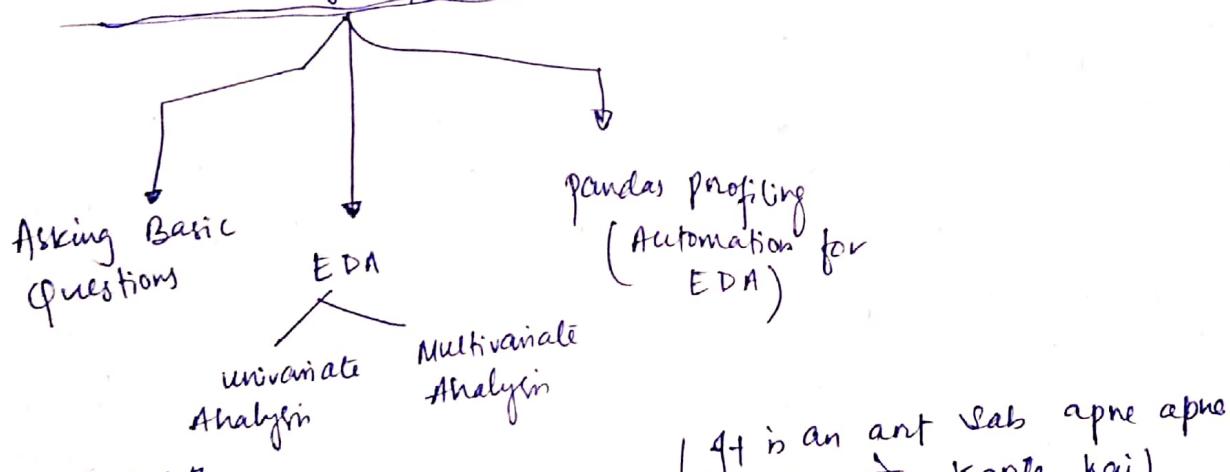
2/ Data Gathering:-

Note: "Bad Algo with huge data will work better than Good Algo with less data".



Laptop → folder Machine Learning on Desktop → Jupyter Notebook.

3/ understanding your Data:-

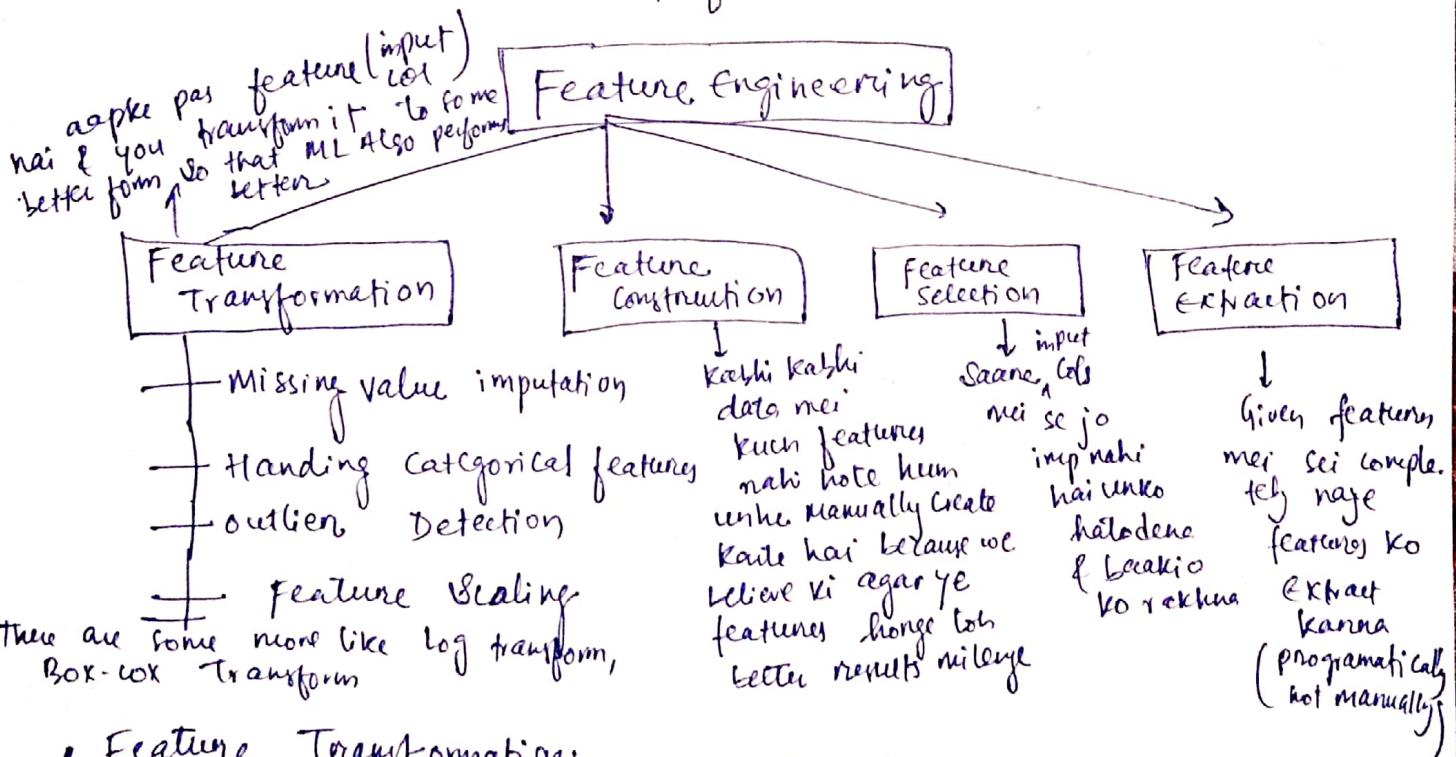
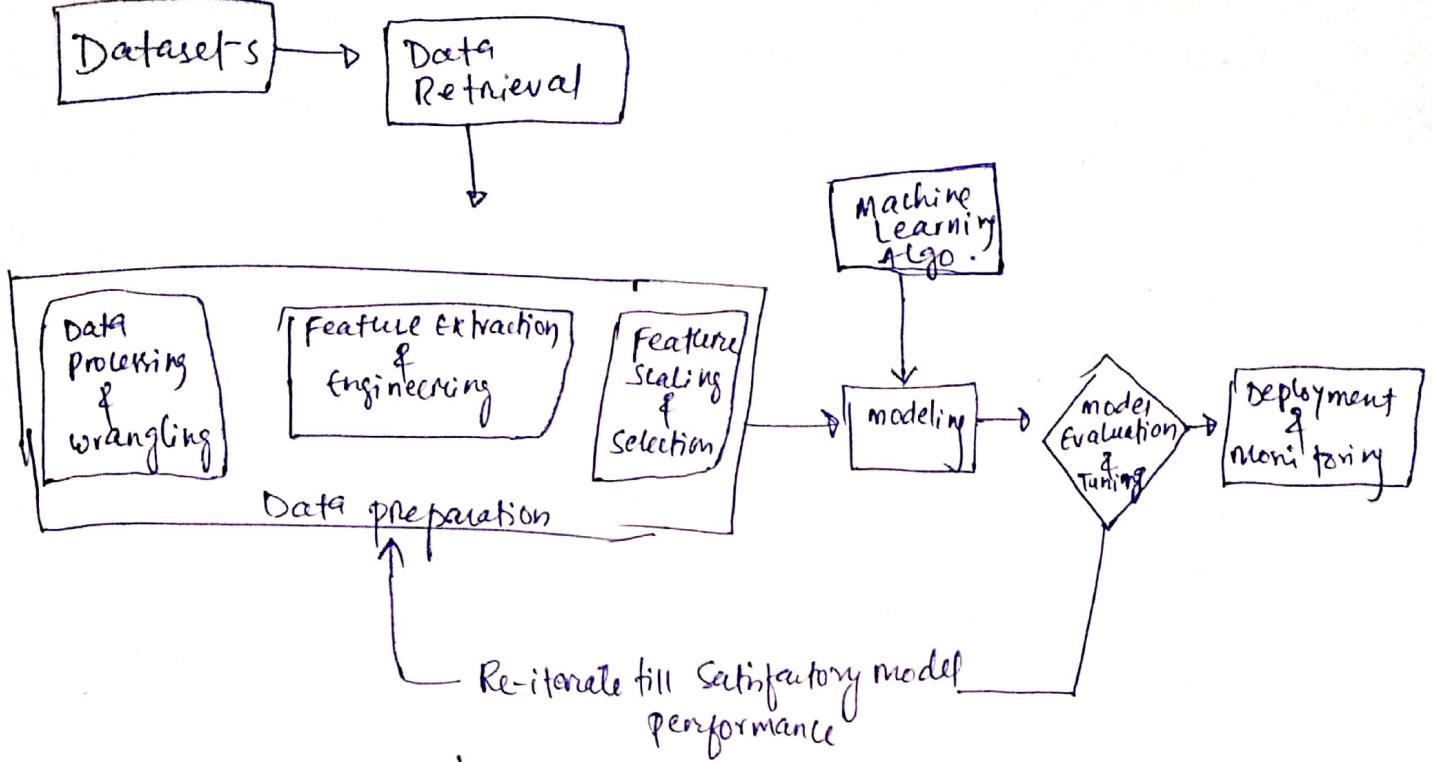


4/ Feature engineering: (gt is an ant lab apne apne tanike & karle kai)

gt is the process of using Domain knowledge to extract features from raw data. These features can be used to improve the performance of ML algorithms.

Bad Algo + Good feature selection → Better performance than Good Algo + bad features selection.

P.T.O.



Feature Transformation:-

→ missing value imputation

ID	city	Age	married
1	Lisbon	25	0
2	Berlin	25	1
3	Lisbon	30	1
4	Lisbon	30	1
5	Berlin	18	0
6	Lisbon	NAN	0
7	Berlin	30	1
8	Berlin	NAN	0
9	Berlin	25	1
10	Madrid	25	1

(in Real life Data some values may be missing due to some reason but this is a problem since our main library sci-kit learn doesn't accept missing values while training the model ∴ Before Training we have to fill those missing values or remove those rows.

→ Av age = 26 *stha NAN mei 26 denge*

• kabhi kabhi bahot kam values missing hoti hai then hum sawal remove kar dete hai ∵ farak nahi padega toh.

Kashii kashii bahot jyada Data missing hota hai Page 18.

So we need to fill → we can replace missing values by Mean, Median in case of Numerical feature & by Mode (most frequent category) in case of categorical Data.

There are also other ways to fill NaN.

→ Handling Categorical values

Index	Animal	Index	Dog	Cat	Sheep	Horse	Lion
0	Dog	0	1	0	0	0	0
1	Cat	1	0	1	0	0	0
2	Sheep	2	0	0	1	0	0
3	Horse	3	0	0	0	0	0
4	Lion	4	0	0	0	1	0

Categorical values

⇒ then 10000 represents a dog... and soon.

Problem is:- Scikit-Learn by Numerical values pe kaam karta hai.

Sometimes we convert Numerical values to categorical:

Like:- Age (Numerical)

0 - 12

Child

13 - 19

Teenage

20 - 35

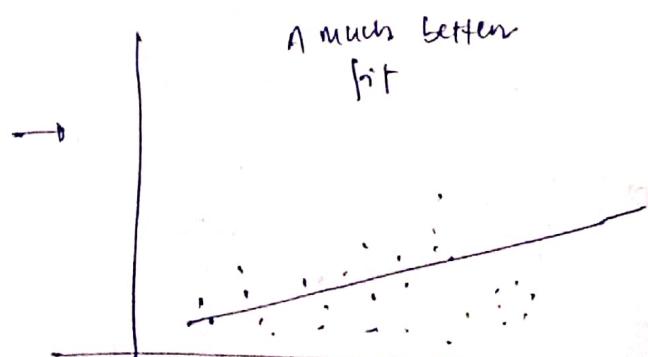
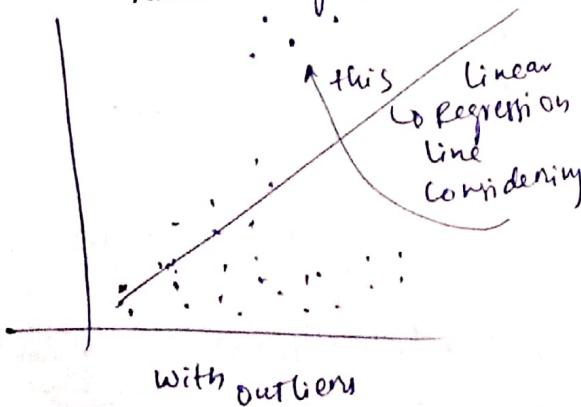
Adult

This is called
Binning.

and so on.

→ Outlier Detection.

Linear regression line is a line that is closest to all points.



ML Algos that are highly affected by outliers & hence it belongs imp to remove outliers but before that we need to detect outliers.

There are some outliers of hence it belongs imp to remove outliers but before that we need to detect outliers.

Feature Scaling → It is the last step we do in the feature tag pipeline. Sometimes the input cols / features have very different scales ex: Age | Salary
 44 | 72000
 27 | 48000
 30 | 51000
 38 | 61000
 40 | 93000

Now, if we are working with an ML Algo (like KNN) that calculates distance b/w two points $(44, 72000)$ & $(27, 40000)$ then $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ → then the salary term will be dominant in the distance calculation & hence age will not be able to control / monitor the ML behaviour.

So generally we scale the features i.e. features ko ek given range mei le aate hai -1 to 1 (min max scaling, standardization, normalization, min absolute scaling)

• Feature construction

completely ek naye feature (input col) create karne (based on intuition, Domain knowledge)
 In the Titanic Dataset:

Sibsp → siblings / spouse
 Parch → parent / child → family
 we can actually add them up & create a new col
 family → we can even convert family to categorical feature
 family categorical/feature
 0 → alone
 0-1 → small family
 4+ → Large family

Some techniques are:
 Feature splitting
 Feature Grouping.

• Feature Selection.

Bachat karne input cols hai & ap important input cols ko select karte hain.

MNIST data set → Deep Learning (Convolutional Neural Networks)

↳ 50000 images of handwritten digits.
 & each image is 28×28 resolution = 784 pixels
 most pixels are white & the black/grey pixels at digit dikhta hai
 This is converted to tabular form.

each row is an image.

label	pixel 0	pixel 1	pixel 2	...	pixel 784
0	1	0	0	.	0
1	0	0	0	.	0
2	1	0	0	.	0
3	4	0	0	.	0

∴ each pixel is a feature (784 features)
 This is huge... But what if we consider only those pixels that are important & remove the others.
 (those that has the digit)

using this the speed & perf of ML Algo increases

some techniques are:- Backward Selection, Forward Elimination,
 many input col.

→ Feature extraction → mostly when we are dealing with high dimensional data etc.
 ex: house pricing → this is not feature extraction completely new
 ∵ Yaha tk features create hua.

# of rooms	# of Bathrooms	price	suppose kisi ek input col	sq. feet area	Price
:	:	:	Kohatana	:	:
			hi hai j		
			Ek hi col se khus		
			hai		

famous feature extraction

Technique :- PCA, LDA, TSNE

Let's say we have 5 features & we create 5 new features by feature extraction then out of these new 5 we use the important say 2 features & remove the remaining 3 less important features

Let's start

→ It is a technique to standardize the independent (input) features present in the data in a fixed range.

Feature Scaling: → It is the last thing we do in the feature engg. pipeline. ML model ko data dene ke jaisa pehle we do feature scaling.

Standardization:

Why do we need feature scaling → already discussed with

age	salary	→ Suppose we use KNN Algo (It uses distance)
50	83000	(50, 83000)
27	47000	(27, 47000)

$$\text{dist} = \sqrt{(y_2 - y_1)^2} = \sqrt{1225000000}$$

then salary factor is dominating the KNN Algo & age does not seem to have much effect on the Algo.

- Agar humare features ka scale accha nahi hogi then hamare kuch ML Algo ka logic hi aisa hai denge ki wo accha perform nahi karenge.

Types of feature Scaling

Standardization

Normalization

+ min max scalar

+ Robust scalers (works good on outliers)

etc.

Standardization is also called Z-score Normalization

Say we have:

Age	Salary
27	
15	
33	
63	
90	
05	
:	
500 Values	

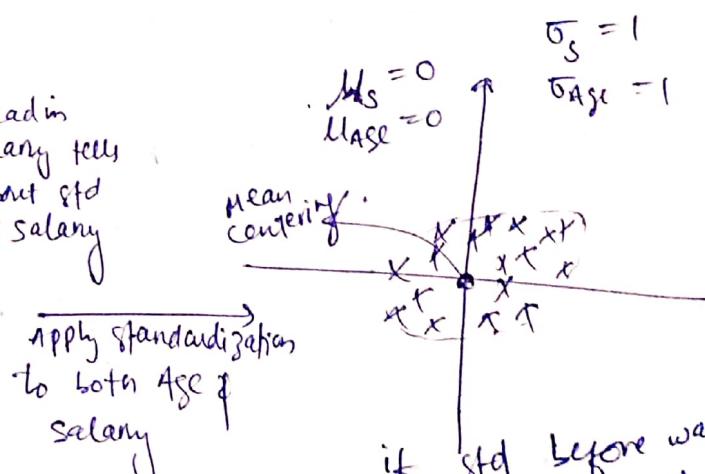
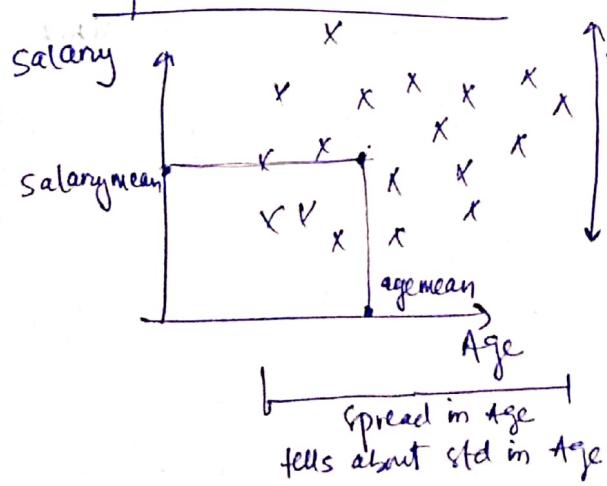
Say we want to standardize Age.
then $(x) \text{Age} \rightarrow \frac{\text{Age}'_i}{\sigma_i}$

$$x'_i = \frac{x_i - \bar{x}}{\sigma}$$

$\bar{x} \rightarrow \text{mean of age col}$
 $\sigma \rightarrow \text{std of age col}$

Note the After standardization
mean of $\text{Age}' = 0$
std of $\text{Age}' = 1$.

Geometric Intuition:



$\bar{x}_S = 1$
 $\bar{x}_{\text{Age}} = 1$
 $s_S = 0$
 $s_{\text{Age}} = 0$
if std before was > 1
then the data was
squeezed & if std
before was < 1 then
data is spread to
make std = 1.
→ 1. standardization.

See Machine Learning on desktop

→ (Supper Note book)

Feature Engineering → 1. Feature scaling

→ 1. standardization.

When to use standardization?

Page 20

Algorithm(s)

1/ K-Means

2/ K-Nearest Neighbours

3/ Principal Component Analysis
(PCAs)

4/ Artificial Neural Network

5/ Gradient Descent

Reason of applying feature scaling:-

uses the Euclidean distance measure

Measures the distances b/w pairs of samples & these distances are influenced by the measurement units

Tries to get the features with the maximum variance

Apply Gradient Descent

The theta calculation becomes faster after feature scaling and the learning rate in the updated equation of Stochastic Gradient Descent is the same for every parameter.

Algorithms where we don't do standardization:-

Decision Tree, Random forest, Gradient Boost,
XG Boost

Jimplies standardization ka koi fikr nahi parega.

Normalization:

It is a technique often applied as a part of data preparation for machine learning.

The goal of Normalization is to change the values of the numeric cols in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

Dealing with Numerical lots \rightarrow it has Numerical value, magnitude & units
eliminate the units & keep the magnitude - that too on a common scale.

Normalization Technique

- Min Max Scaling (This is often done)
- Mean normalization 90% times
- Max absolute
- Robust Scaling.

Min max scaling

Weight (X_i 's)

130

67

81

61

32

59

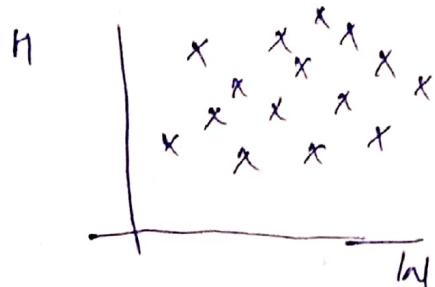
Normalization
Min-Max Scaling

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \rightarrow \text{all pt} \rightarrow \text{Smallest value 0}$$

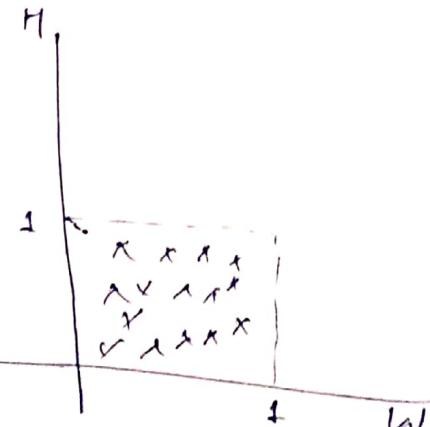
$x_{\max} - x_{\min} \rightarrow \text{Range} := [0, 1]$

Geometric Intuition

weight vs height



Min max scaling



* work - Machine learning - feature engineering - feature scaling - Normalization.

Mean Normalization

wt (X_i 's)

73

86

59

31

Mean Normalization

$$x'_i = \frac{x_i - \bar{x}}{x_{\max} - x_{\min}} \rightarrow \text{Mean Centering.}$$

Range $-1 \rightarrow 1$

Sci-kit Learn has no class for this we will have to write its code if we want to use it.

This is used in those algs where we need centered data \rightarrow But we often use standardization.

— Max Absolute Scaling

Page 21.

Here formula :- $x'_i = \frac{x_i}{|x_{\max}|}$ → class in scikit-learn called MaxAbsScaler

It is used when we have sparse data (Having a lot of zeros)

— Robust Scaling

Here formula :- $x'_i = \frac{x_i - \bar{x}_{\text{median}}}{IQR}$

class in Scikit Learn :- RobustScaler.

It is used when

+ Data has outliers (it generally performs good with outliers)

$$IQR = \text{inter quantile Range} = Q_3 - Q_1$$

$$= 75\% \text{ value} - 25\% \text{ value}$$

Standardization v/s Normalization

fit on: Is feature scaling reqd? (check kaise algos mei reqd. hai & kaise mei nahi)

Maximum times Standardization will perform better

but sometimes when we already know the min & the max of an input dataset then we perform MinMax Normalization.

Handling Categorical Values/Algo:-

vramp

Encoding Categorical Data

Types of categorical data

ML algs expect numbers & categorical Data are in string format ∴ It's but duty to convert them to Numbers.

Techniques

There is also Label Encoding
Similar to Ordinal Encoding

Ordinal Encoding (on ordinal data)

One hot Encoding (on nominal data)

Nominal

Ordinal

There is no order (comparison) b/w the data

Ex. Estates

we cannot say

up is better than mp

(II) Gender

we cannot say Male is better than Female

here we have order

Ex. Distinctions (Ranks in class)

(Excellent, Good, Bad)

(II) Reviews

(Excellent, Good, Bad)

ordinal v/s Label Encoding

Suppose we have data set

x	y

X → input (cols)

Y → output

* if X has an ordinal categorical col & Y is Numerical → ordinal Encoding

* if X has an ordinal categorical col or has not (not numerical)
& Y is categorical → Label Encoding on Y

Like: email-spam classifier & X will be
placement hogya nahi etc. & Y will be
ordinal Encoding

Ordinal Encoding

Suppose we have an input col

Education	→ Education
High School	HS → 0
under Graduate	UG → 1
under Gr...	PG → 2
Post Grad...	PG → 2
Post Grad...	PG → 2
High School...	HS → 0
Post Grad...	PG → 2
High School	HS → 0
under Grad...	UG → 1

PG > UG > HS

∴ PG ko Sabse jada value do
UG ko use karo
& HS ko Sabse karo

Jupyter Note book.

One Hot Encoding :- (for Nominal)

Consider a Nominal Categorical Data i.e. One-Hot Encoding

Color	Target	Color-Y	Color-B	Color-R	Target
Yellow	0	1	0	0	0
Yellow	1	1	0	0	1
Blue	1	0	1	0	1
Yellow	1	1	0	0	1
Red	1	0	0	1	1
Yellow	0	1	0	0	0
Red	1	0	0	1	1
Red	0	0	0	1	0
Yellow	0	0	0	0	0
Blue	0	0	1	0	1

in the color (Nominal Categorical col) we have Page 22.

Three Categories [Y, B, R] then in One-hot encoding
we make col for each category

Now first data is yellow \rightarrow Y B R

2nd Data is yellow \rightarrow 1 0 0

3rd Data is blue \rightarrow 0 1 0

and so on.

$\therefore [1, 0, 0] \rightarrow Y$
 $[0, 1, 0] \rightarrow B$
 $[0, 0, 1] \rightarrow R$

\therefore Humne
String ko
vector mai
convert kar diya.

Q:- what if we have 50 diff categories?

A:- Then we make 50 cols.

Dummy Variable Trap (jo cols hain in one hot encoding aur called dummy variables)

What we do is after one-hot encoding we drop the 1st col (Generally, kisi bhi col ko kar sakte hai) \because we drop the color-y col. If there are n categories in a nominal col then after one-hot encoding we drop the 1st & keep the n-1 cols. only.

This is due to multicollinearity.

If we don't drop then $\sum \text{each row} = 1 \rightarrow$ 1 mathematical relation b/w the cols we don't want this \because they create problems for linear ML models (Regression models)

\therefore we drop 1st color-y: Now: $Y \rightarrow [0, 0]$ B gone
 $B \rightarrow [1, 0]$ $Y \rightarrow [1, 0, 0]$
 $R \rightarrow [0, 1]$ $B \rightarrow [0, 1, 0]$
 $R \rightarrow [0, 0, 1]$

Say Car Brand \rightarrow Nominal data

To say there are 40 brands in one hot encoding there will be 40 cols. (This is the way but this will make ML algo processing slow)

What we do in this case:-

The frequently occurring categories we keep & all the least occurring categories we make a new col for them as others.

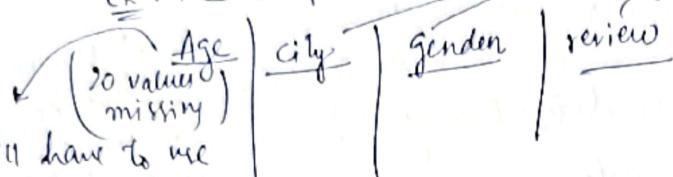
This is how we reduce the dimension of the dataset.

But this is used when kuch categories bahot jada & kuch kam aarahi hai.

Column Transformer:

Suppose we have input cols & in one colth gt is Ordinal, other is Nominal & other has missing values, then we cannot take the three cols as separate parts perform the operation
 ex: 100 data points/rows
 we'll have to use imputer to fill the missing values.

then combine them back
 Nominal: we'll use one-hot encoding
 Ordinal: we'll use ordinal encoding



Same cols pe alog alog transf. lagane hai hence agar alog karke kyo then 3 diff numpy arrays banenge & unko jodna padega → This is not efficient

∴ we use Column Transformer (Class in scikit-learn)

Jupyter Notebooks:

Scikit-Learn Pipelines:

Pipelines chains together multiple steps so that the output of each step is used as input to the next step.

Pipelines makes it easy to apply the same prepossessing to train & test.

Ex: we have an ML model & data → 1st we ~~handle~~ handle missing values then handle categorical data then say we train the model



Project

yaha jo series of steps hum training ke without time like hai woh pipeline series of steps humne production time pe likhe hai (which is not good)

with pipeline.

Jupyter Notebook

Mathematical Transformations:

Feature Eng \rightarrow Feature Transf. \rightarrow mathematical Transformations

Simple bhaaya mei:- hum apne col ke upar mathematical fn laga kar transf. kar sake hain into new col.

+ log Transf

+ Reciprocal Transf

+ power Transf (Power & sqrt. transform)

+ Boxcox Transf (very famous)

+ Yeo-Johnson Transf

+ we can also define a custom transf like $(x^2 + 2x)$

but after that we should set Normal dist of that col

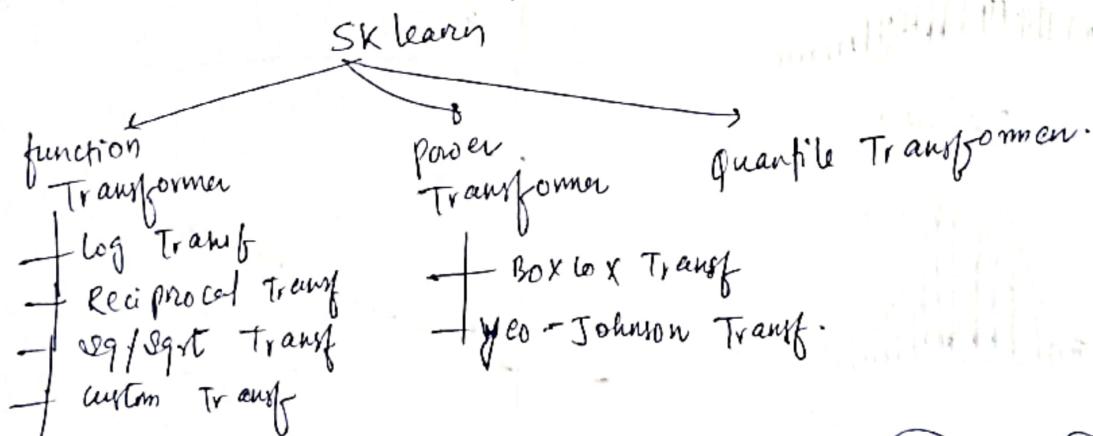
After these transformation the pdf (distribution) becoming

Normal dist is very imp in statistics (since we can predict a lot about the data then)

(Calculations kar sakte hain)

* Algos like linear Regression, Logistic Regression Assume that the data is Normal if not we make it.

Algos like Decision Tree, Random Forest ko farak nahi padta ki data ka dist Normal hai ya nahi.



How to find if data is Normal?

Akh. use Seaborn \rightarrow sns. distplot()

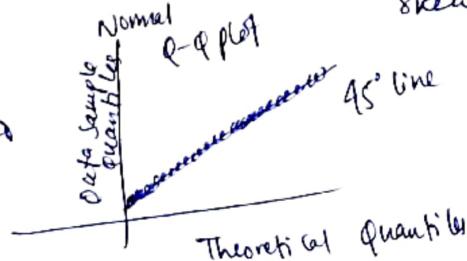
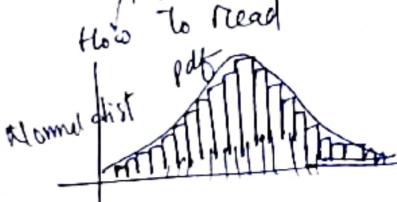
\rightarrow pandas \rightarrow pd. skew()

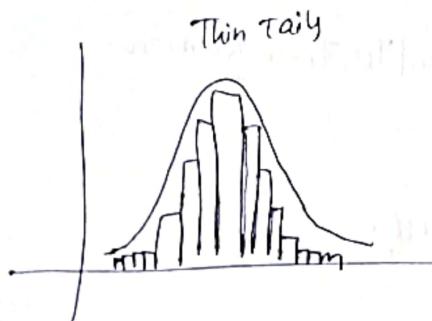
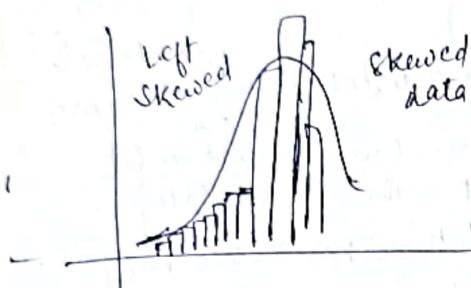
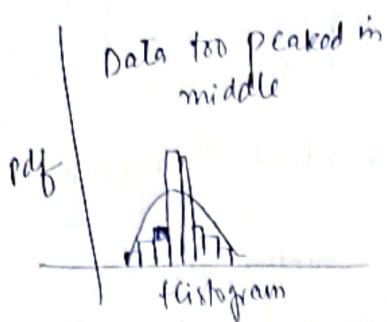


the ya the aage then skewed hai dist.

→ QQ Plot

How to read



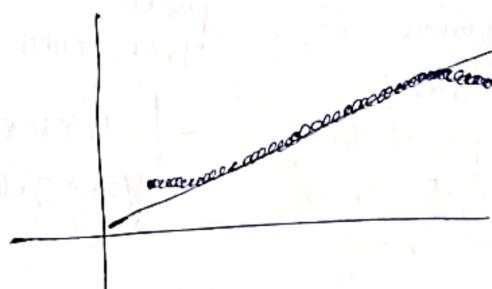
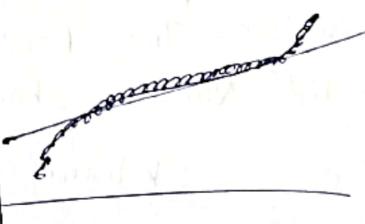
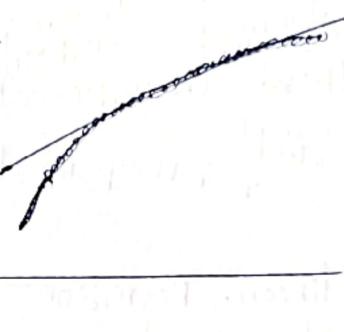
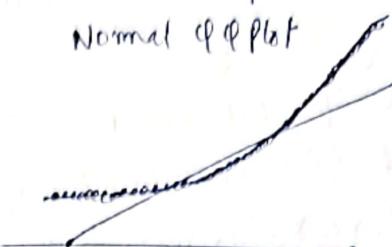


Normal Q-Q Plot



Theoretical quantiles

Normal Q-Q Plot



Hence. in qq dist jitna data jyader Normally dist hai & shag raha hai vetha

deep line kee repair ho uthe rap ka jitne adas line ZI qq plot den data kar Normally dist hai.

Log Transform:-

To apply log transform on a col \rightarrow we take each entry ka log (base 10 ya 2 ya e)

Leisse kphna wo column jo current state hai use jyada

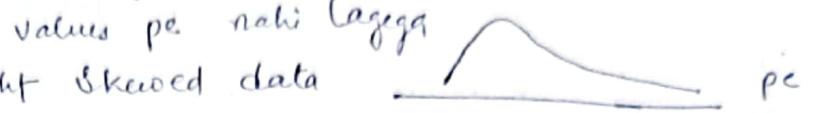
Normally dist. ho jata hai.

when to use:- +ve values pe nahi lagta

- Right Skewed data
 - log transf. lagane pe wo usko centre pe leke aajala hai.

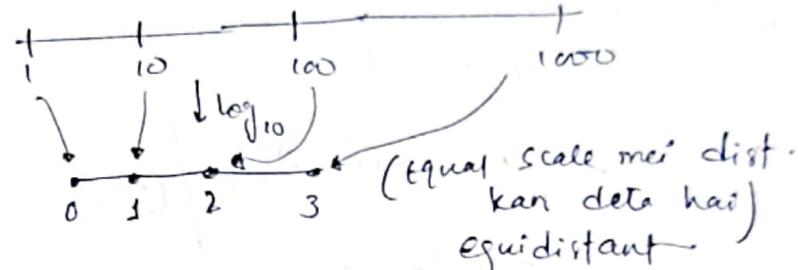
(basically say)

What log does



log transf. lagane pe wo usko centre pe leke aajala hai.

(basically say)



Reciprocal Transform ($\frac{1}{x}$)

Small transf	big values
Big transf	small values

Square Transform (x^2)

used for left skewed data then wo jo data hai wo pchle se jyada Normally dist hojata hai

Sqrt Transf. \sqrt{x}

Jupyter Notebook

Box-Cox Transformation :-

The function used here is :- $x_i^{(\lambda)} = \begin{cases} \frac{x_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(x_i) & \text{if } \lambda = 0 \end{cases}$

The exponent here is a variable called lambda (λ) that varies over the range -5 to 5. In the process of choosing, we examine all values of λ and finally we choose the optimal value resulting in the best approach to the normal dist for our variable (input col/feature).

in fact:- \log , $\sqrt{x_i}$, x_i^2 are special cases of Box-Cox Transform.

$$\lambda = 0 \quad \lambda = 1 \quad \lambda = 2$$

If note Box-Cox Transf. are strictly applicable to cols if $x_i > 0 \ \forall i$ (-ve bhi nahi chalega & x_i^0 bhi nahi chalega)

If this problem is solved by Yeo-Johnson Transf

Yeo-Johnson Transform:-

$$x_i^{(\lambda)} = \begin{cases} [(x_i + 1)^{\lambda} - 1] / \lambda & \text{if } \lambda \neq 0; x_i \geq 0 \\ \ln(x_i) + 1 & \text{if } \lambda = 0; x_i \geq 0 \\ - [(-x_i + 1)^{2-\lambda} - 1] / (2-\lambda) & \text{if } \lambda \neq 2; x_i < 0 \\ - \ln(-x_i + 1) & \text{if } \lambda = 2; x_i < 0 \end{cases}$$

This transformation is somewhat of an adjustment to the Box-Cox Transf, by which we can apply it to negative numbers. (i.e. now $x_i \leq 0$ allowed hai ab).

So if we have a col that is not Normal distributed & we are working with Algo (like linear, logistic regression, knn) that performs better with Normal dist. Then we should use these & see if they perform better.

Jupyter Notebook.

Concrete ka dataset.

Model :- To predict the Strength of concrete (Regression problem)

without Boxcox or
Yeo-Johnson with Boxcox or
Yeo-Johnson.

Encoding Numerical features:-

Page 25

Ex:- google playstore dataset

<u>Apps # of downloads</u>	<u>Bins</u>
23	100+
101023999	1000+
9927	100000+
11	1B+
101	1000000+
39	10000000+
2759862	
13405163	

is data &
we can
convert to
categorical
data

There are two famous Techniques for (Numerical \rightarrow categorical)

Discretization
(Binning)

Binarization

Discretization (binning)

Discretization is a process of transforming continuous variables into discrete variables by creating a set of contiguous intervals that span the range of the variable's values. Discretization is also called binning, when bin is an alternative term for interval.

Why use Discretization:-

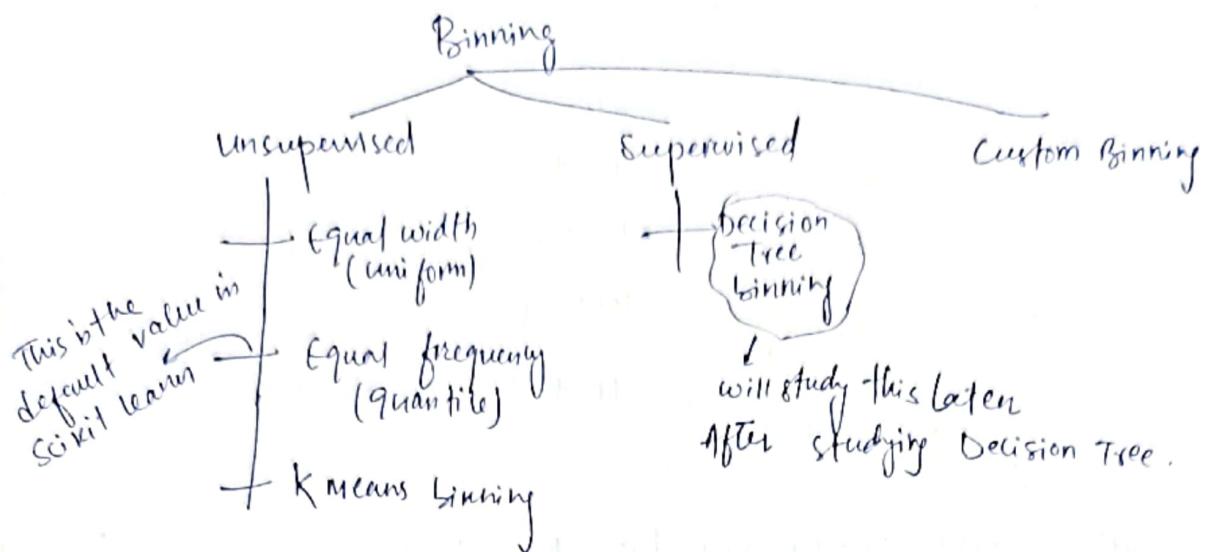
- 1/ To handle outliers
- 2/ To improve the value spread. (Data ke dist ko thora uniform karna)

Ex:- Age:

23, 1, 9, 41, 52, 37, 48, 51, ..., 82, ...
↓

0-10, 10-20, 20-30 (Just like histograms)
5 4 6

Types of Discretization :-



Equal width (uniform binning) :-

Age :- 27, 32, 84, 56, ...

first we give # of bins = 10 (say)

Now a formula is used = $\frac{\text{Max} - \text{Min}}{\# \text{ of bins}} = \frac{100 - 0}{10} (\text{say}) = 10 = \text{intervals length.}$

∴ the bins/intervals are :- $\frac{1^{\text{st bin}}}{0-10}, \frac{2^{\text{nd bin}}}{10-20}, \frac{3^{\text{rd bin}}}{20-30}, \dots, \frac{10^{\text{th bin}}}{90-100}$

and now put your values into these bins.

- outliers handle ho jate hai is wo last ya first bin mei
- And then we can plot like Histogram
- No change in spread of the data.

Equal frequency / quantile binning :-

Say intervals/bins = 10

Each ~~interval~~ interval contains 10% of total observation.

Intervals will be :-

0 - 10 percentile of data 10 percentile - 20 percentile ... 90 percentile - 100 percentile

Then these bins will contain 10% observations/data each.

• Say 0 - 16, 16 - 20, 20 - 22, 22 - 29, ... (bins are not of equal width)

isme 10 log hai yaha thi 10 log hai yaha & so on ...

but each bins hai same # of observations.

Quantile binning performs better than uniform binning because.

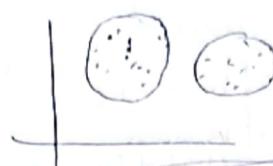
- handles outliers better
- Values spread ko uniform banata hai.

Page 26-

because.

K Mean Binning.

Here we use a clustering algorithm called kMeans.



These clusters are formed by kMeans in any dimensions.

This is used jab hamari data clusters mei ho.

pehla logo ke age phir thore dur ke baad kuch aur logo ka age

... and so on.

Here :- intervals ko centroid bolte hai

Age: $\dots \rightarrow \dots \rightarrow \dots \rightarrow \dots \rightarrow \dots$

say # of bins = 5 then 5 arrows kahi thi randomly place kando

$\dots \rightarrow \dots \rightarrow \dots \rightarrow \dots \rightarrow \dots$



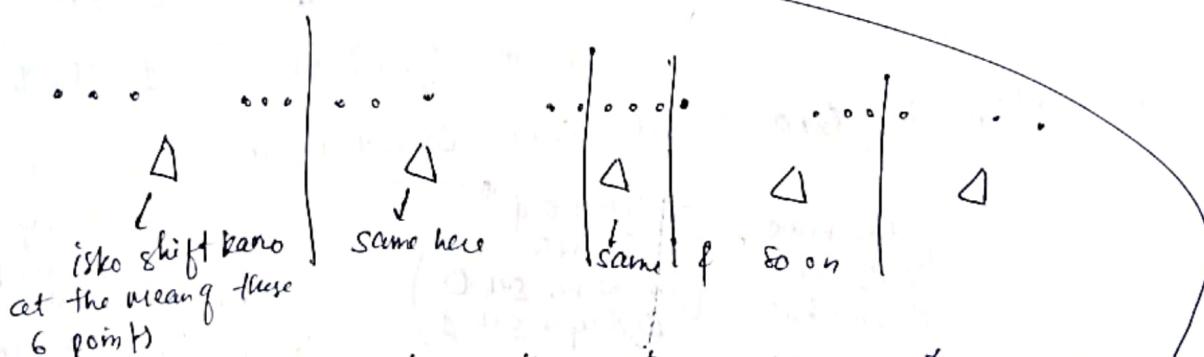
then left wall point left arrow nei jis right wall point, right arrow nei.

Now there are two techniques,

✓ us any one we get
say

→ then do centroids/arrows ka Bisecton lelo

→ ~~EK~~ point ka har wo point jis arrow ke sabse kam hoga wo point usme jaiga.



after this repeat the steps from step 2 is

And Aisa fab-tak karnege jab tak improvement in shift hona band na hoga.

And after that we get the values of the amns/centroids of that will finally give us the bins.
How we use these bins?

sklearn

class → KBinsDiscretizer()

bins=?

strategy

+ uniform

+ quantile

+ K-Means

matlab bin no. of
datas, konca data
vis bin mei ya
iska rep. kaise
hoga.
encoding
+ ordinal
+ ~~nominal~~
One hot Encoding (if Nominal)

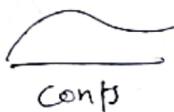
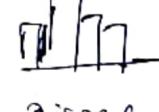
Jupyter NoteBook.

* custom | Domain Based Binning

We can make our own bins if we have the domain knowledge like :-
[0-18], [18-60], [60-80] & soon.

But for these we have to write khudka logic using pandas, no such thing in Scikit Learn.

Binarization → Sometimes it is useful like image processing.

in Discretization:- we convert  to 

in Binarization we convert conts data to binary (0 or 1).

Ex:- Annual

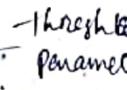
Income Binarization income < 6Lakh  Taxable zone

0 → no need to tax

income > 6Lakh

1 (has to tax)

There is a class in ~~Scikit~~ Scikit Learn called Binarizer()

by default '0' hota hai
is ka value 
parameter (is ka niche Sab 0)
And upper Sab 1

say salary col pe
Binarizer laga rhe hai =
if copy = True then ek naya
col create karega & if
copy = False then ~~ek~~ cui
salary col mei hi changes
karega.



Handling Mixed variables & Datetime columns:

Handling Mixed variable:-

In sometimes in our ^{single} column we have both ~~numerical~~ & categorical values (mixed)

Ex: (1) in Titanic Data col:-

Cabin.

B5

C23

D41

⋮

B 5
C 23
D 41
↓
Categorical Numerical.

These are not categories since they there will be many categories

B5

C 23

D 41

⋮

we split this cat

	Num
B	5
C	23
D	41

Ex (2) In a single col we have:-

Again we split into two:-

cat	num
NAN	7
NAN	3
NAN	4
A	NAN
C	NAN
NAN	4
D	NAN

7
3
1
A
C
4
D

Jupyter Notebook.

Handling Date Time based col:-

Date :-

23 Aug

Date
(23)

Month

year

day of week

quarter of year

which
season

ye date weekend
ya ye date weekdays hai

time:-

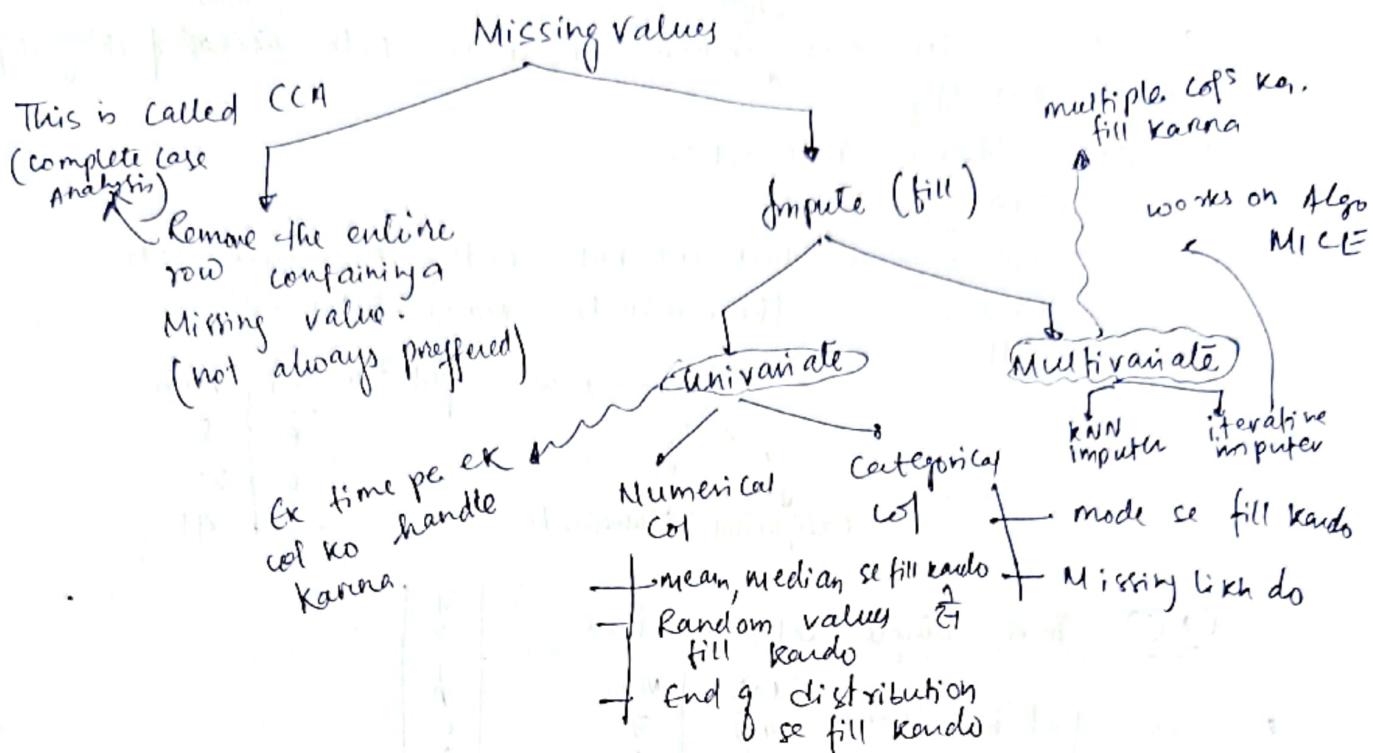
08 : 45 : 03

hrs mins secs.

Jupyter Notebook.

Handling Missing Values

ML algos are not very good with missing values, hence we should handle them before training our model.



Complete Case Analysis (CCA)

Complete Case Analysis (CCA) also called 'list wise deletion' of cases, consists in discarding observations^(row) where values in any of the variables^(cols) are missing.

CCA means literally analyzing only those observations for which there is information in all of the variables in the dataset.

Assumptions for CCA

- Data is missing completely at Random. (MCAR)
↳ Aisa nahi hona chahiye ki survey ke so values missing hai. (Tab hum CCA ya last nahi kar sakte)

Adv / Disadv. of CCAPage 28Adv.

- easy to implement as no data manipulation reqd.
- preserves variable dist (if data is MCAR, then the dist of the variables^(cols) of the reduced dataset should match the dist in the original dataset)

Disadv.

- It can exclude a large fraction of the original dataset (if missing data is abundant)
- Excluded observations could be informative for the analysis (if data is not missing at random)
- When using our models in production, the model will not know how to handle missing data.
↳ hence CCA is not used much.

When to use CCA?

- MCAR (Data missing Completely at Random)
- 5% data missing ho.
(ya kam)

Jupyter Notebook.Imparting (filling) missing values.Handling Missing Numerical DataImputing Numerical DataUnivariate Imputation

say col2 mei kuch value missing hai f center fill karne ke liye agar hum us col2 ke baaki date ko hi use kare then it is called univ...

Multivariate Imputation

say col2 mei ek value missing hai & isko fill karne ke liye col2 ke saath baaki kuch cols ka bhi help le rakte hai then it is called multivariate imputation

UNIVARIATE IMPOTATION

missing values ke jagah

- + mean(median) \rightarrow fill kar dena
 - + Arbitrary fill kar dena
 - + End of dist.
 - + Random values fill kar dena.
- How to select automatically
the best one using
scikit Learn?
- All :- Jupyter Notebook

— Mean / median \rightarrow imputation \rightarrow very obvious.

if dist is
kind of Normal then
fill with mean

if dist is
skewed
then fill with
median.

Advantages:-

- Easy to use

All these should not
happen.

Disadv:

- changes the dist shape
- outliers aajate hai.
- ~~Jis col mei impute kya hoga~~
relationship with other cols mei
changes aajate hai.
This is called changes in
covariance/correlation.

When to use:-

- (i) Data is MCAR
(Missing Completely at Random)
- (ii) missing data $< 5\%$.

Jupyter Notebook

Arbitrary value imputation :- (This is not used much)

\hookrightarrow mostly this technique is used for categorical variables

\hookrightarrow here we replace the NaN with the word Missing,
so that the ML also could differentiate b/w
Missing data & present data.

\hookrightarrow however, we can use this technique for numerical
data/variables also.

we then replace the missing values with an
arbitrary Number like -1, 0, 99 etc. (depends on oneself)
idea is :- we are creating a difference b/w the existing data
and the missing values.

Adv:

- easy to apply

disad.

- pdf becomes distorted
- variance, covariance changes.

\Rightarrow we use this technique when Data is not missing at Random.

— End of Distribution Imputation

\hookrightarrow It is an extension of Arbitrary value imputation.

\hookrightarrow Arbitrary value ^(value) _{shundha} thorak difficult hota hai

\hookrightarrow yaha we choose a value from the end of dist.

for Normal dist the arbitrary value = Mean + 3 σ or Mean - 3 σ

if Dist is skewed, we use IQR $(\sigma = \text{std})$

Calculate the arbitrary value :- Proximity Rule to

arbitrary value = $Q_1 - 1.5 \text{IQR}$

$(\text{IQR} = Q_3 - Q_1)$

or arbitrary = $Q_3 + 1.5 \text{IQR}$

adv:

- easy to apply
(same method)
as above

disad:

- pdf & variance change ho jaoge

covariance.

When to use?

\rightarrow When Data is not missing at Random.

Random Sample Imputation

\hookrightarrow It can be used on both Numerical & Categorical data.

& how to select the best Technique for imputation

Age
=
=
=
NaN
=
NaN
=
=

then the missing values are filled by random Numbers & the random numbers belong to the Non missing values in the col. i.e. col ke data mei zt hi randomly ek number select karke fill karega.

advantage:

- It is easy to apply. almost
- Data dist & variance remains intact does not get disturbed

Note: This technique cannot be performed using scikit-learn, we'll have to use pandas.

- This technique works best with Linear ML models (Linear regression, Logistic regression) etc. regardless of the % of NaN.
- It does not work good on Decision Tree based Algos.
- However the covariance of the col with other cols gets disturbed.
- This technique memory heavy for deployment, as we need to store the original training set to extract values from & replace the NaN in the coming observations.
ie. X-train age mei sei ek random value delect kare phir use incoming data mei agar age missing ho toh fill kar denge.

Jupyter Notebook:

Missing Indicator

In this for every col that has missing value we create a new col. & in new col mei do hi values hongi → True & false
for missing value for value not missing.

Age	Fare	Age-NA
21	23	F
32	10	F
71	9	F
NAN	78	T
38	100	F

→ isse ML model Missing & nonmissing data mei diff create karna sikha jata hai. and hence uska perf improve ho jata hai.

How can we automatically select the best parameter values for any imputation technique. Page 30.

Here we use a technique called Grid Search CV.

Le yaha scikit-learn sare combinacions try karta hai and best result de data hai.

Jupyter Notebook.

Handling Missing Categorical Data

- + most frequent values \Rightarrow replace kardo
- + NaN ke jagah Missing likh do (Missing Category Imputation)
- + Random value filling (already done)

Most frequent technique

- * MCAR (Missing completely at Random)
and missing data $< 5\%$.
And one more imp thing
is the mode should appear
in the data much more than
the remaining categories.

\rightarrow Replace with
mode of the
col data.

jo Sabse jyada bar
appear hua hai.

- adv:- easy to use
- disad:- Data dist is changed.

Missing Category Imputation

Say we have 10% ya use jyada values missing hai
then we make a new category Missing & jaha shi.
NaN hai we replace it with Missing & hence the
no. of categories in the col increases by 1.

\downarrow
This tells the ML algo that the Missing
Category data were missing in the col.

\downarrow
in numerical data the same technique was
called Arbitrary imputation technique.

MULTIVARIATE IMPUTATION TECHNIQUE

KNN imputer Iterative Imputer

KNN Imputer

It works on algo called KNN (apne neighbours ki tanah)

Logan kisi row mei koi value missing hui toh woh us row ke value fill hoga jo sabse jyada similar hai row to be filled sei.

similarity is calculated using Euclidean distance.

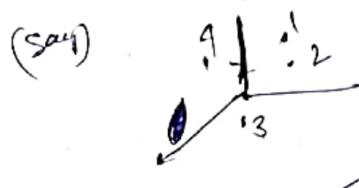
Ex. S-N	Feature1	Feature2	Feature3	Feature4
1	33	-	67	21
2	-	45	68	12
3	23	51	71	18
4	40	-	81	-
5	35	60	79	-

wo point jo dono mei present hai in X & Y

$\text{dist}(x, y) = \sqrt{\text{weight} * \text{sq. of distance from present coord.}}$

where weight = $\frac{\text{Total } \# \text{ g coord}}{\# \text{ g present coord.}}$

This can be represented in a 2-D coord system



this wke say row 2 mei fill do if & karna chalte hai we take

then we find dist (using this) of 2 with all other pts & jo shi row point

2 ke sabse jyada paas/close hoga uske feature 1 ke value 2 mei replace kar diya jaiga.

say agar q fill kare then $\text{dist}(4, y) \rightarrow$ jo sabse close uske feature 2 & 4th row mei fill karega.

In KNN, $k \rightarrow \# \text{ of neighbours}$ we want to start with say $k=1$ then sabse closest neighbour ka feature 1 ka value & replace karega row 2 mei. If $k=2$, then two nearest neighbours from point / row 2 ka feature 1 ka mean & fill karega in row 2 and so on.

Calculating Nan distance

then leaving feature 1 we take F_2, F_3, F_9 for distance

$$\text{dist}(2,1) = \sqrt{(67-68)^2 + (21-12)^2} \times \text{weight}$$

Total pairs = 3
pairs possible = 2

	F_1	F_2	F_3	F_9
Row 1	33	67	21	
Row 2	45	68	12	
Row 3	23	51	71	18

$$\text{dist}(2,3) = \sqrt{\text{weight} \times (45-51)^2 + (68-71)^2 + (12-18)^2}$$

$\frac{\text{# of total pairs } 3}{\text{# of possible pairs } 3} = 3/3$

say we calculate like above dist of point 2 from others:

$$d(2,1) = 11.29, d(2,3) = 9$$

$$d(2,4) = 5.19, d(2,5) = 14.79$$

Now if $k=1$, then q^{th} pt./row is closest to point 2 \therefore

feature 1 or row 1 i.e. 40 will be filled at row 2

feature 1 then 2 nearest neighbours are row 3 & row 4

if $k=2$: features of 3 + features of 4 = $\frac{23+40}{2} = 31.5$ will

be filled at feature 1 of row 2 & so on.

Adv:-

- This is more accurate
- Good technique for small or medium data sets.

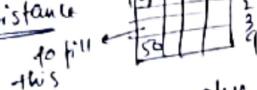
Disad:-

- More number of calculations especially on big datasets
- On production stage we need to deploy the whole training set on the server. Since testing values are calculated based on training data. (Hence memory usage is more)

Jupyter Note Book

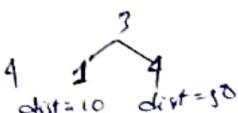
There is a parameter called weights = uniform or distance (Default) what we calculating above to fill. $k=2$ say nearest neighbours are 1 & 4

uniform is the normal weight = distance



then value to be filled at row 3 = $\frac{1}{10} \times 70 + \frac{1}{50} \times 50 = 9$

Issue: jo jyada close hai uska weightage jyada hoga.



Multivariate imputation by chained equations for Missing value (MICE algo) | Iterative Imputation

MICE stands for Multivariate imputation by chained eq's.

Assumptions for using MICE

- Data is MAR (missing at Random)
in this baaki col ko use karke missing col ka data predict karne sake hai

Adv: • It is quite accurate

Disadv: • It is slow ∵ we are using an ML Algo to predict a missing value
• Training dataset ko memory pe rakhna hota hai (memory heavy)

How it works: (MICE is applied on Input cols)

1) Actual Dataset (Startup Dataset)

R&D spend	Administration	Marketing spend
8	15	30
4	5	20
15	10	41
12	15	26
2	16	3

MICE algo
Three categories of Missing Data:
 - MCAR (missing completely at Random)
 - MAR (missing at Random)
 - MNAR (missing Not at Random)

In this there is not much info b/w the col so other cols cannot be used to fill the missing data in a col.

→ Target col.
Profit → Numerical
 ∵ Regression problem.

2) Removing the target col.

R&D	Ad	MS
8	15	30
4	5	20
15	10	41
12	15	26
2	16	3

3) Introducing some fake nan values.

R&D	Ad	MS
8	15	30
NAN	5	20
15	10	41
12	NAN	26
2	15	NAN

Now we need to predict the missing values using MICE algo.



Predicting the Missing values using HICE Algo

Page 32.

Step 1. Fill all the NaN values with the Mean of resp col.

R&D	Ad	MS
8	15	30
15	5	20
12	10	41
2	11.25	86
	15	29.25 + mean MS

Step 2. Remove all col missing values.

R&D	Ad	MS
8	15	30
NAN	5	20
15	10	41
12	11.25	26
2	15	29.25

Step 3. predict the Missing values of cols using other col

How -> using Any ML Algo (regression, decision tree)

Here: our input training data is. with R&D as target col & test data we provide

the input $\frac{Ad}{5}$ to the Algo after training & it predicts the output. say here the prediction

is 23.14.

R&D	Ad	MS
8	15	30
23	5	20
15	10	41
12	11.25	26
2	15	29.5

Step 4. Remove all Col 2 missing values

R&D	Ad	MS
8	15	30
23	5	20
15	10	41
12	NAN	26
2	15	29.5

↓ output

col

We train ML model of input (X) with output Ad

R&D	MS	Ad
8	30	15
23	20	5
15	41	10
12	29.5	15

& then provide input

R&D	MS
12	26

& predict the New output of Ad.

steps. predict the missing values of col 2 using other col's say we got 11.06

Step 6. continue for col 3.



After that say we have:

R&D	Ad	MS
8	15	30
23.14	5	20
15	10	41
12	11.06	26
2	15	31.56

→ Iteration 1

Diff b/w iteration 0 & iteration 1

Iteration 1 - Iteration 0 = $\frac{0}{0}$

13.87

Ad

0

0

0

-0.19

0

0

0.31

→ the diff should
be 0 or
almost 0

So again we repeat the process:-

~~iteration 1~~ By taking iteration 1 as Base Table:

We remove all NaN from Col 1 & predict it using the rest
& so on

& get iteration 2 table & calculate iteration 2 - iteration 1
Say we got iteration 2

R&D	Ad	MS
8	15	30
23.78	5	20
15	10	41
12	11.22	26
2	15	31.56

12-11

Difference

R&D	Ad	MS
0	0	0
0.64	0	0
0	0	0
0	0.16	0
0	0	0

iteration 2 - iteration 1
if all 0 or about 0
then done otherwise
continue.

and again continue
the iterations
till we are
able to get actual
values of NaN.

Outlier (Charmji ka beta)

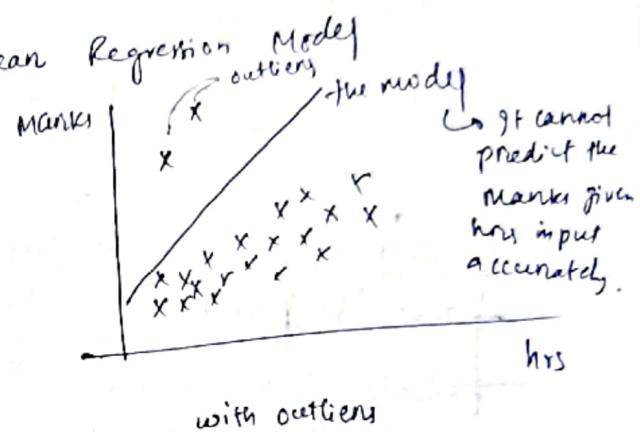
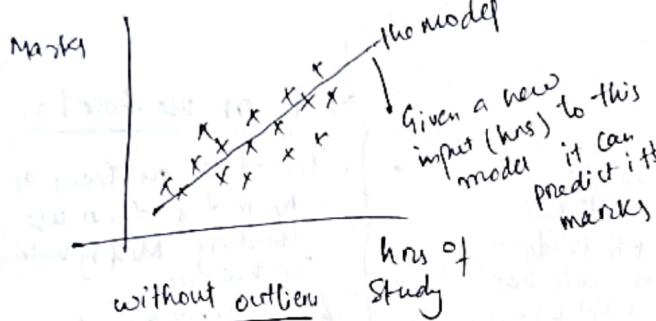
Ek aisa data point jo baaki data se alog behane karta hai & aise data points ML Algos ko distract kar sakte hai.

Suppose ek class hai... we are finding the ~~mean salary~~ of that

class → say ₹5000 per month

Ab usi class mei ek outlier aaya (Bill Gates) then
Now the mean salary will be 50 crores per month
Now this new mean is ~~can't~~ not a true representation of the
salary of that class.

Ex: consider we are training Linear Regression Model



* It is important to understand when is outlier Dangerous & when it is part of Data.

When is outlier Dangerous?

Say age data mei kisi ne galti ki 300 de diya → this has to be removed.

But sometimes outliers are imp (for anomaly detection mechanism)
Hence outliers must kept in the data.

Thus, outliers rakhna hai ya nahi → very imp thing → we should think thoroughly.

Ex in: above ~~hrs~~ of study v/s marks ; we have two outliers but they are still students of the class → so we could make an IQ col of each student & this may justify the outliers (having high IQ)

Effect of outliers on ML Algos

if data has outliers then the performance of following ML algs will decrease →

- Linear Reg
- Logistic Reg
- Adaboost

in these algos we calculate weights even in Deep Learning Algos.

outliers hampers weight-based ML Algos.

Tree Based Algos

- Decision Tree
- Random Forest
- Gradient Boosting
- XG Boosting

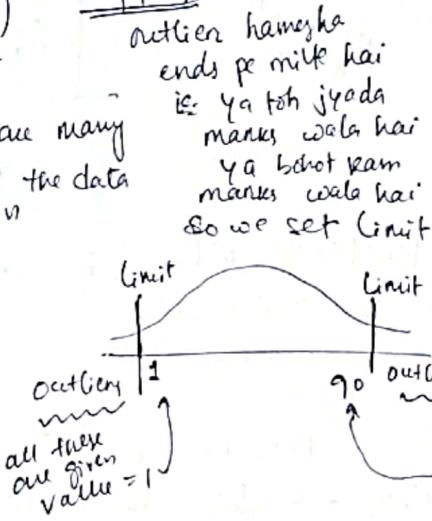
inse outliers ka utna farak nahi pada.

How to treat outliers?

Trimming (Removing)
adv:- Very fast

disadv :- if there are many outliers then the data becomes thin

Capping



→ Kam use karte hai
Treating outliers as NaN & then use handling Missing Values Techniques

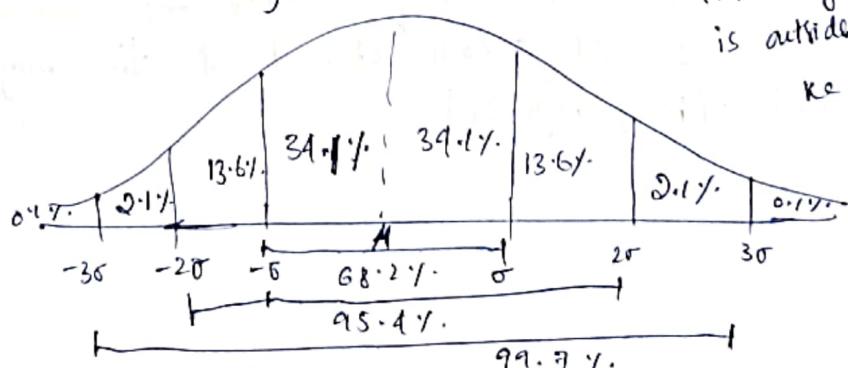
Discretization kardo
↓
Numerical data ko discretize karna -
0-10, 10-20, ...
90-100

then 90 se 4 par jo outlier tha toh wo ab 90 se earth mge ko 75 se ab 90° outlier nahi rha.

How to detect outliers.

Three imp methods.

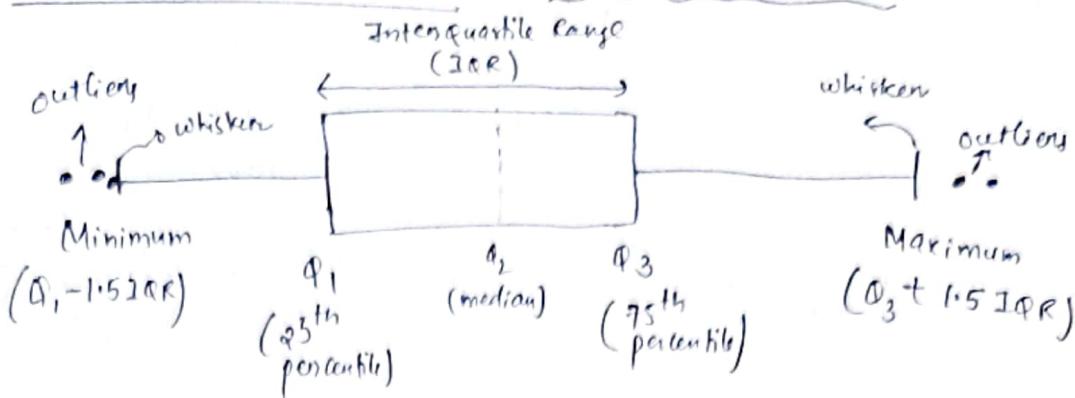
1/ Normal dist. (if the col with outlier is Normally dist)
(Z-score treatment)



ya close to →
Hence agar apne observation is outside ($\mu - 3\sigma, \mu + 3\sigma$)

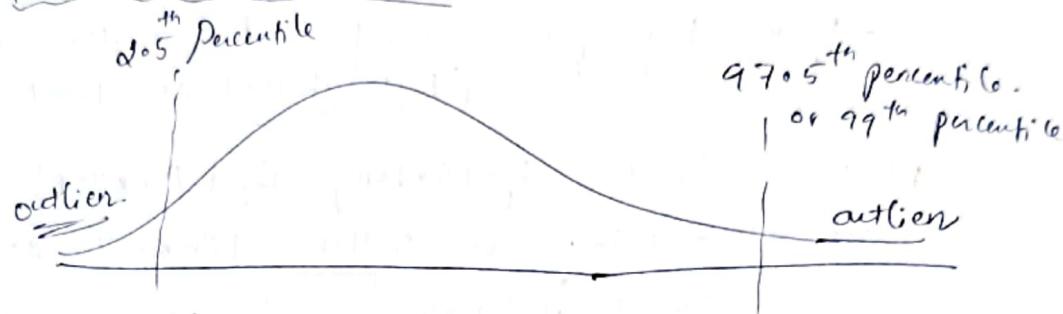
ke bahan hai then wo can treat it as outlier.

2/ If our data is skewed (Left or Right skewed) page 39
Then we use Box Plot (IQR Based filtering)



3/ other Distribution (percentile based filtering)

Percentile Based approach



4/ Winsorization (later).

* Z-score technique for outlier Detection & Removal

↳ The col with outliers must be Normally dist.
↳ outside $(\mu - 3\sigma, \mu + 3\sigma)$ the value is an outlier (or almost Normally dist)

$$\frac{\text{Age}}{27} \xrightarrow{\text{Z-score}} x_i' = \frac{x_i - \mu}{\sigma} \quad \text{say } x_0' = 3.5 \quad \Rightarrow x_i = 3.5\sigma + \mu$$

\therefore if $x_i' \rightarrow (-3, 3)$ $\therefore x_i$ is an outlier.

the data (x_i) is ok correspondingly.

& if $x_i' > 3$ or $x_i' < -3$ then x_i is an outlier

now we ...



Scanned with OKEN Scanner

Say x_i is an outlier then what to do :-

trimming

Remove those rows

capping

if $x_i' < -3$

then take $x_i' = -3$ then $x_i = -3 \times 6 + 4$

if $x_i' > 3$ then take $x_i' = 3$

then $x_i = 3 \times 6 + 11$.

Hence they are no more outliers.

Jupyter Notebook.

(IQR Proximity Rule)

* Outlier Detection and Removal using IQR

→ This technique is used when the data distribution is skewed. (Left skewed or Right skewed).

Values outside $(Q_1 - 1.5 \times IQR, Q_3 + 1.5 \times IQR)$ are outliers → They are either trimmed or capped.

Jupyter Notebook.

* Outlier Detection & Removal Technique using Percentile Method

• say you got 99 percentile in an Exam → this means that 99% people giving that exam is behind you.

• Max value = 95 → It is 100 percentile

min value = 10 → It is 0 percentile.

50 percentile → Median.

• Given Data :-

we make its distribution

then we decide a amount

like upper and nothe values so we decline
outlier. after that we do

Trimming

Capping

(In this Technique Capping is called Winsorization).

Jupyter notebook.



construction

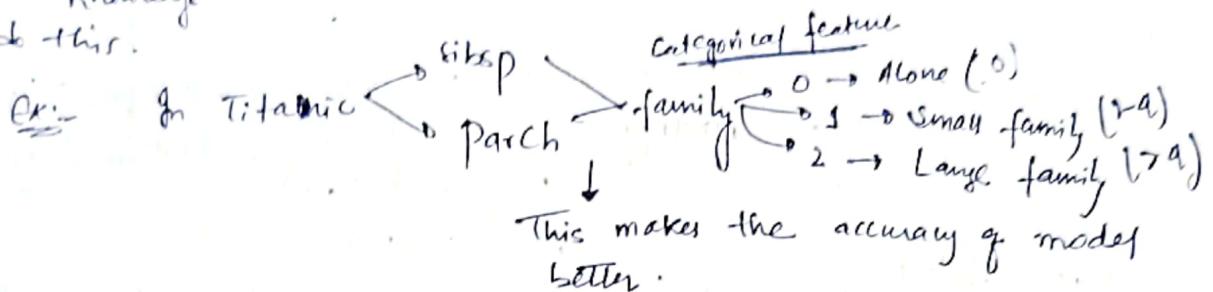
Page 35.

Feature

↳ Given features are Manually new

There is no process,
we need to think,
use intuition,
domain knowledge
to do this.

features create karte hai.
+ feature splitting.



→ feature splitting: sometimes our data is not tidy i.e.
ex single cell mei ok se jyada data - thusa
hua hai - so we split the corresp. col.

Ex: in Titanic data set: Name | Salutation
Mr. Ankit → Mr. | Ankit -
Jupyter Notebook.

CURSE OF DIMENSIONALITY (CoD)

Both the topics Feature selection & Feature Extraction (PCA) in feature eng. are related to CoD.

After this:- we study feature construction
then ML Algos & then
After that feature selection.

(∴ feature selection is generally used
after ~~model~~ Model building that tells
ki feature kahan pe kuch fayda hota hai
ki nahi ML algo performance pe).

In Dataset, we have cols/features (Dimensions).
When we use ~~a~~ ML model - there is a threshold
of dimensions/cols/features, after which if we increase the No. of
cols then the performance of the model doesn't improve infact
in some case it may deteriorate.

Curse of Dimensionality say as we add features the perf of the model
will increase but there will be an optimum point after which
even if we add more features the perf will not improve.

examples of high dimensional / features Dataset:-

- 1) Images } In this case selecting optimum features
- 2) Text data } becomes very useful.

The problem with higher dimensions is : sparsity

n cols \rightarrow n -dimensional ($n \rightarrow$ very large, 1000, 2000, 3000 etc.)

Each row is a coordinate in the n -dim

& if we plot them the points are very sparse (far) from each other. \rightarrow Analogy :- wallet lost on road (at 100-10)
playground (2D)
Building (3D)

then finding wallet on 10 is easier. (since wallet to few houses have same hair).

* Many ML Algos are based on statistical calculations like distance, but in higher dimensions the data points become sparse & hence the ML Algos fail due to sparsity.

The Soln is Dimensionality Reduction:-

The Problem before was:-

- ① perf \downarrow ss

- ② Computation increases.

Dimensionality Reduction

Feature Selection

Say we have features

$$\{f_1, f_2, \dots, f_n\}$$

we take a subset :-

$$\{f_1, f_3, f_7\} \text{ & }$$

run the model on it & see its performance & do it for

Many (prob all) subsets

to see if we can remove some features & improve or maintain the accuracy.

+ Forward Selection

+ Backward Elimination

Feature Extraction

$$\{f_1, \dots, f_n\} \xrightarrow{\text{ET}}$$

completely new set

- + PCA (Principal component analysis) $\{e_1, e_2, \dots, e_k\}$ $\xrightarrow{\text{ET}}$ k cols create better features
- + LDA (Linear Discriminant Analysis) $\{e_1, e_2, \dots, e_k\}$ $\xrightarrow{\text{ET}}$ there are actually linear comb of the $\{f_1, \dots, f_n\}$
- + TSNE.

and use these new features.

PCA (Principal Component Analysis)

- ↳ It is an unsupervised ML problem (we have input & output)
- ↳ It is an useful, old, reliable technique & its maths is quite comple.
- ↳ It is a feature extraction Technique.
- ↳ Its goal is to reduce the dimensionality without distorting the performance of the Model.
- ↳ i.e. PCA best possible lower dimension mei letane data hai while keeping the essence/behaviour of the data intact

Benefits of using PCA

- ↳ Training & prediction data ka size is reduced.
- ↳ faster execution of Algorithms.
- ↳ Visualization (Suppose we have dataset with 10 cols/features/dim.
But we cannot visualise it in 10 dim, PCA reduces it to 3 or 2 & we are now able to visualise it.)

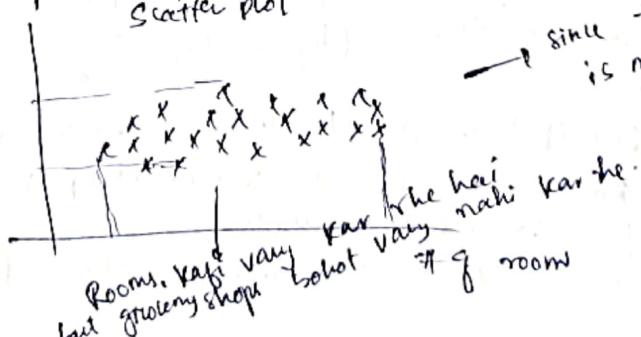
Geometric Intuition Behind PCA

Ex: House price prediction		Output
No. of rooms	No. of grocery shops around house	Price (in Lakh)
3	2	60
4	0	130
5	6	170
2	10	90

This we did ∵ he had some domain knowledge

But what if we don't have domain knowledge then we use mathematics to find input cols on which the output depends the most :-

plot:-



* we now know what feature selection is :- we select a subset of the present features that are important in predicting the output.

→ Agar yaha dono input col mei se ek ko rakhna hai then we will keep the one jispe output jyada depend kar raha hai i.e. # of rooms and remove the other one.

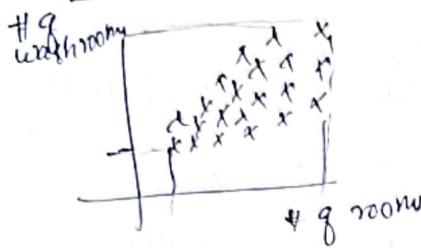
→ since the spread of # of rooms is more ∵ we keep # of rooms as imp col.

we can connect this spread with variance. More variance → More spread

Now suppose we have changes in Data

# rooms	# washrooms	Price
:	:	:
:	:	:
:	:	:
:	:	:

in this case if we
Scatter Plot



→ the DA is same:- what col should we keep if we have to keep only one.

Now both room & washroom pe depend kanga price tence yaha feature Selection will not work (variance wash trick).

→ we see the spread of both the data/cfs are comparable hence mathematically we cannot decide which col to keep using feature selection.

In such case → we take help of feature Extraction

# rooms	# washrooms	Price
:	:	:
:	:	:
:	:	:
:	:	:

Even if real estate Broker ko sole ki kanga col imp hai, kisi ek ko hi select karo → he will not be able to select it → other way around is:-

Collect your data in a different way since more the # rooms + # washrooms more is area

size (area)	Price
:	:
:	:
:	:

& Now we have only one col.

↓ This is exactly what PCA does

it forgets the existing features, create a new set of features & chooses a subset from these new features that are more important for output prediction.

→ upan to Broker ne bata diya :- he had domain knowledge:-

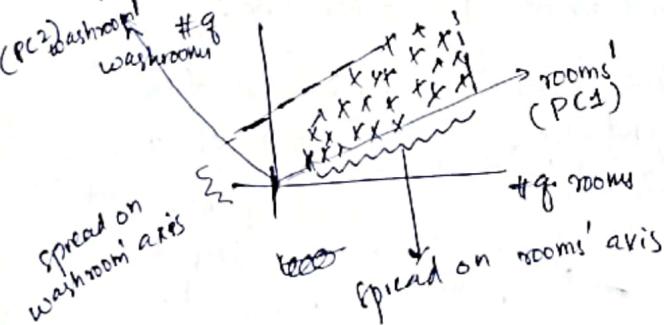
as a data scientist

how do we do it:-

→ yaha feature selection technique fail kar gya tha since variance dono column same aarha tha.

↓ so what will PCA do :-

PCA will find new set of coord axes (by rotating the original coord axes)



rooms → PC1 → principal component 1, washrooms → PC2 → Principal Component 2.

& Now rooms' (PC1) ko varience & washrooms' (PC2) ko kala denge.
↳ ∵ variance of data on PC1 is more.

Now we transform the Data according to the PC1

If Note:-

No. of principal components = # of original cols/features / Dimensions :-

What variance represents

It is a statistical measure that gives info about the spread of the data.

If we have dataset:

$$\text{then } \text{Var} = \sigma^2 =$$

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

Note: spread exactly variance nahi hoga.
But spread of variance.

* Mean abs. deviation

(MAD) is not used to represent spread in PCA since MAD is not diffible -- so we g optimization model then is not possible.

Now X axis data has more spread hence in PCA we choose the X axis data

Why?

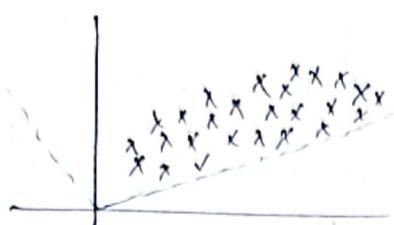
∴ When we choose X-axis data then the ~~actual~~ distance b/w orange & green is d_1 which is comp to d (actual dist)
hence ML algo (based on distances) will not be affected
but suppose we take the less spread data (y-axis data) as input & removed the x-axis data
then the dist b/w orange & green pt is d_2 which is nowhere near the actual distance of hence if our ML model is based on distance, will never know the actual distance d , & will work on ' d_2 ' & the output will be affected.

Hence we take the data/features with none spread / variance as input.

Hence jab hum higher dim & lower dim mei between aat hai toh data pts ke aapne see distance/behaviours disturb na ho, isliye hum variance ko maximize karte hain.

Problem Formulation in PCA

Suppose we have input data in 2D & we want it in 1D

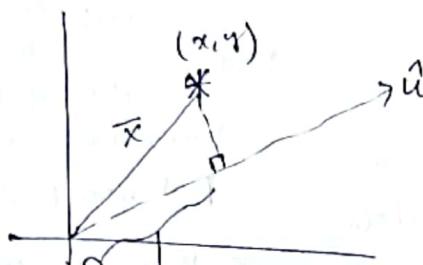


Consider
only one point..

This direction
we need
is the unit
vector we need.

(ii)

Humm ek single aisa axis find
Karne hai jiske upar ham
apne data ko project kar
paye & humne utne hi acche
results milte jitne 2D mei mil
sakte hain.



$$\text{Proj } \vec{x} \text{ on } \vec{u} = \frac{\vec{x} \cdot \vec{u}}{|\vec{u}|} = \vec{x} \cdot \hat{u} = |\vec{x}| \cos \theta = \hat{u} \cdot \vec{x} = \hat{u}^T \vec{x}$$

Hence: we need to choose that unit vector \hat{u} such

that the variance is maximised.
choose a \hat{u} : project the points say $u^T x_1, u^T x_2, \dots, u^T x_n$
on \hat{u} . Now to find the variance of these
projected point use $\sigma^2 = \text{var} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$.

Now PCA finds that \hat{u}
corresponding to which these

$$= \frac{1}{n} \sum_{i=1}^n (u^T x_i - \bar{u}^T \bar{x})^2$$

Variance is maximised

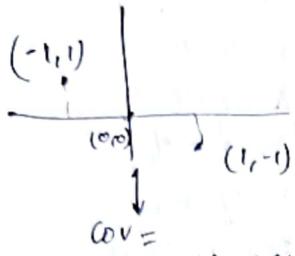
Now this becomes an optimization problem.

This is called
Mathematical
objective fn.

To understand the solⁿg of the Opt. problem, Page 38
we need to understand the Covariance & covariance Matrix.

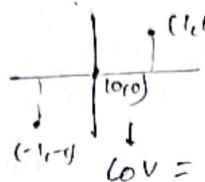
Covariance & Covariance Matrix

- * Variance is a metric of a single axis, a single feature.
- * Covariance is a metric of two axes, two features.



$$\text{cov} = \frac{(-1-x_1) + (0-x_0) + (1-x_1)}{3}$$

\hookrightarrow -ve means
as $x \uparrow$ sees $y \downarrow$.



$$\text{cov} = \frac{(-1-x_1) + (0-x_0) + (1-x_1)}{3}$$

$$= 2/3 \rightarrow \text{the means as } x \uparrow \text{ sees } y \uparrow$$

\hookrightarrow cov is similar to corr; except corr is restricted to $[-1, 1]$ but cov has no restrictions.

\rightarrow same diag.
 \rightarrow do acc. to variance they are same but they are not.

Covariance Matrix:

say we have:- $\begin{array}{|c|c|} \hline x_1 & x_2 \\ \hline \vdots & \vdots \\ \hline \end{array}$

then Cov Matrix \rightarrow will be 2×2 matrix

$$\begin{matrix} x_1 & \begin{pmatrix} \text{cov}(x_1, x_1) & \text{cov}(x_1, x_2) \\ \text{cov}(x_2, x_1) & \text{cov}(x_2, x_2) \end{pmatrix} \\ x_2 & \end{matrix} = \begin{pmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) \\ \text{cov}(x_2, x_1) & \text{var}(x_2) \end{pmatrix}$$

then Cov Matrix.

$$\begin{Bmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) & \text{cov}(x_1, x_3) \\ \text{cov}(x_1, x_2) & \text{var}(x_2) & \text{cov}(x_2, x_3) \\ \text{cov}(x_1, x_3) & \text{cov}(x_2, x_3) & \text{var}(x_3) \end{Bmatrix}$$

Square Symmetric Matrix

$$\begin{aligned} \because \text{cov}(a, a) &= \text{var}(a) \\ \text{and } \text{cov}(a, b) &= \text{var}(b, a) \end{aligned}$$

This Matrix is important since it has all the variances that tells the spread of data on each axis. $\text{var}(x_1), \text{var}(x_2), \text{var}(x_3)$

And also tells the relation b/w orientation of all the pairs of axes by cov. Cov tell the

Eigen Decomposition of Covariance Matrix :-

Point:- Eigen vectors & Eigen values. (Matrices as linear transformation)

for this Read inside 100 days of ML

Lecture 47 PCA part 2 Link in Descip. visiondummy.com.

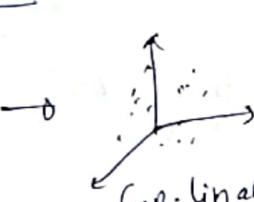
The covariance matrix defines both the spread (variance) & the orientation (covariance) of our data. So, if we would like to represent the covariance matrix with a vector and its magnitude, we would simply try to find the vector that points into the direction of the largest spread of the data & whose magnitude equals to the spread (variance) in this direction. means jiska eigen value sabse jiska ha

In other words, the largest eigen vector of the covariance matrix always points into the direction of the largest variance of the data, & the magnitude of this vector equals to the corresponding eigen value.

The second largest eigen vector is always orthogonal to the largest eigen vector & points into the direction of the 2nd largest spread of data.

Step by step so:-

Input dataset:-

$f_1 | f_2 | f_3$
 $\vdots \quad \vdots \quad \vdots$
 \rightarrow 
(np. lin alg)

Step 3:- find Eigen values & Eigen vectors of the covariance matrix.

Step 1:- Mean centering of Data

not a mandatory step

but PCA improves if done mostly.

Step 2:- find cov Matrix.

$$\begin{matrix} f_1 & [v_{f_1} & g_{f_1} & f_{f_1}] \\ f_2 & [c_{f_1 f_2} & v_{f_2} & c_{f_2 f_3}] \\ f_3 & [g_{f_1 f_3} & c_{f_1 f_3} & v_{f_3}] \end{matrix}$$

(np. cov using numpy)

& jis eigen vector ka Eigen Value sabse jyada hai wo hi PC1 banta hai, then PC2 then PC3 (all these are orthogonal to each other)
Now it's our choice if we want PC1, PC2 \rightarrow 2D or only PCs \rightarrow 1D.

Now after getting the principal components then Page 39
how do we transform the points. $f_1 | f_2 | f_3 \rightarrow p_{c1} | p_{c2}$
i.e. originally in 3D \rightarrow 2D if we choose p_{c1}, p_{c2}
" \rightarrow 1D if we choose p_{c1} .

3D \rightarrow 1D.

Say there
are 1000 pts
in 3D

$p_{c1} \& p_{c2}$ (plane)

Take the projections of the points on p_{c1} is.

dot product of pts with p_{c1} .

The shape of pts is $(1000, 3)$ matrix

& the p_{c1} is $(1, 3)$ matrix

dot product gives $(1000, 1)$ matrix
which will be our p_{c1}

3D \rightarrow 2D.

projection on the $p_{c1} p_{c2}$ plane.

again shape of datapoints is $(1000, 3)$ & shape of p_{c1}, p_{c2} plane

is $(2, 3) \rightarrow$ take dot product i.e. $(1000, 3) \cdot (2, 3)^T$

$= (1000, 2) \rightarrow$ $p_{c1} | p_{c2}$ this is w^T

Jupyter NoteBook.

Then working with scikit learn PCA implementation
of MNIST dataset.

MNIST Dataset is collection of images & images
are of digits (0 - 9) by different people.

And it is a classification task to predict which digit
is written on it.

The code is on Kaggle

search Digit Recognizer Kaggle (first link)

then go to Code \rightarrow New notebook (rename as ^{regd.})
run the cell & use the train data.

Each image consist of $28 \times 28 = 784$ pixels & in the dataset
each pixel is a col.

And we have 42000 images : The dataset has
42000 rows and 784 cols. and 1 label col.
the label col indicate which digit is there in the image

at finding the optimum no. of principal components:-

Say we have MNIST dataset \rightarrow 784 cols. (input)

then the eigen values of the covariance matrix

$$\text{are } \lambda_1, \lambda_2, \dots, \lambda_{784}$$

this denotes
the spread/variance of
data on PC₁.

This explains the spread of data on PC₁.
Converting them to %.

$$\left(\frac{\lambda_1}{\sum_{i=1}^{784} \lambda_i} \times 100 \right) \%, \left(\frac{\lambda_2}{\sum_{i=1}^{784} \lambda_i} \times 100 \right) \%, \dots, \left(\frac{\lambda_{784}}{\sum_{i=1}^{784} \lambda_i} \times 100 \right) \%$$

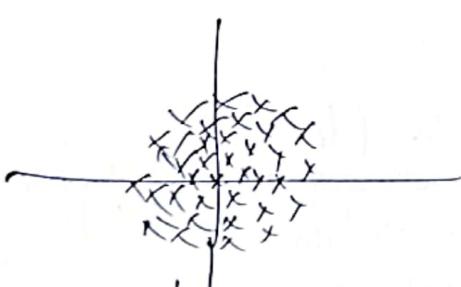
this is how much %
of variance of data is explained
by λ_1 and so on & similarly total
90% variance explained

hence humne utne values (λ_i 's) lete hai jitne mei
jante jante hum 90% poorsch jaye.

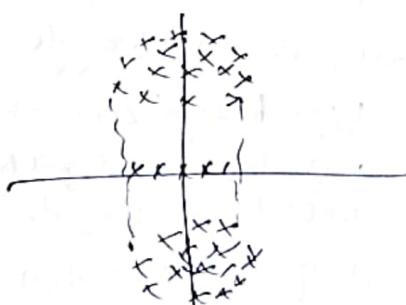
Say $(\lambda_1 \% + \dots + \lambda_{15} \%) = 90\%$. then we take the
PC₁ - - - PC₁₅ \rightarrow ie 15 principal components are the
Optimum components for good/best accuracy.

Supyfer Notebook (Kaggle).

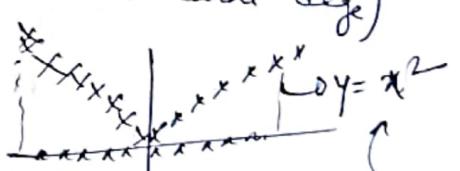
When PCA Does not work? (ie data is aisa hai
ki wo PCA ko kaam hi
nahi karne dege)



The data is a circle
hence the spread/variance
on both axes is same & if
we even rotate the axes,
the variance will again be
same in the new coord system
hence PCA will not work.



Dono ka projection
le toh wo ek dusre
pe overlap kar jaoge &
PCA do points ko diff.
hi nahi kar payega.



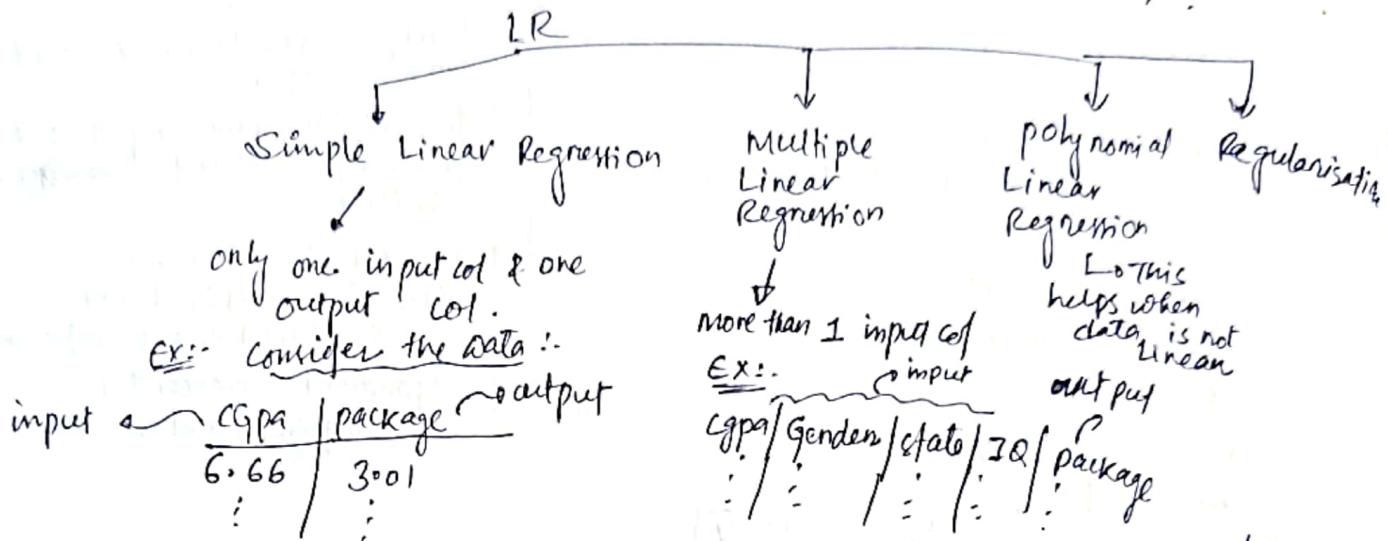
Data has some patterns
iska projection it
in lower dim the pattern
is now linear hence
the pattern is lost.



MACHINE LEARNING ALGORITHMS

* Linear Regression (LR) :-

- It is a supervised ML algo
- used on regression problem (Dataset whose output is Numerical)



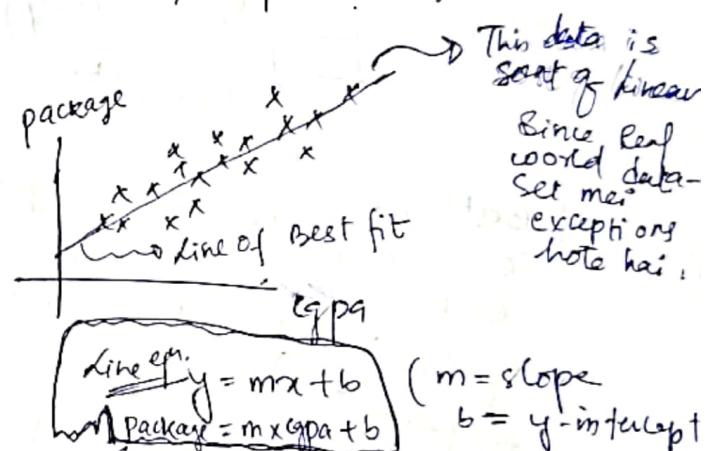
• Simple Linear Regression

input	CGPA	package	output
	7.1	3.5	
	4.7	1.2	
	8.9	4.2	
	8.1	3.9	

plot this data

we need to make a model that when given CGPA it predicts the package.

And this line of best fit is the Linear Regression Model.



In this case, we drew the line of best fit, the line that is as close as it can be to all the points.

finding m and b .

Jupyter Notebook

finding the optimum no. of principal components:-

Say we have MNIST dataset \rightarrow 784 cols. (input)

then the eigen values of the covariance matrix

$$\text{are } \lambda_1, \lambda_2, \dots, \lambda_{784}$$

this denotes

the spread/variance of data on PC1.

This explains

the spread of data on PC1.

Converting them to %.

$$\left(\frac{\lambda_1}{\sum_{i=1}^{784} \lambda_i} \times 100 \right) \%, \left(\frac{\lambda_2}{\sum \lambda_i} \times 100 \right) \%, \dots, \left(\frac{\lambda_{784}}{\sum \lambda_i} \times 100 \right) \%$$

this is how much % of variance of data is explained by λ_1 and so on & humne total 90% variance explain karwana hoga hai

hence humne utne values (λ_i 's) lene hai jitne mei jonte jonte hum 90% poorsch jaye.

Say $(\lambda_1 \% + \dots + \lambda_{15} \%) = 90\%$. then we take the PC1 ... PC15 \rightarrow i.e. 15 principal components are the optimum components for good/best accuracy.

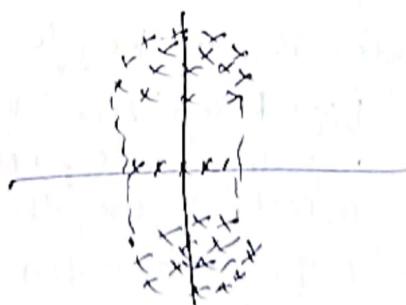
Supyfer Notebook (kaggle).

When PCA Does not work? (i.e. data is aisa hai ki wo PCA ko kaam hi nahi karne dege)

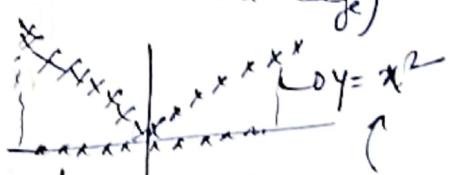


The data is a circle

hence the spread/variance on both axes is same & if we even rotate the axes the variance will again be same in the new coordinate system hence PCA will not work.



Dono ke projection ke tab wo ek dusre pe overlap karijenge & PCA do points ko diff. hi nahi kar payenge.



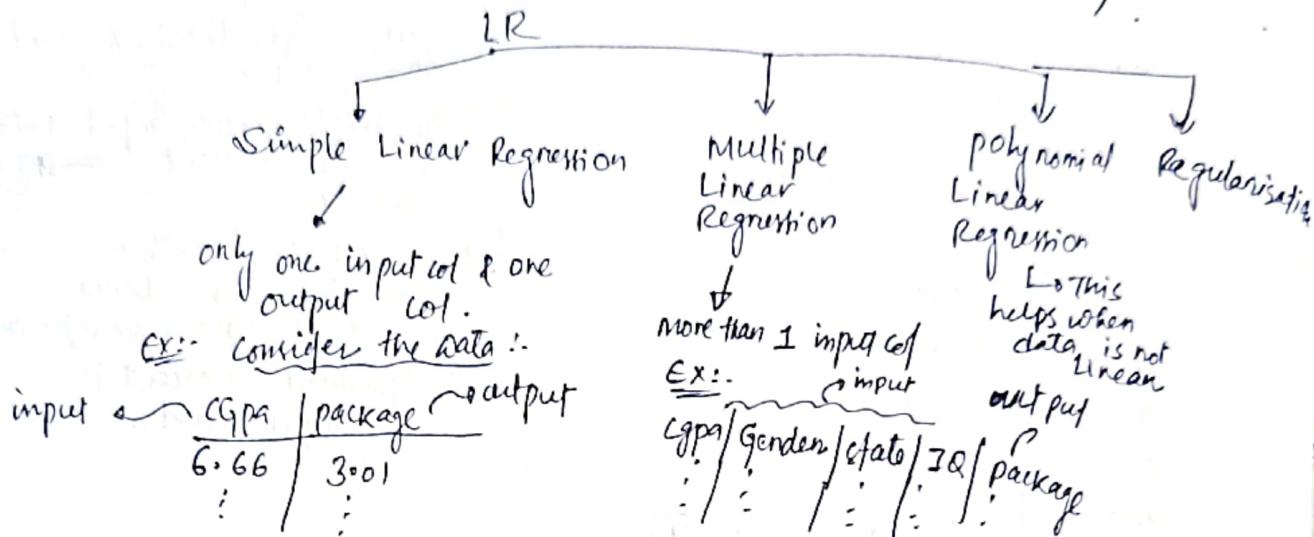
PCA has some pattern iska projection ti in lower dim the pattern is now linear hence the pattern is lost.



MACHINE LEARNING ALGORITHMS

* Linear Regression (LR) :

→ It is a supervised ML algo
Used on regression problem (Dataset whose output is Numerical)



• Simple Linear Regression

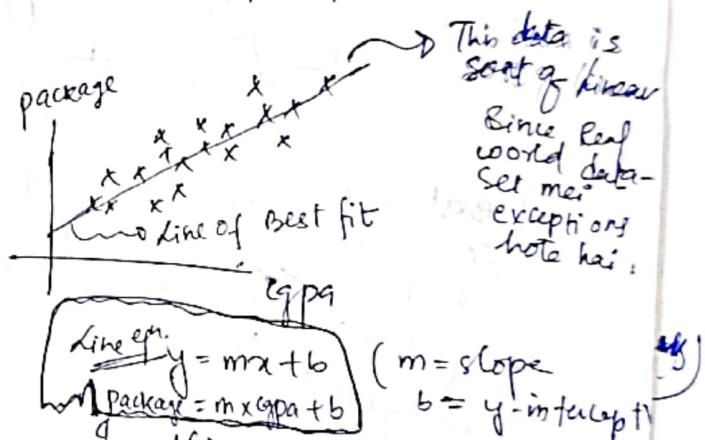
input	CGpa	package	output
	7.1	3.5	
	4.7	1.2	
	8.9	4.2	
	8.1	3.9	

plot this data

We need to make a model that when given CGPA it predicts the package.

And this line of best fit is the

Jupyter Notebook



In this case, we drew the line of best fit. The line that is as close as it can be to all the points.

finding m and b .
Linear Regression Model.



Mathematical model of Linear Regression

How to find m & b of the line of best fit

Closed form solⁿ

↓
in this case we have formulae to calculate values/sols
This technique is called OLS (ordinary least squares)
↳ scikit Learn uses this internally when we use the class LinearRegression()

↓
here the formula is:-

$$b = \bar{y} - m \bar{x}$$

$$y = \text{package}(\bar{y} + \text{mean})$$

$$x = \text{GPA} (\bar{x} + \text{mean})$$

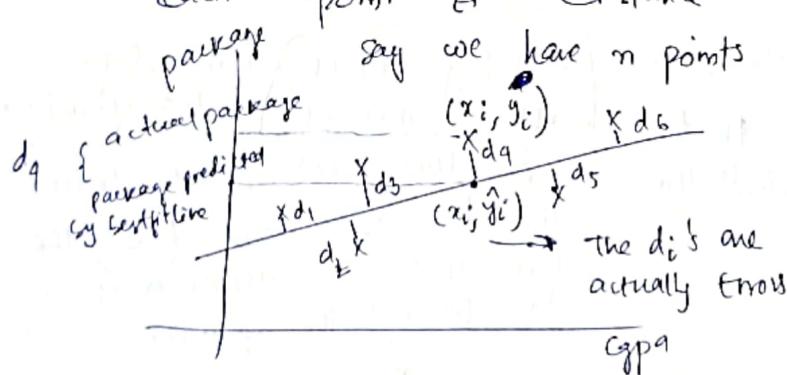
$$m = \text{slope}$$

$$\& m = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Non-closed form solⁿ.

Here we use approximation techniques to reach to the solⁿ.
↳ Here we use calculus
↳ Using Gradient Descent in ML.
↳ In higher dim input data Gradient Descent ~~works~~ works better
↳ In ML :- there is a class in scikit Learn called SGDRegressor() → in it Gradient Descent is implemented.

* Best fit line ko hota hai jo saare points ke paas ce shokan gijanne ka try karega, ye line each point ki distance minimize karta hai.



Thus we need such and m, b such that E is minimised.

So $d_i = y_i - \hat{y}_i$; $\forall i = 1, 2, \dots, n$ (\hat{y}_i is the predicted value)

$$\therefore E = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\rightarrow \text{average error ke liye } \frac{1}{n} \text{ karo})$$

$$\therefore E = d_1 + d_2 + \dots + d_n \quad \text{Total error}$$

$$\text{or } E^2 = d_1^2 + d_2^2 + \dots + d_n^2$$

need to minimize this (square liya mod nahi since \rightarrow we need to penalise outliers → mod is not diff at '0')

$$\therefore E = \sum_{i=1}^n d_i^2 \rightarrow \text{error fn or loss fn}$$

Hence we need such a line (m, b) s.t.

$$E = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \text{ is minimum.}$$

Note: $\hat{y}_i \rightarrow$ predicted value
 $\therefore \hat{y}_i = mx_i + b$

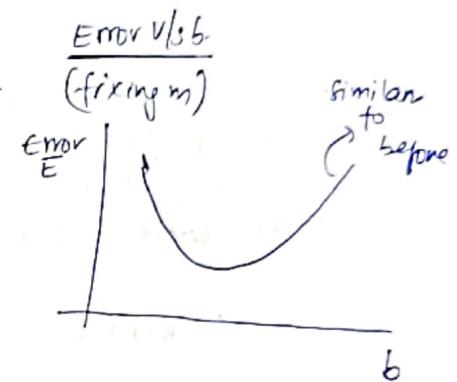
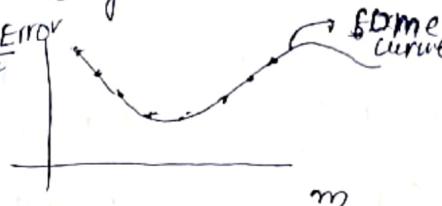
$$\therefore E = \sum_{i=1}^n (y_i - mx_i - b)^2$$

$$\Rightarrow E(m, b) = \sum_{i=1}^n (y_i - mx_i - b)^2 ; x_i, y_i \text{ are the input & output data resp. \& they are fixed.}$$

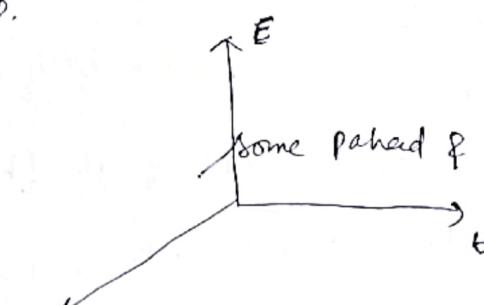
Now we want to choose those m, b s.t. $E(m, b)$ is minimised.

Note:
 m fixed; b change \rightarrow Translation
 m change, b fixed \rightarrow Rotation about b ,
 m change, b change \rightarrow Translation.

Error v/s m .
 E (fixing b)



\therefore in 3-D.



and using Maxima & minima (multivariable calculus)
we can find the pt. of minima.

$$\therefore \frac{\partial E}{\partial m} = 0 \quad \text{&} \quad \frac{\partial E}{\partial b} = 0 \Rightarrow \sum -2(y_i - mx_i - b) = 0$$

$$\sum (y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

$$\Rightarrow \sum (y_i - mx_i - b) = 0$$

$$\sum (y_i - \bar{y} + m(x_i - \bar{x})) (x_i - \bar{x}) = 0$$

$$\sum y_i - \frac{m \sum x_i}{n} - \frac{\sum b}{n} = 0$$

$$\sum (y_i - \bar{y} - m(x_i - \bar{x}))^2 = 0$$

$$\bar{y} - m\bar{x} = \frac{nb}{n}$$

$$\sum ((y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2) = 0$$

$$\boxed{b = \bar{y} - m\bar{x}}$$

$$\boxed{m = \frac{\sum (y_i - \bar{y})(x_i - \bar{x})}{\sum (x_i - \bar{x})^2}}$$

Jupyter Notebook.

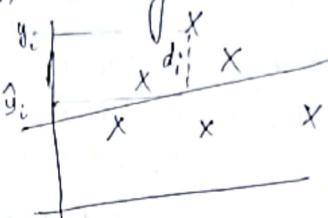
Regression Metrics

- (1) MAE → mean Absolute Error,
- (2) MSE → Mean Squared Error,
- (3) RMSE → Root Mean Squared Error,
- (4) R²-score → Coefficient of determination or Goodness of fit.
- (5) Adjusted R²-score.

① MAE :-

We are working on a Regression problem let's say simple

Linear Regression :-



Regression Line (Line of Best fit)

We know what d_i is → d_i → i^{th} point Regression line

ke respect mei kisi Galta kar raha hai. (we saw: $d_i = y_i - \hat{y}_i$)

then ~~total error~~ = $|y_1 - \hat{y}_1| + \dots + |y_n - \hat{y}_n|$

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

Given in the data

→ It is basically the loss function.

Advantage :- (1) MAE has same unit as y-axis (here LPA) lakhs per annum

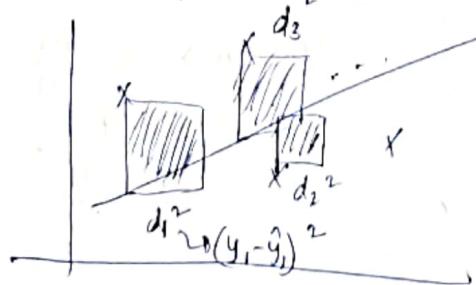
(ii) Robust to outliers (handles outliers relatively better)

Disadvantage :- (1) MAE is not diff'ble at '0' (\because mod)

② MSE (Mean Squared Error)

$$\text{MSE} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \rightarrow \text{In Linear Regression, we used MSE as our loss function.}$$

Geometrically what MSE represents:-



Explain: we take the mean/average of this area = MSE & try to minimize it.

Adv: it is diff'ble everywhere

Disadv: (1) the unit of MSE is square of the unit of y-axis, hence it is a bit difficult to interpret

like: $\text{MSE} = 11.25 (\text{LPA})^2 \rightarrow \text{loss} = \sqrt{11.25} \text{ LPA}$

(ii) Not Robust to outliers ∵ for outlier very big areas are considered in MSE

Hence: agar data mei jyada outliers hai then go with MSE & if kam hai then go with RMSE.

Page 42

* (3) RMSE (Root Mean Squared Error) = \sqrt{MSE}

$$\therefore RMSE = \sqrt{MSE} = \sqrt{\frac{\sum (y_i - \hat{y}_i)^2}{n}}$$

→ It's unit is same as the output (y -axis)

→ Not robust to outliers, just like MSE.

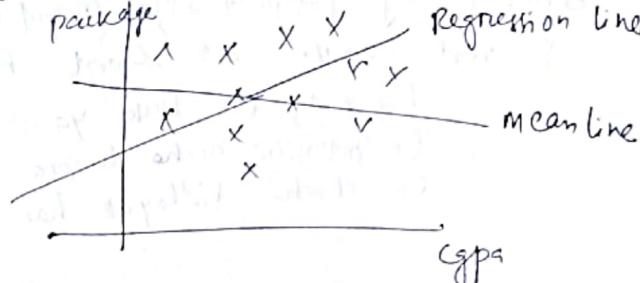
→ As a beginner → focus metrics viral lena.

→ Coeff of determination or Goodness of fit.

(4) R2-Score → It tells us ki model kitha accha perform kar raha hai.

If working with any type of data set with Regression problem.

Suppose if we had only the package carried how much package he only one measure (the mean), of all the packages if tell that as answer but now we have CGPA on which package depends.



Now R2-Score is the calculation ki Linear Regression wala line ; mean line se kitha better hai.

$$\text{formula: } R^2\text{-Score} = 1 - \frac{SSR}{SSM}$$

SSR = sum of squared di's

SSM = " " " "

$$\therefore R^2\text{-Score} = 1 - \frac{\left[\sum_{i=1}^n (y_i - \hat{y}_i)^2 \right]_{\text{regression line}}}{\left[\sum_{i=1}^n (y_i - \hat{y}_i)^2 \right]_{\text{mean line}}}$$

Interpretation of R2-Score:

- if $R^2\text{-Score} = 0 \Rightarrow SSR = SSM \rightarrow$ not good
- if $R^2\text{-Score} = 1 \Rightarrow SSR = 0 \rightarrow$ Best/case scenario

Hence: kitha Reg. line perfection ke taref move karla hai pe hota hai utna R2-Score 1 utna R2-Score 0 ke taref move karla hai.

* what if R²-score is -ve \Rightarrow $SS_R > SS_M \Rightarrow$ Regression line, hamara mean will line (worst case) \hat{y} thi jyada Galhya kar raha hai.
This should not happen.

Another Interpretation of R²-Score:

Say we have $\frac{\text{GPA}}{\vdots} / \text{package}$
& R²-score = 0.80

This means GPA explains the 80% of variance in (X-axis) (input)

the package (IQA) (Y-axis) (output)

i.e. 80% jo variance hai in package (\hat{y} visi ka 3.1 IQ
hai & visi ka 9.2 IQ etc.).

that is due to $\frac{\text{GPA}}{\vdots}$

& rest 20% variance in package \rightarrow we don't know.

fr. $\frac{\text{GPA}}{\vdots} / \frac{\text{IQA}}{\vdots} / \text{package}$

\rightarrow R²-Score = 0.80

Then again 80% variance of package is explained by (GPA + IQ) (Input col)

& rest 20% we don't know.

\hookrightarrow may be Luck ya

\hookrightarrow Interview accha hoga ya

\hookrightarrow chacha Vidhayak hai etc.

(5) Adjusted R²-Score :-

Say we are using R²-Score with dataset $\frac{\text{GPA}}{\vdots} / \frac{\text{package}}{\vdots}$ of the R²-Score = 0.80 & now if we add further input col say IQ \rightarrow $\frac{\text{GPA}}{\vdots} / \frac{\text{IQ}}{\vdots} / \text{package}$ then

now the R²-Score will increase say 0.90. \rightarrow which is OK
package may depend upon ~~IQ~~ IQ.

But say if we add an irrelevant col ie temperature (this should not have any impact upon package) but even then sometimes R²-Score increased indicating temp is affecting package. Hence we use adjusted R²-Score.

formula :- Ad. R²-Score = $1 - \left[\frac{(1 - R^2)(n-1)}{n-1-k} \right]$

\rightarrow in Y-test

n \rightarrow # of rows / datapoints

K \rightarrow total no. of indep. cols

(only GPA \rightarrow K=1)
(GPA + IQ \rightarrow K=2)
(GPA + IQ + temp \rightarrow K=3)

The formula for Ad. R²-score removes the flaw of the R²-score.

Page 43.

Since fallen ke col add karne pe K pse deno hoga
(like temp) $\therefore \frac{(1-R^2)(n-1)}{n-1-K} = R^2_{adj}$

Adding Relevant col say IP $R^2_{adj} < R^2$ bcs.

$K \uparrow$ deno ↓ ; again in $1-R^2$ in num ↓es rapidly ;
R² pse hence Num ↓es. & in such case the ↓e in Num
is greater than the ↓e in Deno, hence R²-adj pse hogya.

Hence R²-adj is better if we are dealing with
Multiple Linear Regression.

Jupyter Notebook

Real world data are of multiple linear regression.
Ex:

MULTIPLE LINEAR REGRESSION

So more than 1 input col Ex:-

cgpa		gender		IQ		package
:	:	:	:	:	:	:

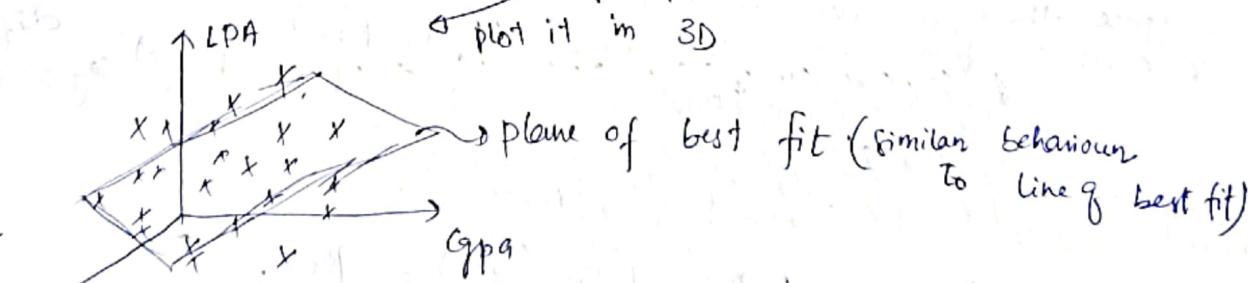
input/independent.

x_1	x_2	x_3	y dependent
:	:	:	:

- Simple LR is a special case of Multiple LR
- The concepts in Simple LR are as it is applicable to Multiple LR.

Consider the dataset :-

cgpa		IQ		package
:	:	:	:	:



And in further higher Dimensions we draw hyperplane of best fit.

Now we need to find the eqn of the plane of best fit:-

$$y = mx_1 + nx_2 + b$$

$$\text{or } y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 = \beta_0 + \beta_1 \text{cgpa} + \beta_2 \text{IQ}$$

so we need to find :- $\beta_0, \beta_1, \beta_2$



So in n -dim, we would be searching the opt of
 $(n-1)$ dimensional hyperplane:-

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_{n-1} x_{n-1}$$

$$y = \beta_0 + \sum_{k=1}^{n-1} \beta_k x_k.$$

The coeff $\beta_1, \dots, \beta_{n-1}$ are like weights. (discussed in Linear Regression)

Say in 3D:-

we are finding:- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
 $LPA = \beta_0 + \beta_1 CGPA + \beta_2 IQ$

Now if β_1 is very less (close to 0) then LPA determine
 karne mei IQ ka jyada contribution hai.

if $\beta_1 > \beta_2$ means ; LPA depends more on CGPA then IQ
 and so on.

$\beta_0 \rightarrow$ intercept / offset value.

Mathematical formulation of Multiple LR

→ finding out $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ (ie $\beta_0, \beta_1, \beta_2$)

for dataset:-
$$\begin{array}{|c|c|c|} \hline CGPA & IQ & LPA \\ \hline : & : & : \\ \hline \end{array}$$

and if dataset is:-

$$\begin{array}{|c|c|c|c|} \hline CGPA & Gender & IQ & LPA \\ \hline x_1 & x_2 & x_3 & \\ \hline \end{array}$$

we find $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$.
 prediction wala y Then
plotting this
in 4D

Suppose there are 100 students \therefore 100 rows:-

In matrix form we can write:- $(\beta_0, \beta_1, \beta_2, \beta_3)$ are known) say:

~~This is the predicted stud. of all 100 students.~~

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_{100} \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \beta_2 x_{12} & \beta_3 x_{13} \\ \beta_0 & \beta_1 x_{21} & \beta_2 x_{22} & \beta_3 x_{23} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_0 & \beta_1 x_{100,1} & \beta_2 x_{100,2} & \beta_3 x_{100,3} \end{bmatrix}$$

The step is iterative so when do we stop writing the same in n rows & $(m+1)$ cols.

Page 44

$$\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \cdots & \beta_m x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_0 & \beta_1 x_{n1} & \cdots & \beta_m x_{nm} \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1m} \\ 1 & x_{21} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nm} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}$$

predicted output $\hat{Y} = X\beta \rightarrow$ The terms have their usual meaning.
we need to find β

Now: The output Data:-

actual output $Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$ & $e = Y - \hat{Y} = \begin{bmatrix} y_1 - \hat{y}_1 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$

in simple Linear Regression:-
 $E = \sum_{i=1}^n (y_i - \hat{y}_i)^2$
 ↓
 Error, or loss func

and Here:-
 $E = e^T e = \sum_{i=1}^n (y_i - \hat{y}_i)^2 =$ error function or loss function.

 $\Rightarrow E = e^T e = (Y - \hat{Y})^T (Y - \hat{Y}) = (Y^T - \hat{Y}^T)(Y - \hat{Y})$
 $= (Y^T - (X\beta)^T)(Y - X\beta) \because \hat{Y} = X\beta$
 $= (Y^T - \beta^T X^T)(Y - X\beta) \rightsquigarrow$ All are matrices so dist. Prop holds.
 $= Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta$
 $= Y^T Y - Y^T X\beta - (X\beta)^T Y + (X\beta)^T X\beta$

Now: $Y^T X\beta = Y^T \hat{Y} = [y_1 \cdots y_n] \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = y_1 \hat{y}_1 + \cdots + y_n \hat{y}_n$

& $(X\beta)^T Y = \hat{Y}^T Y = [y_1 \cdots y_n] \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = y_1 \hat{y}_1 + \cdots + y_n \hat{y}_n$

so $Y^T X\beta = (X\beta)^T Y \rightsquigarrow$ or Basically we need to prove

$$\Rightarrow E = Y^T Y - 2Y^T X\beta + (Y^T X\beta)^T = Y^T X\beta \rightsquigarrow \text{as } Y^T X\beta \text{ is symmetric}$$

↳ Error or loss function

↳ $Y^T X\beta$ is 1×1 matrix hence it is symmetric



$E = \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X} \beta + \beta^T \mathbf{X}^T \mathbf{X} \beta \rightarrow$ we need to minimize this
choose β so that this gets
minimized.
same method (diff w.r.t β):-

$$\frac{dE}{d\beta} = \frac{d}{d\beta} (\mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X} \beta + \beta^T \mathbf{X}^T \mathbf{X} \beta) = 0$$

$$-2\mathbf{y}^T \mathbf{X} + \frac{d}{d\beta} \beta^T \mathbf{X}^T \mathbf{X} \beta = 0$$

$$-2\mathbf{y}^T \mathbf{X} + 2\mathbf{X}^T \mathbf{X} \beta = 0$$

$$\mathbf{y}^T \mathbf{X} = \mathbf{X}^T \mathbf{X} \beta$$

$$\beta = \mathbf{X}^{-1} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{y}^T \mathbf{X}$$

$$\Rightarrow \beta = (\mathbf{y}^T \mathbf{X})^T ((\mathbf{X}^T \mathbf{X})^{-1})^T$$

$$\beta = \underbrace{(\mathbf{X}^T \mathbf{y})}_{\substack{\text{Shape} \\ ((m+1), 1)}} \underbrace{(\mathbf{X}^T \mathbf{X})^{-1}}_{\substack{\text{Shape} \\ (m+1, m+1)}} \rightarrow \text{Now: } \quad \begin{array}{l} x = x_{\text{train}} \\ y = y_{\text{train}} \end{array}$$

Learn
 $\frac{d}{d\beta} \beta^T$

$$\frac{d}{dA} (A^T X A) \\ = 2X A^T$$

Why Gradient Descent?

skikit-learn uses two methods to solve linear regression problems

- OLS (closed form) → Here we find formulae the one we found above.
- (non-closed form) → Gradient Descent

The reason is The inverse operation used above to obtain β ; the $(\mathbf{X}^T \mathbf{X})^{-1}$ → The time/complexity of $n \times n$ computations using Gauss Jordan Elimination is $O(n^3)$ and so if we have 1000 cols then $T.C. = 1000^3 = 1000000000$

This many computations just to find the inverse. (This is take a lot of time & make the model slow)

Hence if we have:- text data or image based data (which have many cols) and we have to apply

Multivariable Linear Regression then we use Gradient Descent.

skikit Learn

OLS

GradientDescent

SGDRegressor class.

approximation
Technique.

Jupyter Notebook



Gradient Descent

Gradient Descent is a first order iterative optimization algorithm for finding a local minimum of a Differentiable fn

The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at a current point, because this is the direction of the steepest descent. Conversely, stepping in the direction of the gradient will lead to a local maximum of that function, this procedure is known as gradient ascent.

* Gradient Descent is a General Algorithm

↳ Linear Regression

↳ Logistic Regression

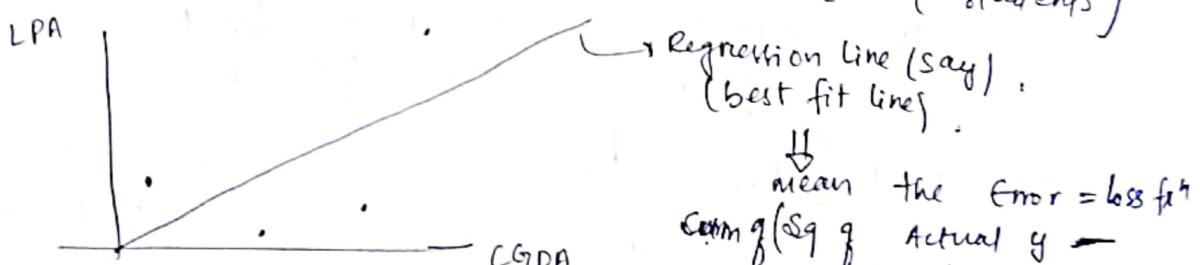
↳ T-SNE

↳ Backbone of Deep Learning.

Understanding Gradient Descent through the lenses of Linear Regression (But can be applied elsewhere also):-

Intuition:-

Consider a data set of 2 cols & 4 Rows (i.e. CGPA/LPA of 4 students)



To keep the discussion simple, suppose we know what m is (using closed form loss fn.) say, $m = 78.35$

then Loss fn. $\Rightarrow E(\theta) = \sum_{i=1}^n (y_i - 78.35x_i - b)^2$

Now:- we need to find b such that $E(b)$ is min.

mean the error = loss fn
error (sq of Actual y - predicted y) should be minimum.

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

$$\text{& } \hat{y}_i = mx_i + b$$

$$\Rightarrow E = \sum_{i=1}^n (y_i - mx_i - b)^2$$

$$\Rightarrow E(m, b) = \sum_{i=1}^n (y_i - mx_i - b)^2$$

The reln b/w f' and b is quadratic \therefore (in fact upward parabola)

- after 0.790°
of successive iterations
slope dec.
- This is the value of b where L is min.
- This is what we want.

Now we start with Gradient Descent

Step 1: select a Random b_{old} say $b = -10$

Now for this b ; L will have some value

& from this b we will have to reach the b_{min} using iterations.

Now at this point $b = -10$, the algo needs to move in such a way that the function value goes down. There are two directions to move; how would the algo know which direction to move so that the fn value goes down? This is slope.

The algo should move in the direction of b where the slope is decreasing, hence the algo should move in increasing direction of b for the above graph.

→ How to know:

find the slope of the curve at point $b = -10$
if slope \rightarrow +ve then use b
if slope is -ve then use b .

and $b_{\text{new}} = b_{\text{old}} - \text{slope}$ → This is the Gradient Descent eqn.

Note as we go closer & closer to actual b the slope goes to zero & hence the jump in b less.

This eqn will bring drastic changes in values of b ∵ we use iterations.

∴ Hence we use a transformed eqn:

$$b_{\text{new}} = b_{\text{old}} - \eta \text{slope}$$

where; η = Learning Rate

Generally its value is 0.01 (we can keep it 0.1 or 0.0001 etc.)

- Learning rate (η) should not be big (like 1); otherwise again drastic changes will occur & we might miss the actual sol
- and also we should not keep learning rate very small; then we move to zero very slowly.



The step is iterative, so when do we stop.

Page 16

We stop when $b_{\text{new}} - b_{\text{old}} = 0.0001$ or even less than that

that means we are converging to the actual b value

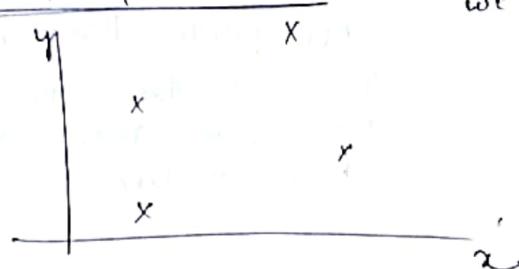
Actually: $b_{\text{new}} = b_{\text{old}} - \eta \text{slope}$

$\Rightarrow b_{\text{new}} - b_{\text{old}} \rightarrow -\eta \text{slope}$ which should be very small

means the slope is becoming 0.
to x-axis hence we are reaching our point of min.

Or: we can fix the no. of iterations (called epochs), ki hum after the ^{say 100 or 1000} iterations karnege $\&$ actual b will get after that is the approx value of b .
we will see how to find the optimum no. of iterations
so that we find the best possible value for b .

Mathematical formulation



we need to find the Regression line using Gradient Descent
say we already know $m = 78.35$
we need to find $b \rightarrow y\text{-intercept}$ (for simplicity)

Step 1: start with a random value of b

for i in epochs \rightarrow Iterations:-

$$b_{\text{new}} = b_{\text{old}} - \eta \times (\text{slope at } b_{\text{old}}), \quad \eta = 0.01 \rightarrow \text{Learning Rate}$$

$$b_{\text{old}} = b_{\text{new}}$$

How to find?

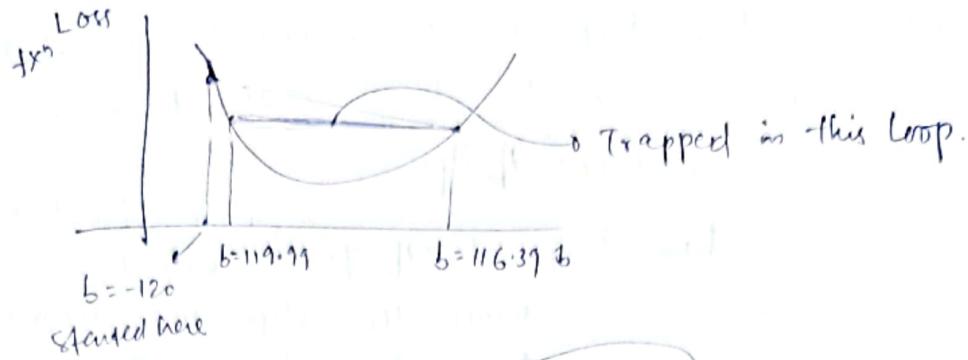
$$\begin{aligned} L &= \sum_{i=1}^n (y_i - mx_i - b)^2 \\ \frac{\partial L}{\partial b} &= \sum_{i=1}^n -2(y_i - mx_i - b) \end{aligned}$$

This is Gradient descent

Jupyter Notebook for upto this part.

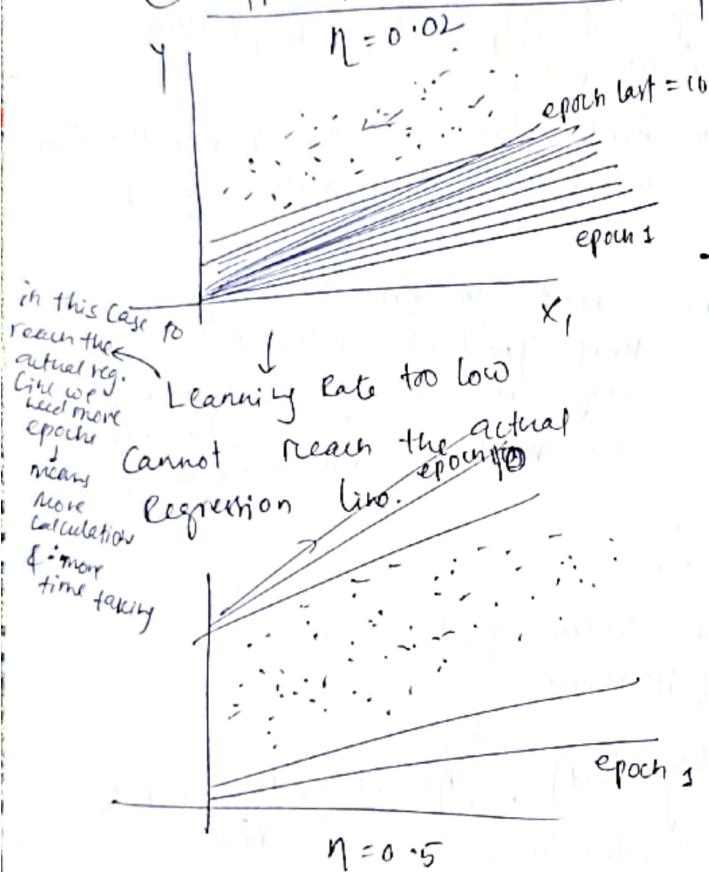
$$\text{slope at } b_{\text{old}} = \left. \frac{\partial L}{\partial b} \right|_{b=b_{\text{old}}}$$

Gradient Descent \rightarrow file 2 \rightarrow Drawing 1:



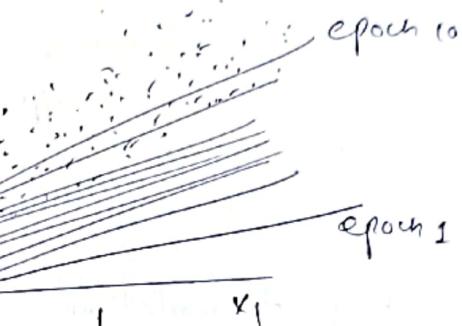
Few Discussions.

(1) Effect of Learning Rate



Learning rate too high, then we might jump high & miss the actual Regression line and we might get a weird/big number for intercept, b after epoch = 10.

say epoch = 10
 $\eta = 0.1$



Appropriate Learning Rate:- reached after in the epoch range, we reach the actual Regression line.

Learning Rate Dara pe depend karta hai. chal kar hi dekho hog. It is a hyperparameter. Sochi tuning of Learning Rate in Gradient Descent is very important.

(2) The universality of Gradient Descent

So far we studied Gradient Descent w.r.t linear Regression, but it can be applied along with other algorithms also.

In some other algo; the loss function may be different & the again we can use the Gradient descent similarly to find the parameters so that the loss fn is minimum. The only condition is that the loss fn should be differentiable; so that we can calculate the slope by differentiating.

Now calculating both m & b using Gradient Descent

i.e. Gradient descent for two variables

Step 1:- initialize random values for m & b

say $m=1$ & $b=0$

Step 2:- set epochs & learning-rate say epochs = 100 & $\eta=0.01$
then

for i in epochs:

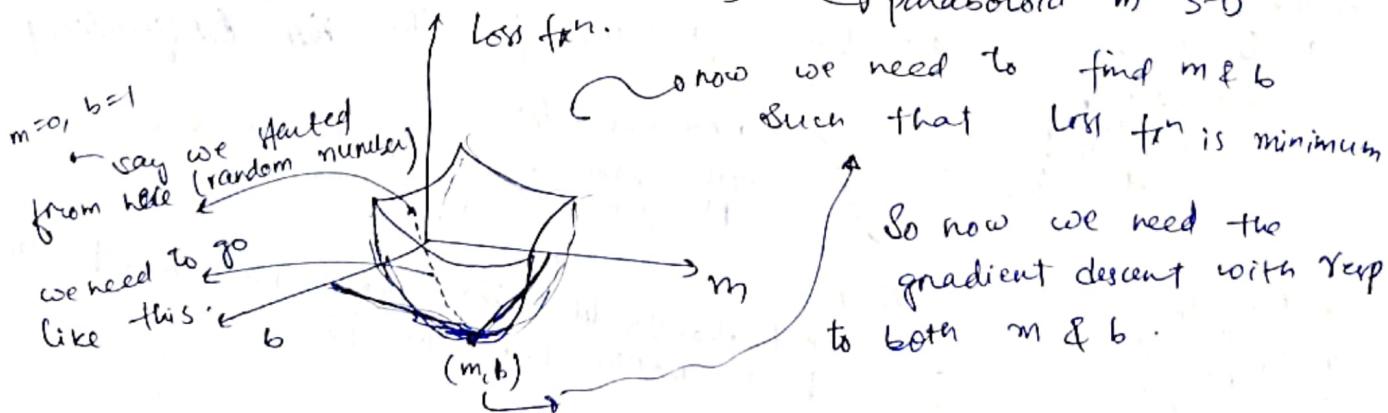
update b: $b = b - \eta \text{ slope}$ This slope we know, we saw previously

update m: $m = m - \eta \text{ slope}$

Loss fn:- $L = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ let's see what this slope is.

$$L(m, b) = \sum (y_i - mx_i - b)^2$$

paraboloid in 3-D



Now on this paraboloid at any point of the slope will be a tangent plane \rightarrow it will have two components; one component in direction of ∇f by diff other in direction of b .

¶ diff of Multivariable func is gradient. \therefore In the formulas.

$$b = b - \eta \text{ slope} \quad \text{this is slope wrt } b$$

$$\& m = m - \eta \text{ slope} \quad \text{ie. } \frac{\partial L}{\partial m} \Big|_{m=m_{old}}$$

$$L = \sum_{i=1}^n (y_i - mx_i - b)^2$$

$$\therefore \frac{\partial L}{\partial b} = -2 \sum_{i=1}^n (y_i - mx_i - b) \quad \begin{matrix} \text{slope of } L \\ \text{wrt } b \end{matrix}$$

$$\& \frac{\partial L}{\partial m} = -2 \sum_{i=1}^n (y_i - mx_i - b) x_i \quad \begin{matrix} \text{slope of } L \text{ wrt } m. \end{matrix}$$

$$\text{ie. } \frac{\partial L}{\partial m} \Big|_{m=m_{old}}$$

(m_{old}, b_{old}) is particular point jaha pe currently hai

Now same can be extended to higher dimensions \rightarrow dataset with more than one input col.

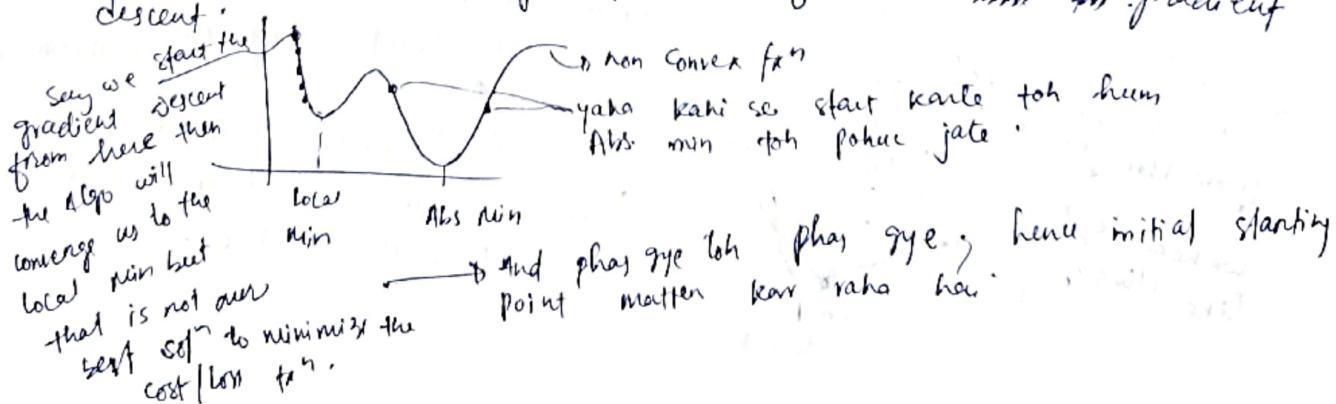
Fayyer NoteBook.

Such fns are good for gradient descent

Effect of loss fn on Gradient Descent:-

so far we had loss function: $L = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ \rightarrow is a convex loss fn.

But if we have nonconvex functions since they have only one minima (Global minima). then there can be more than 1 minima (local minima) but to minimize the loss fn we need to find the global minima. & local minima are big problems for the gradient descent.



In higher dimensions say we have 10 cols, we cannot check for convex hai ya nahi & agar initialisation agar aisa hua ki humara Algo local minimum pe converge kar jya then we have to accept it.

Lage bache ke kuch tankhe hote hai like tricks for initialization depending on cost function. (this we do in deep learning as wohi pe jyada non convex fcn milti hai).

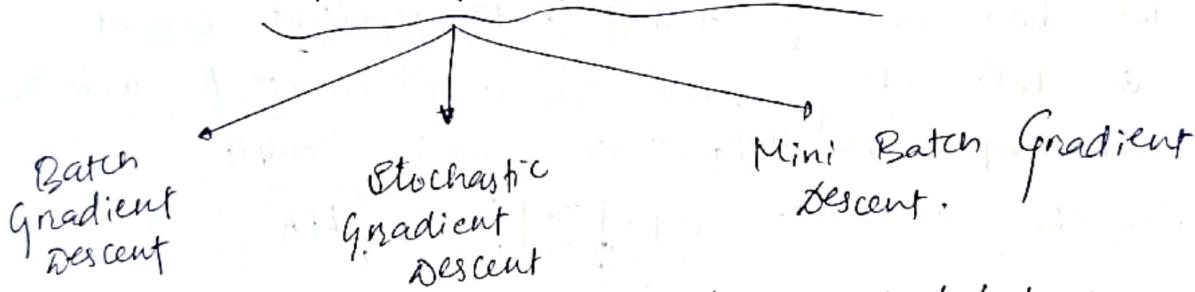
There is yet another problem if the loss/cost function has Saddle points (where the slope change karta hai & hence gradient nor pe Jane ke liye both jyada time/epochs lagega).

Effect of Data on Gradient Descent:

Agar features in the data are equally scaled then the contour plot of the cost fn/loss for ~~is~~ circular and we descent faster to the point of minima.

Hence for Linear Regression \rightarrow scale the input cols to the same scale.

Types of Gradient Descent



What is the basis of this Types/distinction:

we had the eqn:

$$m_{\text{new}} = m_{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial m} \Big|_{m=m_{\text{old}}} \right) \quad ; \quad L = \sum_{i=1}^n (y_i - mx_i - b)^2$$

$$b_{\text{new}} = b_{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial b} \Big|_{b=b_{\text{old}}} \right)$$

Now in Batch Gradient Descent:- To calculate the slope we use the entire data. & then make update in m & b. Hence so far we were applying the Batch Gradient Descent.

Hence Batch Gradient Descent is slow.

Now in stochastic Gradient Descent, we only consider one row of data at a time & make changes in $m \& b$ faster with large dataset but

in pure dataset per iteration there were 300 updates ^{there were 300 total rows/ dataset}. Again there is

Mini Batch Gradient Descent (A mix of Batch & stochastic Gradient descent).

In this we decide a batch size say 30, & there were 300 data originally; then the update is in $m \& b$ will be based on this 30 rows hence per iteration dataset pe ghumenge (300 rows) matlab 10 bar update karenge $m \& b$ ko.

Batch gradient descent is quite slow to use:-

→ It is used generally when we have convex loss function like in Linear Regression. But still there is one condition that the dataset should not be big.

Batch Gradient Descent :-

We have already studied Batch Gradient Descent.

So let's study Batch Gradient descent for more than one input variable/feature

Say the dataset is:-

LPA	GPA	IQ	Gender	LPA
1	2.5	120	M	1

Now the regression hyperplane will be:-

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \rightsquigarrow 3 \text{ dimensional}$$

So if we have n input col's:-

$$\text{then } Y = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

func we need to find

$$\beta_0, \beta_1, \dots, \beta_n$$

→ This will be the eqn of the regression hyperplane



mathematically formulating the n-Dim
Batch Gradient descent (Multiple Linear Regression)

To make it easier consider :-

rows=2	cols=(n+1)	CPT	x1	x2	LPA
		2.1	8.1	9.3	3.2
		7.5	9.5	9.5	3.5

we will extend this.

$$\rightarrow \text{eqn of Reg plane: } y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \rightarrow \text{to find } \{\beta_0, \beta_1, \beta_2\}$$

\downarrow LPA \downarrow CPT \downarrow TQ

Step 1: start with Random values of $\beta_0, \beta_1, \beta_2$

Generally: $\beta_0 = 0$ & slopes, $\beta_1, \beta_2 = 1$
 w/ intercept

Step 2: choose epochs = 100 & learning rate = 0.01 (say).

then updating the values for each epochs:

$$\beta_0^{\text{new}} = \beta_0^{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial \beta_0} \right)$$

$$\beta_1^{\text{new}} = \beta_1^{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial \beta_1} \right)$$

$$\beta_2^{\text{new}} = \beta_2^{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial \beta_2} \right)$$

Hence if we have n input cols then we will need n+1 such eqns with (n+1) Partial Derivatives.

Calculating: Mean squared error (MSE) $\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1}, \frac{\partial L}{\partial \beta_2}$ Lagrange $\frac{\partial}{\partial \beta_i}$ we are calculating Mean Squared Error (MSE) for our above data

$$\text{Loss fn} = L = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

to minimise karegi w/o with $\frac{1}{n}$ wale total error.

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

$$\text{we have: } \hat{y}_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

to minimise karegi.

$$L = \frac{1}{2} \sum_{i=1}^2 (y_i - \beta_0 - \beta_1 x_1 - \beta_2 x_2)^2$$

$$= \frac{1}{2} (3.2 - \beta_0 - \beta_1 \times 8.1 - \beta_2 \times 9.3)^2 + \frac{1}{2} (3.5 - \beta_0 - \beta_1 \times 7.5 - \beta_2 \times 9.5)^2$$

$$\text{so } \hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$$

$$\text{& } \hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22}$$

$$\text{so } L = \frac{1}{2} [(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2]$$

$$L = \frac{1}{2} [(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2]$$

Now:

$$\frac{\partial L}{\partial \beta_0} = -\frac{1}{2}(y_1 - \hat{y}_1) - \frac{1}{2}(y_2 - \hat{y}_2)$$

$$\therefore = -\frac{1}{2}[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2)]$$

for n rows

Note we are increasing the dimension not the dataset
the no. of input features is still 2.

$$\frac{\partial L}{\partial \beta_0} = -\frac{2}{n} [(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + \dots + (y_n - \hat{y}_n)]$$

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

Again if we have the same dataset: $L = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

$$\frac{\partial L}{\partial \beta_1} = -\frac{2}{2} [(y_1 - \hat{y}_1)x_{11} + (y_2 - \hat{y}_2)x_{21}] \quad L = \frac{1}{2} [(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2]$$

for n rows.

$$\frac{\partial L}{\partial \beta_1} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i)x_{i1} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i)x_{i1}$$

$\hookrightarrow x_{i1}$ is the 1st column i.e. Gpa col. (x_1)

$$\frac{\partial L}{\partial \beta_2} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i)x_{i2}$$

$\hookrightarrow x_{i2}$ is the 2nd col of the dataset i.e. IQ col (x_2)

so now if we have ($m+1$) input rows: (n rows)

then the hyperplane $y = \beta_0 + \beta_1 x_1 + \dots + \beta_m x_m$

and

$$\text{slope wrt } \beta_k = \frac{\partial L}{\partial \beta_k} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i)x_{ik}$$

~~General formula~~

Jupyter Notebook:

Stochastic Gradient Descent :- (most imp)

Page 50.

The problem in Batch Gradient Descent is that han coeff (of hyperplane of best fit) be respect mei loss $f_m^m(L)$ ka derivative nikalna hoga hoga has to make one update in the coeffs.

Again the loss fn is a summation over the rows.
Now suppose there are 1000 rows (which is very easy to
& cols = 5 (input cols)
and also say there is epochs = 50.

$$\therefore L = \sum_{i=1}^{1000} (y_i - \hat{y}_i)^2$$

∴ we have 5 input cols $\therefore y = \beta_0 + \beta_1 x_1 + \dots + \beta_5 x_5$
 \therefore to find $\frac{\partial L}{\partial \beta_0}$ \rightarrow we need to do 1000 derivatives
(open & diff)

slly for $\frac{\partial L}{\partial \beta_1}, \frac{\partial L}{\partial \beta_2}, \dots, \frac{\partial L}{\partial \beta_5}$

so there are 6000 derivatives in each iteration

\therefore for 50 epochs/iteration: there are

$$6000 \times 50 = 3,00,000 = 3 \text{ lakhs derivat. ver.}$$

Now if we have:- rows = 10^5 ; & cols = 10^2

& epochs = 10^3

then Total derivatives to be calculated

till the loop completes = 10^{10}

hence our Algo will become computation
very slow on Big Data.

Hence for Big Data sets \rightarrow we use stochastic grad. descent.

↑ like
in Deep Learning

+ CNN (on Images)

+ RNN (on Texts)

On Smaller Data set we can use Batch Grad descent.

Another problem in Batch Grad descent! -

In the class that we defined for Batch Grad descent we used vectorization (calculation of $y - \hat{y}$) for this X-train ko memory mei rakha hogा. Taki dot product ho payega lekin sometimes X-train may se huge

then the calculation of \hat{y} -hat will not be possible.

So there is a hardware problem (i.e. if the system RAM is small)

In Batch Grad descent :- After reading the entire data (X matrix) in a epoch we update the values of coefficients.

In Stochastic Grad descent :- we just read a single row & update then again see the next row - update of so on till last row for a epoch. Randomly selected (not sequentially). So in a single iteration / epoch → there are n iterations of n rows.

& because of those frequent updates we reach the correct answer in less epochs. Converges to mean Stochastic

Hence in stochastic Grad descent :-

- + faster convergence (since less # of epochs)
- + we don't have to load the entire data at a time hence no hardware problem.

Stochastic Meaning :- having a random prob. distribution that may be analysed statistically but may not be predicted precisely.

Problem with Stochastic GD is → if we run the algo on the same dataset we get different answers each time & all are closer to the actual answer - this is due to the random selection of rows.

Jupyter Notebook.

In Stochastic. we select a Random row say k^{th} row

Loss fn becomes :-

$$L = (y_k - \hat{y}_k)^2$$

$$\frac{\partial L}{\partial \beta_0} = -2(y_k - \hat{y}_k)$$

$$\frac{\partial L}{\partial \beta_j} = -2(y_k - \hat{y}_k) \cdot x_{kj}$$

Time Comparisons:-

say we use same epochs = 100 for both stochastic Grad. Descent.

then Batch GD will be faster than SGD.

Since in BGD there are only 100 updates to be made

but in SGD there are $100 \times \# \text{ of rows}$ updates to be made.

But since SGD will converge to a soln. before 100 epochs
 so we will use say 50 epochs in SGD
 & it will converge to almost same soln as BGD &
 hence # of epochs in SGD is less - in that way it's faster
 on big data sets.

When to use stochastic grad descent

- + Big data
- + when the loss function is Non Convex - since the non convex
- & due to this jumps in the SGD are random & we saw in the contour plot in stochastic GD.
 it may jump out of the local minima of the convex fn (The problem we faced with Batch GD).

Note: A problem with SGD is that due to Randomness even if the Algo reaches near the soln, it fluctuates & we need that as the Algo approaches the soln the fluctuation must decrease so that we are closer to the actual soln - for this we use a concept called

Learning Schedule.

Learning Schedule is a concept where we vary the learning rate with epochs.

Lower we... $t_0, t_1 = 5, 50$

we can define our own defn accordingly.

due to this epoch per; ith

$\therefore t_{\text{per}}$

$\therefore Lr$ b/w; for i in range(epoah):
 & hence fluctuation stabilizes towards end.

def learning_rate(t):

return $\frac{t}{t+t_1}$

Hence now Lr is a fn of epoch i

for j in range(X.shape[0]):
 & # of rows.

$Lr = \text{learning_rate}(i * X[\text{shape}[0]] + j)$
 multiply

The SGDRegressor() class provided by Sklearn has a parameter called learning_rate = 'constant', 'optimal', 'invscaling', 'adaptive' → to handle

if it is a comb of Batch & stochastic.
and mostly in Deep Learning Learning schedule.

Mini Batch Gradient Descent
suitable for small & large loss fn.

in Batch Grad Descent → 1 update in 1 epoch

in stochastic GD = #q rows update in 1 epoch

in MiniBatch GD = #q chosen update in 1 epoch

↳ Here we decide #q rows say = 100 → there are random batches 100 rows out of 1000 rows.
total rows = 1000 ∴ total batches = 100 of 100 rows.
∴ in each epoch 100 batches update ie 100 updates in 1 epoch.

if Batch size = #q rows → if becomes Batch GD

if batch size = 1 → " " Stochastic GD.

↳ Mini-Batch improves (Isa) the Randomness in the Stochastic GD.

Jupyter Note Book.

POLYNOMIAL REGRESSION

Linear Reg. is all about { We so far know :- $y = \beta_0 + \beta_1 x$ → Simple Linear Reg.
& $y = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$ → Multiple Linear Reg.
linear Data i.e. the data visualisation is kind of linear.

What if the scatter plot of the data is not linear?



→ in this case linear Regression will not help much & hence here we use polynomial Regression.

Say we have:- $\begin{array}{c|c} x & y \\ \hline 35 & 12 \end{array}$ & there is a quad. not w/ them
then we find $\begin{array}{c|c|c|c|c} x^0 & x^1 & x^2 & x^3 & y \\ \hline 1 & 35 & 35^2 & 35^3 & 12 \end{array}$ → & send this as x-train & y-train.

And the coeff we are finding is :-

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 \rightarrow (\beta_0, \beta_1, \beta_2)$$

Now finding this is same as Linear Regression Techniques.

If if degree = 3.
we will have:

x^0	x^1	x^2	x^3	y
1	35	35^2	35^3	12
:	:	:	:	:
:	:	:	:	:

eqn is :-

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

How to find which degree we need to impose:-
degree is a hyperparameter & we have to find $(\beta_0, \beta_1, \beta_2, \beta_3)$

If this value is less than the equation curve will underfit of data ki saani attributes/properties fetch na kar paye.
If this value is high than the regression curve will overfit (if tries to touch every point).

What if Dataset is :- $\begin{array}{c|c|c|c} x_1 & x_2 & y \\ \hline : & : & : \end{array}$ → Then
& degree = 2

So the dataset becomes:-

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2$$

$\begin{array}{c|c|c|c|c|c|c} x_1^0 & x_1^1 & x_1^2 & x_2^1 & x_2^2 & y \\ \hline : & : & : & : & : & : \end{array}$
Now consider this:- Copy just change the col names

$\begin{array}{c|c|c|c|c|c|c} x_0 & x_1 & x_2 & x_3 & x_4 & y \\ \hline : & : & : & : & : & : \end{array}$ → New dataset

$$y = \beta_0 + \beta_1 \bar{x}_1 + \beta_2 \bar{x}_2 + \beta_3 \bar{x}_3 + \beta_4 \bar{x}_4$$

Now this is equivalent to solving
Multiple Linear Regression

Jupyter Notebook.

slope na name

Page 52.

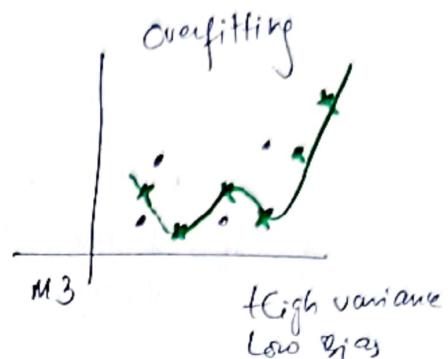
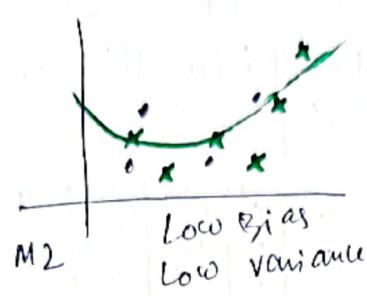
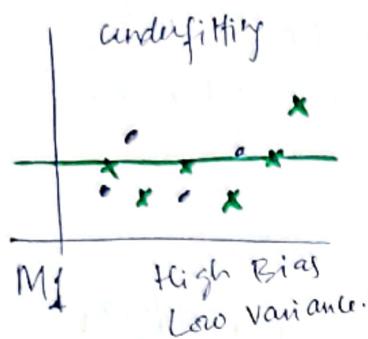
~~Star~~

Bias Variance Trade off

This concept is applicable
on every ML Algorithm.

Consider a data set P on it we apply three different
ML models:

X - training set
- test set.



Bias :- It is the inability of a ML model to truly capture the relationships in the training data.

i.e. the ML model training data pe hi accha perform nahi kar pa raha. Then we call it bias (since wo training data ko samajh ni nahi pa raha).

M3 → low bias

M2 → There is curve in the training data & the model is able to capture that curve.

M1 → high bias since it is not capturing the essence of the training data itself (the curved nature).

Variance:- It is the difference of fit on different data set.

Say we have training data set & test data & the model give error of 10 on training data & error of 100 on test data. → This means variance is $90/(100-10)$.

Model 1 & Model 2 (M1, M2) will have low variance since they are closer to the test data points, also hence error in train data - error in test data is not much.

In M3: the error in training data is almost 0 & error in test data set is high → : high variance.

Overshooting: When the model works good on the training data but does not work good on the test data. \Rightarrow Training error is extremely low but the test error is high \Rightarrow Variance is high.

Underfitting: Model jo training data pe hi accha kaam nahi kam paresha \rightarrow high bias.

Thus we need a model with Low Bias & Low Variance.

* Kuchhi kuchhi ke keep a bit bias so that variance less, we need to find the sweet spot b/w underfitting and overshooting and there are three ways to do it

- +
- Reguralization
- +
- Bagging
- +
- Boosting

we use this to reduce overshooting

Reguralization

\rightarrow It is a technique using added info in a ML model which you induce some

overshooting.

so that you can reduce

overshooting.

There are three kind of most used

Reguralization Techniques

\rightarrow Ridge Reg. or L₂ Reg.

\rightarrow Lasso Reg or L₁ Reg.

\rightarrow Elastic Net (Combination of above two)

Let's talk overshooting w.r.t Linear Regression :-

for one input & one output :- we are finding

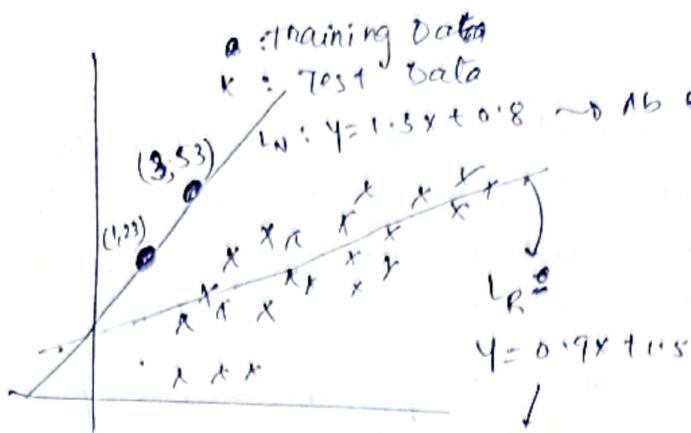
$$y = mx + b,$$

m = slope = weightage $g \times w$ finding.

in Linear regression if we have overshooting the the value of m will be extremely high. and if value of m is very low then it will be underfitting.

so Regularization for Linear regression means that slope ka value ko thora kam karne hai.

consider.



calculate points par
of Best fit nikalna
hai - koi undonosei
par manne-wala line
hi niklega.

and clearly ye
line actual data
ke essence ko miss
kar dega.
& it is a case of
overfitting.

consider this - the
hypothetical line jo training
data pe zah galti kaunsa best
test data pe better perform karoge.

ab humme ab kisi faise ML model ko
convince kouna shai ki do not choose L_N ; choose L_R .

How do we choose the line on training data set:

By minimizing the loss function/cost fnⁿ on the training set.

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

So to perform regularization we add something to the

loss fn:-

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

M is the slope

Now Let's see how we can convince our ML model to choose L_R instead of L_N .

Loss $L_N (\lambda=0)$	Loss $L_R (\lambda=1)$
$L = 0 + 1.5^2 \times 1$ $= 2.25$	$L = \sum_{i=1}^2 (y_i - \hat{y}_i)^2 + m^2$ $= (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (0.9x_1)^2$ $= (23 - 0.9 \times 1 - 1.5)^2 + (53 - 0.9 \times 3 - 1.5)^2 + 0.9^2$ $= 2.03$

since L_N pass hi
undon ke upar
koi raha hai

Hence now the ML model will choose the line as line of best fit even though it makes more mistakes on the training data since the loss of LR line is \leq loss of LN line.

\rightarrow ~~gives although Bias more than bad variance significantly than that made by.~~ And this is called Bias-Variance trade off.

Note: if there was two input cols of a output then:

$$n = \text{# of data in training dataset} \quad L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda(m_1^2 + m_2^2) \quad \begin{array}{l} \text{since sum} \\ \text{square each} \\ \text{multiply} \\ \text{lambda} \\ \text{here hence} \\ \text{it is L2 Norm} \\ \text{hence the name} \\ \text{L2 Regularization} \end{array}$$

Jupyter Note Book

Mathematical formulation of Ridge Regression & Building our own Ridge Regression class:-

Ridge Regression for 2D Data (n input col) (n rows)

for this case the Loss function is:-

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda m^2 \quad \rightarrow \text{to find } b \text{ from we want to reduce the slope}$$

$$\text{Now: } \frac{\partial L}{\partial b} = 0 \rightarrow b = \bar{y} - m \bar{x} \quad (\bar{y} = \text{mean } y, \bar{x} = \text{mean } x)$$

$$\begin{aligned} \text{Again: } \frac{\partial L}{\partial m} &= \frac{\partial}{\partial m} \sum_{i=1}^n (y_i - mx_i - b)^2 + \lambda m^2 \\ &= \frac{\partial}{\partial m} \sum_{i=1}^n (y_i - mx_i - \bar{y} + m \bar{x})^2 + \lambda m^2 \\ &= \sum_{i=1}^n 2(y_i - mx_i - \bar{y} + m \bar{x})(-x_i + \bar{x}) + 2\lambda m = 0 \\ &= -2 \sum_{i=1}^n (y_i - mx_i - \bar{y} + m \bar{x})(x_i - \bar{x}) + 2\lambda m = 0 \end{aligned}$$

- minimum data

$$\Rightarrow \lambda m - \sum_{i=1}^n [(y_i - \bar{y}) - m(x_i - \bar{x})] (x_i - \bar{x}) = 0$$

$$\Rightarrow \lambda m - \sum_{i=1}^n ((y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2) = 0$$

$$\Rightarrow \lambda m - \sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) + m \sum_{i=1}^n (x_i - \bar{x})^2 = 0$$

$$\Rightarrow \left(\lambda + \sum_{i=1}^n (x_i - \bar{x})^2 \right) m = \sum_{i=1}^n (y_i - \bar{y}) + (x_i - \bar{x})$$

$$m = \frac{\sum_{i=1}^n (y_i - \bar{y}) + (x_i - \bar{x})}{\lambda + \sum_{i=1}^n (x_i - \bar{x})^2}$$

This is the new
 m : note
that $\lambda > 0$

$$\text{earlier } m = \frac{\sum_{i=1}^n (y_i - \bar{y}) + (x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2} ; \text{ hence}$$

Since now we are adding $\lambda > 0$ to deno $\therefore m$ yes
 \because bias \uparrow but variance \downarrow & we control overfitting.
Hence : if $\lambda = 0$, then we are performing Linear
Regression itself & if $\lambda > 0$ then we will be
performing Ridge Regression.

Code for this Jupyter Notebook.

Ridge Regression for ND data :

Dataset : $X_1 | X_2 | \dots | X_n | Y \sim (m+1) \text{ cols} \& m \text{ rows}$

Loss fn in this case :

We learnt earlier that

$L = \sum_{i=1}^m (y_i - \hat{y}_i)^2$ can be written as

$$L = (Xw - Y)^T (Xw - Y) ; \text{ where } Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \& X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}$$

$y = w_0 + w_1 x_1 + \dots + w_n x_n$
So we need to
find (w_0, \dots, w_n)
 $w_0 \rightarrow \text{intercept}$
 $w_i \rightarrow \text{slope/weights}$



And the Loss fn for the Ridge Regression
(by adding a regularization term) :-

Page 55

$$L = (xw - y)^T (xw - y) + \lambda \|w\|^2$$

↳ $\|w\|^2 = w_0^2 + w_1^2 + \dots + w_n^2$

$$L = (xw - y)^T (xw - y) + \lambda w^T w$$

↳ This is the Loss fn for Ridge Reg for n Dim Data.

We need to find W matrix:

$$L = ((xw)^T - y^T)(xw - y) + \lambda w^T w$$

$$L = (w^T x^T - y^T)(xw - y) + \lambda w^T w$$

$$L = w^T x^T x w - w^T x^T y - y^T x w + y^T y + \lambda w^T w$$

↳ These two terms are the same (did earlier)

Now:

$$L = w^T x^T x w - 2y^T x w + y^T y + \lambda w^T w$$

$$\text{or } L = w^T x^T x w - 2w^T x^T y + y^T y + \lambda w^T w$$

Now $\frac{dL}{dw} = 2x^T x w - \cancel{2x^T y} + 2\lambda w = 0$

$$w(x^T x + \lambda I) = x^T y, \quad [I_{n \times n, n \times 1}]$$

$$w = \frac{x^T y}{\lambda I + x^T x}$$

$$\Rightarrow w = (x^T x + \lambda I)^{-1} x^T y$$

Some common vector derivatives:-

$$x^T B \xrightarrow{\frac{\partial}{\partial x}} B$$

$$x^T b \rightarrow b$$

$$x^T x \rightarrow 2x$$

$$x^T B x \rightarrow 2Bx$$

code for this part

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

Jupyter Notebook.

earlier it was:-



Now:: what if the date set is huge & we have multiple input cols \rightarrow Then calculation for the inverse will take a lot of time & hence we'll choose

Ridge Regression Technique using Gradient Descent

So for multiple input cols (say n of rows = m)

Loss fn with added regularization term

choose starting w

$$L = (Xw - y)^T (Xw - y) + \lambda \|w\|^2 \quad w = [0, 1, \dots, 1] \text{ for gradient descent.}$$

$$L = (Xw - y)^T (Xw - y) + \lambda w^T w, \text{ where the terms have the same meanings as discussed in previous section}$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}; \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}; \quad w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \quad \begin{array}{l} w_0 \text{ - intercept} \\ w_1 \dots \text{slopes.} \end{array}$$

Now we need to update the slopes & intercept values in Gradient Descent using:-

$$w_0 = w_0 - \eta \frac{\partial L}{\partial w_0}, \quad \dots, \quad w_i = w_i - \eta \frac{\partial L}{\partial w_i}, \quad \dots, \quad w_n = w_n - \eta \frac{\partial L}{\partial w_n}$$

\downarrow in a single formula

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\Delta L}{\Delta w} = w_{\text{old}} - \eta \text{ grad } L$$

$$\begin{bmatrix} w_{0\text{new}} & \dots & w_{n\text{new}} \end{bmatrix} = w_{\text{old}} - \eta \nabla L = \bullet \begin{bmatrix} w_0 & \dots & w_n \end{bmatrix} - \eta \left[\frac{\partial L}{\partial w_0}, \dots, \frac{\partial L}{\partial w_n} \right]$$

$$\text{we have: } L = \frac{1}{2} (Xw - y)^T (Xw - y) + \frac{1}{2} \lambda w^T w$$

$$L = \frac{1}{2} \left[w^T X^T X w - 2 w^T X^T y + y^T y \right] + \frac{1}{2} \lambda w^T w$$

$$\frac{\partial L}{\partial w} = \frac{1}{2} \left[2 X^T X w - 2 X^T y \right] + \frac{1}{2} 2 \lambda w$$

$$\frac{\partial L}{\partial w} = X^T X w - X^T y + \lambda w \rightarrow \text{or } \left[\frac{\partial L}{\partial w} \right]$$

choose start $w = [0, 1, 1, \dots, 1]$, then in each epoch we update

Jupyter Notebook

$$w = w - \eta \frac{\partial L}{\partial w}$$

$\frac{1}{2}$ multiply
kiya for
mathematical
convenience
 \downarrow
minimization
karen ha.



5 Key Points in Ridge Regression

this is also called shrinkage coeff. Page 56
 $\lambda \|w\|^2 = \lambda w^T w$

In Ridge Regression we add a regularization term to the loss fn so that the values of slopes have bias & variance tradeoff.

1/ How the coefficients get affected in Ridge Regression?

$\lambda > 0$ then if $\lambda = 0$ then we are using Simple Linear Regression.

if $\lambda > 0$ and as we \uparrow λ then the coefficients (coefficients i.e. slopes) starts moving towards '0' but never '0'.
see jupyter Notebook.

2/ Higher coefficients are affected more.

say.

x_1	x_2	x_3	y
1	1	1	1
1	1	1	1
1	1	1	1

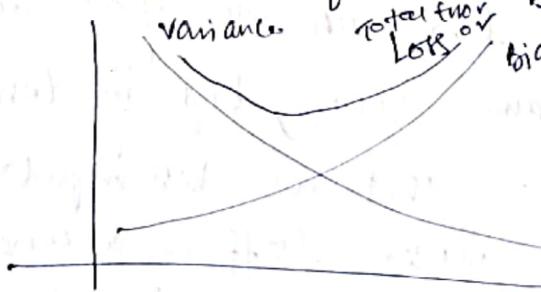
 → this means they calculate Kappa to like x_1 is most imp. then x_2 & then coeff $w_1=1000$ $w_2=10$ $w_3=1$. x_3 is least important.

Now as we \uparrow λ then w_1 will be affected the most (less faster) & w_3 is the least affected (less slowly) regularization on

3/ Effect of Bias Variance Tradeoff:-

As λ \uparrow the Bias \uparrow but the variance \downarrow but if we continue \uparrow it then Bias will \uparrow but the curve may underfit ~~overfit~~.

if λ is very small then Bias \downarrow & Variance \uparrow if there is overfitting.



We need to choose the intersection point of Bias & variance for our optimal λ . → Then both bias & variance are least.

4. Effect of Regularization on Loss fn.

The Loss fn:-

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|w\|^2$$

this if we divide by n then it becomes,

mean squared error (MSE).

$$\text{Loss} \propto \lambda \|w\|^2$$

Loss is pos⁺, hence the loss

fn. goes up upon regularization.

- * Mostly Ridge Reg. fails apply Karna chahiye when # of input cols is ≥ 2 .

Lasso Regression or LS Regularization \rightarrow It also helps in reducing the overfitting.

in Ridge Regression: we added a Reg. term

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|w\|_1; \|w\|_1 = w_1 + \dots + w_n$$

& in Lasso also we do the same. $w_i \rightarrow$ coefficients.

here we just use L1 Norm:-

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|w\|_1; \text{ where: } \|w\|_1 = |w_1| + |w_2| + \dots + |w_n|$$

Jupyter Notebook - using scikit Learn.

as $\lambda \uparrow$ we try to min $\|w\|_1 \rightarrow$ & decreasing too much may cause underfitting.

In Ridge Reg. \rightarrow no matter how much we used alpha/lambda, the coeff never became zero, but in Lasso, as we increase lambda, the x_i 's that are less important to determine y (output) make coeff becomes zero.

\hookrightarrow Hence Lasso regression inherently performs feature selection & helps in dimensionality reduction.

Hence Lasso is preferred over Ridge for high dimensional data.

Why Lasso Regression creates sparsity?

we see as the value of λ goes then the coefficient (slopes) in case of Lasso regression becomes zero \rightarrow thus creates sparsity.

As the λ value becomes very high \rightarrow all the coefficients become zero \rightarrow how does this happen?

Consider one input col & one output col to understand:-

$$\text{X/Y} \rightarrow \text{Linear Regression case:- we have to find the line}$$

$$\& b = \bar{y} - m\bar{x} \quad \text{we know.} \quad \bar{y} = mx + b$$

$$\& m = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$\bar{y} = \text{mean of } y$
 $\bar{x} = \text{mean of } x$.

These were the formula for Simple Linear Regression.

f for case of Ridge Regression

$$b = \bar{y} - m\bar{x}$$

$$\& m = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{1 + \sum_{i=1}^n (x_i - \bar{x})^2}$$

Now let's see what the formula becomes for the Lasso Regression:-

$$b = \bar{y} - m\bar{x} \quad (\text{same})$$

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda |m|$$

$$L = \sum_{i=1}^n (y_i - mx_i - b)^2 + \lambda |m| = \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 + \lambda |m|$$

$m > 0$

$$L = \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 + 2\lambda m$$

$m < 0$

$$L = \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 - 2\lambda m$$

This 2 is added just for mathematical conv.

~~Because~~ since ye modulus hai toh iska direct diff nahi kar sake, we have to break it.

$m > 0$:

$$L = \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 + 2\lambda m$$

$$\frac{\partial L}{\partial m} = \sum_{i=1}^n 2(y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) + 2\lambda = 0$$

$$\Rightarrow \lambda - \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})(\bar{x}_i - \bar{x}) = 0$$

$$\Rightarrow \lambda - \sum_{i=1}^n ((y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2) = 0$$

$$\therefore \lambda = \sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) - \sum_{i=1}^n m(x_i - \bar{x})^2$$

$$\Rightarrow m = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) - \lambda}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

This is fixed

clearly as $\lambda \uparrow \uparrow$
the value of m
decreases. If we can
use $\lambda = 8 \cdot t \cdot m = 0$
if we further
use t then
 m gets neg.
value!

if $m = 0$: then we'll have Simple Linear Reg f

$$\text{then } m = \frac{\sum (y_i - \bar{y})(x_i - \bar{x})}{\sum (x_i - \bar{x})^2}$$

& if $m < 0$:

$$m = \frac{\sum (y_i - \bar{y})(x_i - \bar{x}) + \lambda}{\sum (x_i - \bar{x})^2}$$

$$\text{say: } \frac{m > 0}{m = \frac{yx - b}{x^2}}$$

$$\text{if } \frac{m < 0}{m = \frac{yx + b}{x^2}}$$

$$\text{say } yx = 100 \text{ & } x^2 = 50$$

$$\lambda = 0 \rightarrow \text{we get } m = 2$$

$$m = \frac{100 - b}{50}$$

as λ goes m goes & as $\lambda = 100$

$$m = \frac{150 - b}{50}$$

& if $\lambda > 100$ then m

gets -ve but as m gets negative

this formula is used

New slope increased even more than the simple linear ($\lambda = 0$) Reg case for

hence the algorithm stops increasing or using m after $m = 0$.



Nao if $m < 0$

$$m = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) + \lambda}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{yx + \lambda}{x^2}$$

Page 58

$\therefore \lambda > 0 \therefore$ for $m < 0$

$$m = \frac{-100 + \lambda}{50} \rightarrow \text{Nao as } \lambda \text{ goes}$$

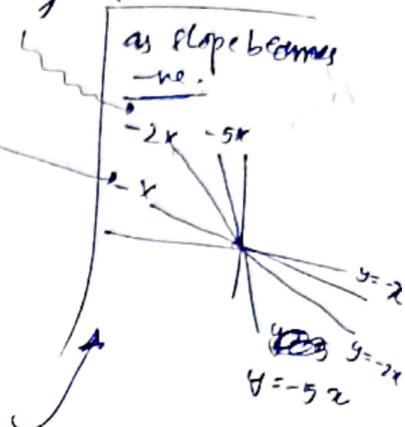
i say $y\bar{x} = -100$ & $x^2 = 50$
 m goes (simple linear case: $\lambda = 0$
 $m = -2$)

$$\lambda = 50 \rightarrow m = -1$$

$$\lambda = 100 \rightarrow m = 0$$

$\lambda = 150 > |100|/|yx|$ then m becomes pos but in that
case $m > 0$ formula will be used.

$$m = \frac{yx - \lambda}{x^2} = \frac{-100 - 150}{50} = -5 \rightarrow \text{The line becomes even worse than before}$$



and hence after $m = 0$ we stop the algorithm.

Again: Ridge Reg. mei spansify ($w_{eff} = 0$) nahi hota since.
waha lambda (λ) deno mei divide ho raha hota

hai; agar num mei add ya subtract nahi.

Hence: λ kitha bhi bada kardo. w_{eff} zero nahi hoga,
ekuch na such value aya hi; Ridge mei w_{eff}
tabhi zero hoga jab num = 0. (num mei λ hai hi nahi)

ELASTIC NET REGRESSION

It is a combination of Ridge & Lasso Regression.

Lasso is used when we are sure that there are some input features that are not useful in output prediction but what if there are many input cols & we are not sure if there are input features which are not useful in output prediction \rightarrow then we use Elastic Net Regression.

The formula for Loss fn in Elastic Net Regression is:

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \cdot 0.5 \times \|w\|^2 + \beta \|w\|$$

in scikit learn we get two hyper parameters:-

$$\lambda = a + b \quad \& \quad \text{l1-ratio} = \frac{b}{a+b}$$

By default: $\lambda = a + b = 1$ & $(l1_ratio = 0.5)$
 $\therefore a = 0.5, b = 0.5$

But say $l1_ratio = 0.9 \rightarrow 90\%$ we use Lasso Reg &
10% Ridge Regg.

The scikit learn with $(\lambda, l1_ratio) \rightarrow$ is used in
the elastic net Reg., kitha y. Ridge Lagage of kitha y.
Lasso Lagage.

Apart from big Data sets ; when the input col has multi-
collinearity hota hai (input cols apas mei bahot jyada depend karte hain)
then we use elastic net Regression.

Jupyter Notebook

t1: weight height

Now we will do a simple regression problem.
We have a dataset of house prices which contains
the following columns: (a) area (b) bedrooms
(c) distance from center (d) age (e) sqft_sqft
(f) tax rate (g) price. We want to predict
the price of a house based on its area, distance
from center, number of bedrooms, age and tax
rate. We will use linear regression to solve this
problem.

We will start by importing the required libraries:
First we will load the data and then we will use
the following steps: (a) Data Cleaning (b) Feature
Engineering (c) Model Selection (d) Model Training
(e) Model Evaluation (f) Model Deployment
We will follow these steps to build a machine
learning model for house price prediction.



Logistic Regression

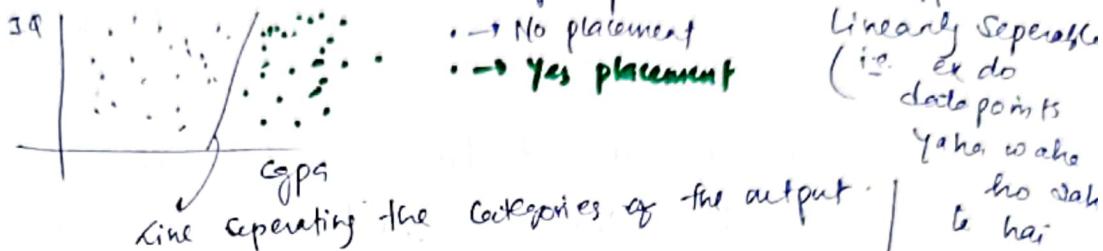
Page 59.

There are two perceptions of Logistic Regression

- ⊕ Geometric (Learn yourself).
- ⊕ probabilistic. (we learn this)

On what type of dataset should we apply Logistic Regression?

L+ dataset which are linearly separable or almost



Perception Trick

Consider the same above dataset if we have to predict for a given new child ki uska placement kya nahi.

The data set is linearly separable & Eqⁿ of line used in Logistic Regression is $Ax + By + C = 0$ (General Eqⁿ)

This is the line that separates the dataset.

$$\begin{aligned} Ax_1 + Bx_2 + C &= 0 \\ \text{GPA } &\downarrow \\ \text{SAT } &\downarrow \end{aligned}$$

Hence to find A, B & C.

we start with a random line; random value of A, B & C say A = 1, B = 1, C = 0

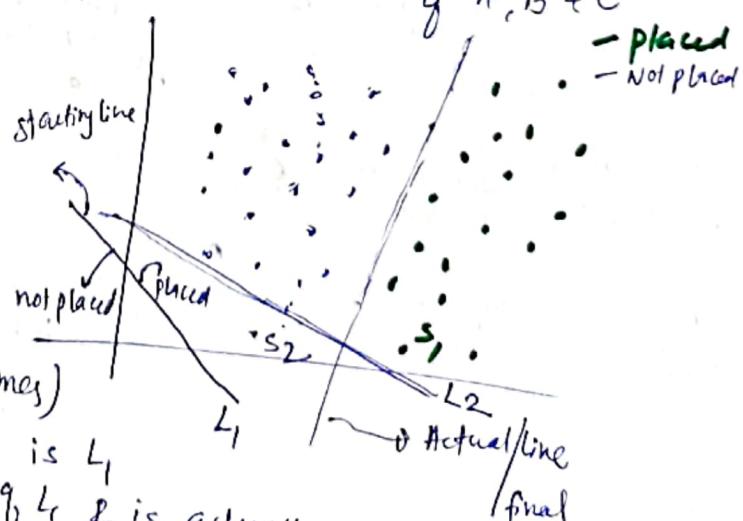
& for this line we consider one side all placed & other side not placed (see diag) so according to this line everyone is placed.

Now we run a loop (nearly 1000 times) we take a point say s_1 & ask is L_1

good & s_1 is on placed side of L_1

Placed & s_1 say L_1 is ok. so L_1 & is actually

value of A, B, C; Again in 2nd so for s_1 we don't change the iteration of loop we pick s_2



if ask s_2 : is L, OK \rightarrow it says well i am not placed but
 L, say I am placed so L is not OK for me. so we
 now make transformations in A, B, C & change L_1 in
 such a way that s_2 becomes not placed. (see diag)
 if Now we have a new Line L_2 & we proceed
 in the loop till the Last iteration & we finally get our
 actual Answer/line. or we can also run the loop till
 convergence. (i.e. No. of mis classified
 points = 0)

1st on: for the initial line, how do we decide which side is green & which side blue (not placed) (placed)

How to decide pos region & -ve Region.

say initial line is :- $2x + 3y + 5 = 0$

the pos region is :- $2x + 3y + 5 > 0$

& -ve region is :- $2x + 3y + 5 < 0$.

using this we decide if a point is in pos region or a -ve region.

Transformations: (wrt a point $\xrightarrow{\text{jis pt retaun}}$ more karna hai)

Line is :- $Ax + By + C = 0$ $\xrightarrow{\text{wrt C}}$ we shift line down
 change in C \rightarrow parallel shifts $\xrightarrow{\text{wrt C}}$ " " " up
 change in A \rightarrow rotation about y-intercept
 change in B \rightarrow rotation about x-intercept

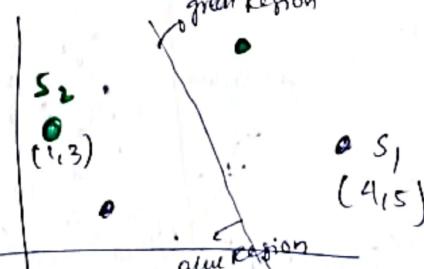
Consider.

Jab line ko -ve direction mei more karna hai then add karo

Again. s_2 says the initial line is faulty:-
so transforming:-

$$\begin{array}{r} 2 \ 3 \ 5 \\ + 1 \ 3 \ 1 \\ \hline 3x + 6y + 7 = 0 \end{array}$$

\hookrightarrow for this line s_2 will be on pos/green side.



say $2x + 3y + 5 = 0$

s_1 says line is not good
so we move line forwards

$s_1 \rightarrow (4, 5, 1)$

$$\begin{array}{r} 2 \ 3 \ 5 \\ - 4 \ 5 \ 1 \\ \hline -2x - 2y + 9 = 0 \end{array}$$

for this line

s_1 will be in blue region (-ve. Region)

Jab line ko
pos direction
mei more karo
mai tab subtract
karo-

$Ax + By + C = 0 \rightarrow$ original line
Thus summary of transformation is:- current

Page 60

Agar apka pot point (x_1, y_1) -ve side mei hai then
we need to move the line to -ve side to make (x, y)
on pot side for that new line:-

$$\begin{matrix} A & B & C \\ + & x_1 & y_1 & 1 \end{matrix}$$

& if \bullet -ve point is on pot side then
the line to pot side to make the we need to move $= 0$
Transformation is:- $\begin{matrix} A & B & C \\ - & x_1 & y_1 & 1 \end{matrix}$
 $\begin{matrix} (A+x_1)x + (B+y_1)y + (C+1) \\ (A-x_1)x + (B-y_1)y + (C-1) = 0 \end{matrix}$

But these transformations are huge hence we use η = learning rate
 $\eta = 0.01$ or 0.1 generally & Transformations then become:

$\begin{matrix} \pm \eta x_1 & \pm \eta y_1 & \eta \\ \hline (A \pm \eta x_1) + (B \pm \eta y_1) + (C \pm \eta) = 0 \end{matrix}$ with this the change / transformation
is not abrupt \rightarrow but it still does the job.

Algorithm for perceptron trick:-

S _i	Cgpa	Iq	placed
1	7.5	81	1
2	8.9	109	1
3	7.0	81	0

so The eqn of line will be:- $Ax + by + c = 0$

for this Data :- $w_0 + w_1 x_1 + w_2 x_2 = 0$

Cgpa Iq

$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$ (assuming the a col X₀ of all ones)

How does it make predictions:- $\sum_{i=0}^2 w_i x_i = 0$ \rightarrow General eqn of line.

say for S₁: (we have w_0, w_1, w_2)

This is our model.

$$w_0 x_0 + w_1 x_1 + w_2 x_2$$

$\rightarrow 0$ means placed

$\rightarrow 0$ means not placed.

$$(w_{eff} = (w_0, w_1, w_2)) \cdot ((1, 7.5, 81) = (1, \text{cgpa}, \text{iq}) = (1, x_1, x_2)) \rightarrow 0 \text{ places}$$

dot product

$$[w_0 \ w_1 \ w_2] \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$\rightarrow 0$ Not placed



Algorithm is:

Epochs = 1000 generally ; $\eta = 0.01$ (generally) Learning rate

for i in range (epoch) :

randomly select a student i.e. student

if $x_i \in$ Negative Region and $\sum_{i=0}^2 w_i x_i > 0$ Actually belongs to +ve Region

placement nahi but model is wrong

new but model nahi but model is wrong

new $w_{\text{new}} = w_{\text{old}} - \eta x_i$ but model is wrong

$$= [w_0, w_1, w_2] - \eta [x_0, x_1, x_2]$$

This is 1

placement nahi but model nahi but model is wrong

new $w_{\text{new}} = w_{\text{old}} + \eta x_i$ but model is wrong

$$= [w_0, w_1, w_2] + \eta [x_0, x_1, x_2]$$

Simplifying the Algo

for i in range (epoch) without any if else

select a random student:

update: $w_{\text{new}} = w_{\text{old}} + \eta (y_i - \hat{y}_i) x_i$

Predicted output actual placement but model ne bhi bola

Why it works. placement nahi but model ne bhi bola

Actual output placement nahi but model ne bhi bola

y_i placement nahi but model ne bhi bola

\hat{y}_i placement nahi but model ne bhi bola

$y_i - \hat{y}_i$ placement nahi but model ne bhi bola

0 $\rightarrow w_{\text{new}} = w_{\text{old}}$ (✓)

0 $\rightarrow w_{\text{new}} = w_{\text{old}}$ (✓)

1 $\rightarrow w_{\text{new}} = w_{\text{old}} + \eta x_i$ placement nahi but model ne bhi bola

-1 $\rightarrow w_{\text{new}} = w_{\text{old}} - \eta x_i$ placement nahi but model ne bhi bola

Jupyter Notebook.

So far we had perceptron trick :-

using $b_{in} = \text{Wt} \cdot (\mathbf{y}_i \cdot \mathbf{x}_i)$, but we saw that Logistic Regression performed better than perceptron trick; so we have to improve the algorithm.

In perceptron trick :- the line was transformed only if the point under observation was misclassified.
the ok points did nothing to the line.
by changing this approach.

Now both correctly classified & misclassified points will

ye points kine ko
apne paas bulayenge
(as always). ↴

transform the line
ye line ko apne ki line
bulayenge

isse kya hoga ki :- job same points
classify hojenge & line bias hai then good points
line ko clear bulayenge & agar jyada clear bulayenge
toh desire wala ^{classified} points phir is clear bulayenge (\because all are classified)
& hence there will be an equilibrium.

Note :- The magnitude of push & pull of the line by a point will also depend on the distance b/w the pt. & line.

Agan point misclassified hai then the pull of the ~~point~~ point
on the line is proportional to the distance b/w the pt & line
jyada distance \rightarrow jyada pull
kam distance \rightarrow kam pull.

for correctly classified points:- then the push of the point on
the line is ^v proportional to the dist. b/w pt & line
intensity

jyada distance \rightarrow kam push
kam distance \rightarrow jyada push.

↓
This is logical also.

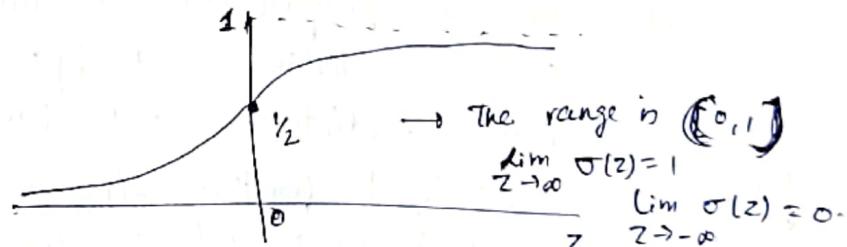
This is the change that Skunk Learn does in
Logistic Regression class.

To achieve this :- we need to make some changes
in $W_{new} = W_{old} - \eta (y_i - \hat{y}_i) x_i$ → we discussed this eqn & the meaning of each term.

Hamara dono cases
mei thi change karne
hai,
ie. in cases mei nahi
 $W_{new} = W_{old}$
chahiye ie.
 $\eta (y_i - \hat{y}_i) x_i$ kyo? kyu?
nahi move dena hoga
 $y_i - \hat{y}_i \neq 0$ move chahiye
pehle ham predicted
placed & not placed ko
represent karne ke liye
the ab kuch aur koi rep.
kam change hoga.
Now $y_i \rightarrow 0$ values leta hai
f if \hat{y}_i bhi 0,1 value lega toh
hence target is ki \hat{y}_i 0,1 ke alawa kuch aur values
le. \hat{y}_i ka calculation mechanism change karne hoga.
hence \hat{y}_i ka calculation mechanism change karne hoga.
↳ in the perception trick we were using step function
which gave 0,1 to find \hat{y}_i → Now we need to
use something else function to find \hat{y}_i instead of step func.
The function used is:- Sigmoid function:-

The Sigmoid fn.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Now to calculate \hat{y}_i hoga $= \hat{y}_i = w_1 x_1 + w_2 x_2 + w_0 > 0 \rightarrow 1$
Now we replace the step fn with sigmoid fn

$$\hat{y}_i = \text{sigmoid}(w_1 x_1 + w_2 x_2 + w_0)$$

Domain $(-\infty, \infty)$

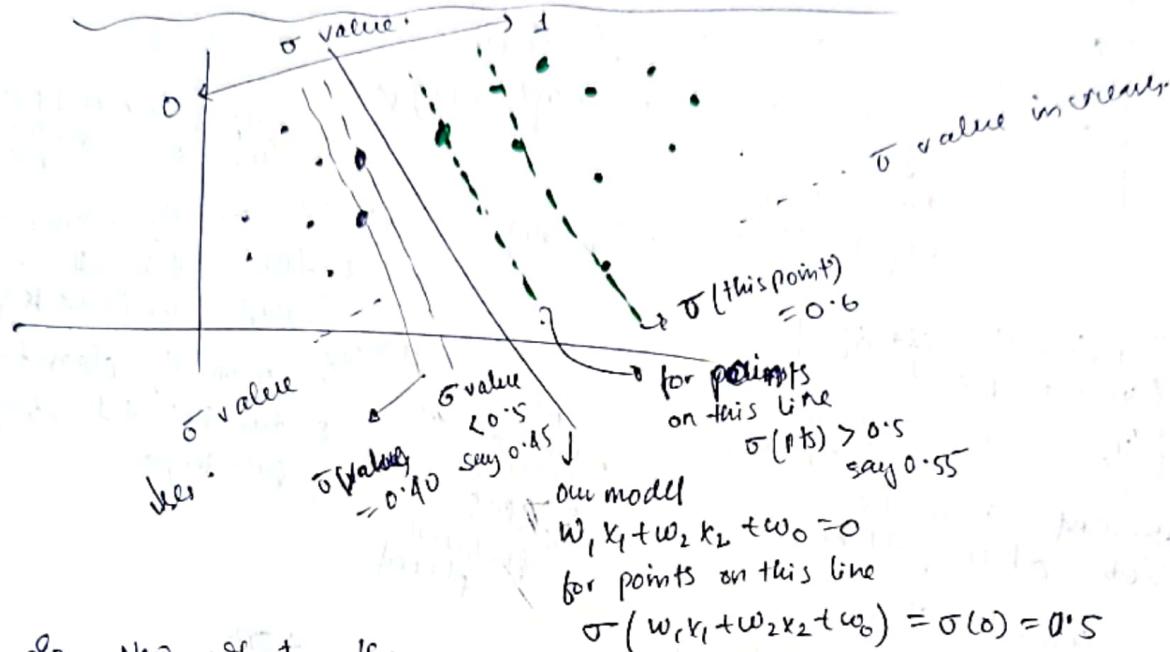
$T > 0 \rightarrow \hat{y}_i > 0.5$ step fn
 $T < 0 \rightarrow \hat{y}_i < 0.5$

So given input:- $\{x_1, x_2\} \rightarrow \{w_0, w_1, w_2\}$ placement hogya nahi?

Page 62.

$$\text{then } T = x_1 w_1 + x_2 w_2 + w_0 \quad \begin{cases} > 0 \\ \leq 0 \end{cases} \quad \begin{cases} \text{then } \sigma(T) > 0.5 \rightarrow \text{placed} \\ \text{then } \sigma(T) \leq 0.5 \rightarrow \text{not placed.} \end{cases}$$

Let's see how the Sigmoid fn works.



So now what the Sigmoid fn represent is probability dist of placed on a line say σ value = 0.5 means $P(\text{placed pts. on that line}) = 0.5$.

if σ value = 0.6 $\rightarrow P(\text{pts on that line being placed}) = 0.6$

if σ value of a line = 0.4 $\rightarrow P(\text{pts on that line being placed}) = 0.4$

\therefore we can find $P(\text{Nahi hogaj}) = 1 - P(\text{placed})$.

Hence abhi ham student ka placement hone ka prob nahi hogaj instead of flip coin.

& using this probabilistic interpretation we will make changes in our formula to get the desired result.

$$W_{lin} = w_0 + \eta (y_i - \hat{y}_i) x_i$$

$$\hat{y}_i = \sigma(z); \quad z = \sum_{i=0}^2 w_i x_i \quad \& x_0 = 1$$

Consider an Example :- $\hat{y}_i > 0.5$

mean predicted placed else
predicted not placed.

y_i	\hat{y}_i	$y_i - \hat{y}_i$
1	0.8	$0.2 \rightarrow w_n = w_0 + \eta(0.2) X_i$
0	0.65	$-0.65 \rightarrow w_n = w_0 + \eta(-0.65) X_i, \hat{y} = \sigma(z) = 0.15$
1	0.3	$0.7 \rightarrow w_n = w_0 + \eta(0.7) X_i$
0	0.15	$-0.15 \rightarrow w_n = w_0 + \eta(-0.15) X_i$

↓
not zero for any point (unlike step fth)

3rd case - incorrectly classified

$$\begin{cases} y_i = 1 \rightarrow \text{placed} \\ \hat{y}_i = 0.3 < 0.5 \rightarrow \text{predicted not placed} \end{cases}$$

~~and~~ same hai
tak dheere pull hua

4th case - correctly classified

$$y_i = 0 \quad (\text{Actual not placed})$$

$$\hat{y}_i = 0.15 \quad (\text{Predicted not placed})$$

aur hai tak dheere ~~↑~~ push kiya.

1st case - correctly classified
& far tak aheere ~~↑~~
push kiya (add kiyा)

2nd case - incorrectly classified
& dur hai tak strongly
pull kiya

Case I : $w_n = w_0 + \eta \times 0.2 X_i \rightarrow$ add kar rha hai teh niche shej
shej line \rightarrow hence (I) ne push kiya line

Case II : $w_n = w_0 + \eta \times (-0.65) X_i \rightarrow$ subtract \rightarrow upar shej rha hai
matlab pull kiya line ko (II) ne

Case III : $w_n = w_0 + \eta \times (0.3) X_i \rightarrow$ add \rightarrow niche shej : matlab
(III) pull kar rha hai line ko upar takay

Case IV : $w_n = w_0 - \eta \times 0.15 X_i \rightarrow$ subtract \rightarrow line ko upar shej
matlab line ko push kar rha hai (IV)

\therefore Misclassified pt \rightarrow pull karne hai

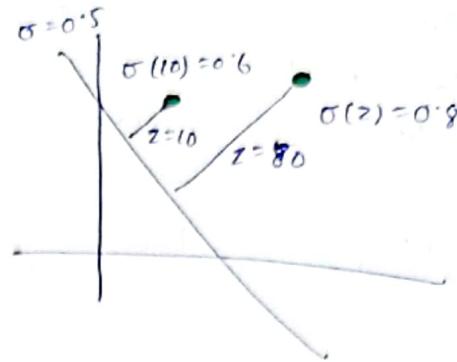
& correctly classified pts \rightarrow push kar rha hai line ko.

what offset

Let's see the distance of pts. from line have on the magnitude of push & pull: $\sigma = 0.5$

page 63

pt. no.	y_i	\hat{y}_i	$y_i - \hat{y}_i$	w_n
(1)	1	0.6	0.1	$w_0 + w_1 x_0 + 0.1 x_1$
(2)	1	0.8	0.2	$w_0 + w_1 x_0 + 0.2 x_1$
(3)	0	-	-	-
(4)	0	-	-	-



case I: add kiya toh line niche jaygi & jyada add kiya compared to (2) toh jyada niche jaygi hence correct pt. jo same hai wo jyada push kar rahi hai

Case II: add kiya toh line niche jaygi & kam add kiya compared to (1) toh kam niche jaygi compared to (1) & hence correct point jo dur hei wo kam push kar rahi hai.

Case III: same thing (reversed order) can be seen

& Case IV:

Jupyter Notebook

* even with sigmoid function the code was not as good as the sklearn Logistic Regression.

so further improvement :-

Sigmoid fn + perception trick flaw :-
hum 1000 baar loop run kar rahi hai & har iteration mei ek random point lekar pata kar rahi hai if mis classified line ko apni taraf pull karo nahi to push karo \rightarrow finally jo answer aya wo hum pure conviction ke saath nahi lekar kaise ki sake hai. Since again hum durr baan code run karne the phir 1000 random pts select honge & on that basis a final line will be selected but the random pts selected now may be diff from the previous case & hence the line may be different \rightarrow This is not how ML works \rightarrow Every time we run the code for the same data set the line must be same.

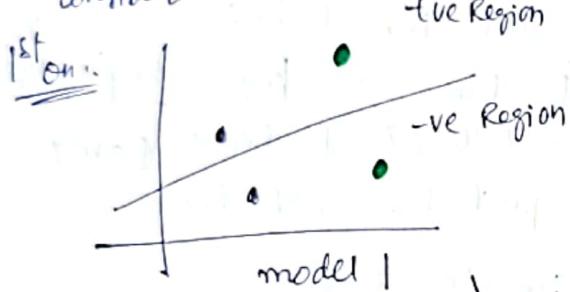
So our Algo is only finding a θ_0 but it does not guarantee that the θ_0 obtained is the best θ_0 . Since next time we run it, it may give a better θ_0 , but again we have no guarantee it is the best one.

So using the ML way:- finding the Loss fn or Error fn

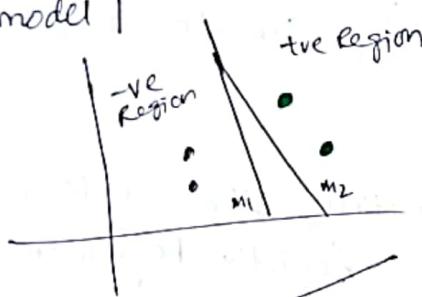
Finding the Loss fn

Maximum Likelihood:-

Consider two models:-



Next on:

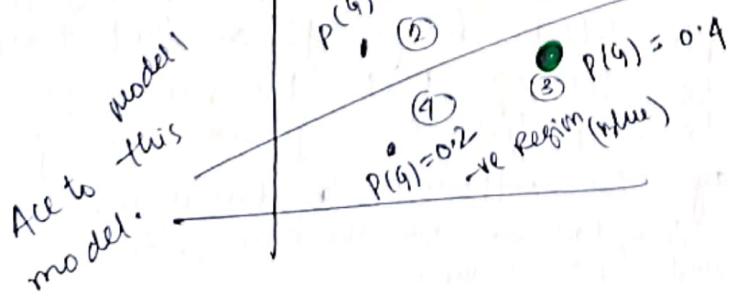


→ of course Model 2 is better since data is classified atleast.

Now if asked which model is better M1 or M2 → Difficult

Here Loss fn will help & tell which one is better. & using it we can fetch the best model.
To say To derive / find Loss fn we use concept called

Consider the models in O.H.



Maximum (green) likelihood (probability concept).

$P(G) = 0.7$ (1) $P(B) = 0.3 = \text{Prob}(\text{placed}) - P(\text{green})$

$$P(G) = 0.6$$

$$(2) \quad P(B) = 0.4 = \text{Prob}(\text{not placed}) - P(\text{green})$$

$$P(G) = 0.2$$

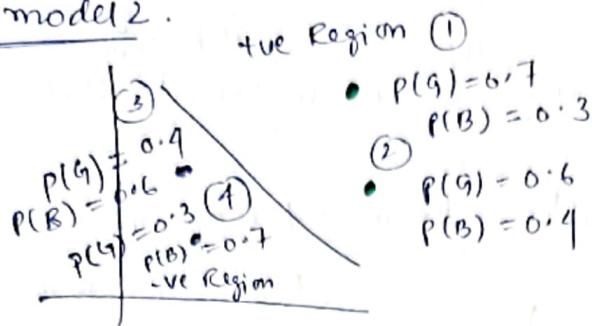
$$(3) \quad P(B) = 0.9 = \text{Prob}(\text{placed}) - P(\text{green})$$

→ To find $P(B)$ for all point

$$P(B) = 1 - P(G)$$

$$\text{Prob}(B) = \text{Prob}(\text{not placed})$$

Acc to model 2.



Page 69.

$$\begin{aligned} \hat{y} &= \sigma(z) = \text{Prob} \\ z &= \sum w_i x_i \end{aligned}$$

(g that
point
to be
placed)

Now Acc. to maximum likelihood says:
for a model multiply all the prob obtained of the model with the highest prob is better.

But: AAP Sif wo prob logo ek point ka jis color ka wo point hai. \rightarrow maximum likelihood for model 2

for model 2: Total prob = $0.7 \times 0.6 \times 0.6 \times 0.3 = 0.176$
 $= P(1) \times P(2) \times P(3) \times P(4)$

for model 1: Total prob = $0.7 \times 0.4 \times 0.4 \times 0.8 = 0.089$
 $= P(1) \times P(2) \times P(3) \times P(4)$ \rightarrow maximum likelihood for model 1.

∴ Maximum likelihood of model 2 $>$ max likelihood of model 1.
∴ Model 2 is better than model 1.

Hence humne ek aisa model line chahiye jiska maximum likelihood max aaye. Calculation of

The only problem is \rightarrow product of small Nos.

upar bas 4 prob ko multiply karne pe 0.089 aya. In real world there would be 10000 or more data. & hence the product will be very small & hence the comparison will not be valid.

So we convert the product to sum

i.e. the model whose $\log(\text{sum of prob})$ is max is best.

i.e. for model 1: $\log(\text{max likelihood}) = \log(0.7) + \log(0.4)$
 $+ \log(0.4) + \log(0.8)$

for model 2: $\log(\text{max likelihood}) = \log(0.7) + \log(0.6)$
 $+ \log(0.6) + \log(0.3)$

Something to Note :-

$\log(x)$ if $x \in (0, 1)$ is -ve \rightarrow so to avoid this

for model 1 : $\log(\text{max likelihood}) = -(\log(0.7) + \log(0.9) + \log(0.4) + \log(0.8))$

for model 2 : $-\log(\text{max likelihood}) = -(\log(0.7) + \log(0.6) + \log(0.4) + \log(0.3))$

This is called cross entropy.

Now as we multiplied by -ve

\therefore the model with minimum cross entropy will be the best

we were maximizing max likelihood & log is 1st for so $\log(\text{max likelihood})$ ko bhi maximize karna log ka but $-\log(\text{max likelihood})$ = cross entropy ko minimize karna naga for best model selection.

\therefore formula of cross entropy \rightarrow (note hume wo prob lena hota hai jis color ka wo ball hai).

for placed pts / Green pts $y_i = 1$

& for blue pts / not placed pt. $y_i = 0$

\therefore for any pt. the cross entropy value is :-

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

\therefore for Green pts.

$$\text{cross entropy} = -\log(\hat{y}_i)$$

& for blue pts.

$$\text{cross entropy} = -\log(1-\hat{y}_i)$$

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

$$\& \text{to find average error} = \frac{1}{n} \sum_{i=1}^n y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)$$

\hookrightarrow This is called log loss error or Binary cross entropy function.



And Now we need to minimize this

page 65

fun.

i.e. we need to find w_0, w_1, w_2 such that:-

$$L = -\frac{1}{n} \sum_{i=1}^n y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)$$
 is minimized.

→ This does not have any closed form soln (i.e. direct formula for w_0, w_1, w_2) & hence we'll need to use Gradient Descent to find the soln.

→ This is what happens in the Scikit Learn Logistic Regression.

Dervative of Sigmoid function:-

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

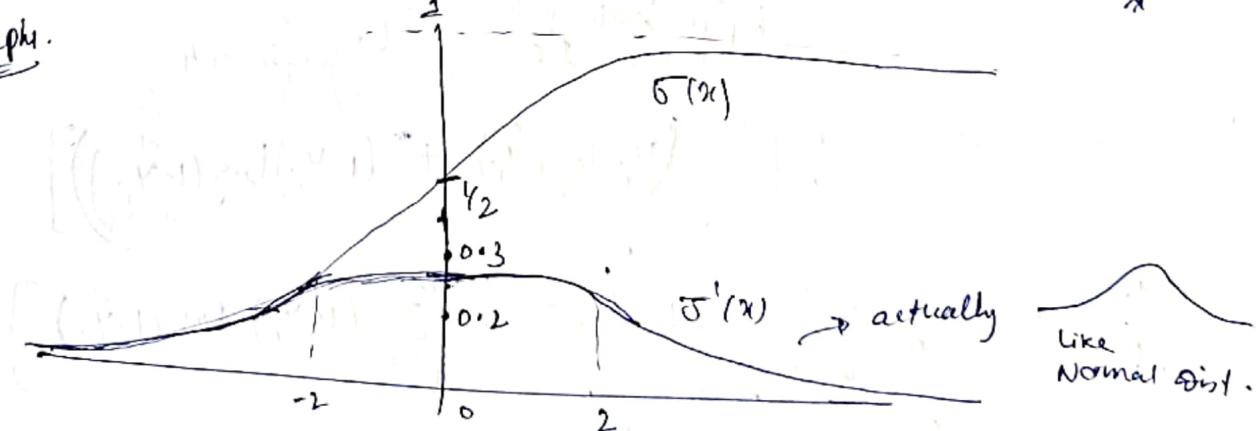
$$\sigma(x) = \frac{1}{1+e^{-x}} \rightarrow \sigma'(x) = \frac{d}{dx} \left(\frac{1}{1+e^{-x}} \right) = \frac{-[0+e^{-x}(-1)]}{(1+e^{-x})^2}$$

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}}{(1+e^{-x})}$$

$$\sigma'(x) = \sigma(x) \left[\frac{e^{-x}}{1+e^{-x}} \right] = \sigma(x) \left(1 - \frac{1}{1+e^{-x}} \right)$$

$$\sigma'(x) = \sigma(x) - (\sigma(x))^2 = \sigma(x) (1 - \sigma(x))$$

Graph.



Optimizing the loss fn using gradient descent

consider a data set with m rows & n input cols

1	2	3	...	n	output(y)
x_{11}	x_{12}	x_{13}	...	x_{1n}	y_1
x_{21}	x_{22}	x_{23}	...	x_{2n}	y_2
.
x_{m1}	x_{m2}	x_{m3}	...	x_{mn}	y_m

as in Logistic Regression we need to find

the coeff - w_1, w_2, \dots, w_n &
intercept - w_0

$\hat{y}_i = \text{sigmoid of}$

$$w_0 + w_1 x_{i1} + \dots + w_n x_{in}$$

Total (n+1) coeff

$$= \sigma(w_0 + w_1 x_{i1} + \dots + w_n x_{in})$$

matrix of
predicted
values

$$\text{So: } \hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} = \begin{bmatrix} \sigma(w_0 + w_1 x_{11} + \dots + w_n x_{1n}) \\ \sigma(w_0 + w_1 x_{21} + \dots + w_n x_{2n}) \\ \vdots \\ \sigma(w_0 + w_1 x_{m1} + \dots + w_n x_{mn}) \end{bmatrix}$$

This Sigmoid operation is on all terms

$$\hat{y} = \sigma \left(\begin{bmatrix} w_0 + w_1 x_{11} + \dots + w_n x_{1n} \\ \vdots \\ w_0 + w_1 x_{m1} + \dots + w_n x_{mn} \end{bmatrix} \right)$$

$$\hat{y} = \sigma \left(\begin{bmatrix} [x_{11} \ x_{12} \ \dots \ x_{1n}] \\ [x_{21} \ x_{22} \ \dots \ x_{2n}] \\ \vdots \\ [x_{m1} \ x_{m2} \ \dots \ x_{mn}] \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \right) = \sigma(XW)$$

input data with
1st col of all 1's.

coefficients.

$$\text{Now: Loss fn} = -\frac{1}{m} \left[\sum_{i=1}^m (y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)) \right]$$

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

taking this out:

$$\sum_{i=1}^m y_i \log(\hat{y}_i) = y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + \dots + y_m \log \hat{y}_m$$

$$\sum_{i=1}^m y_i \log(\hat{y}_i) = [y_1, y_2, \dots, y_m] \cdot \begin{bmatrix} \log \hat{y}_1 \\ \log \hat{y}_2 \\ \vdots \\ \log \hat{y}_m \end{bmatrix}$$

$$= [y_1, y_2, \dots, y_m] \cdot \log \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} = Y \cdot \log(\hat{Y})$$

$$= Y \cdot \log(\sigma(xw)) \quad \text{→ matrix form}$$

Page 66.

Actually transpose
ka problem hai
start se handle
karo).

$$\text{Silly. } \sum_{i=1}^m (1-y_i) \log(1-\hat{y}_i) = (I-Y) \log(\sigma(I-xw)) \quad \text{Loss fn in matrix form.}$$

$$\text{Thus. Loss fn } L = -\frac{1}{m} \left[Y \log(\hat{Y}) + (I-Y) \log(I-\hat{Y}) \right]$$

This does not have any closed form soln. ∴ we have to use gradient descent
to minimize it: we initialize $W = [\dots]$, random entries

& we decide epochs & then we update the W with formula:- $W_{\text{new}} = W_{\text{old}} - \eta \frac{\Delta L}{\Delta W}$ → gradient of L w.r.t w .
i.e., learning rate

Finding $\frac{\Delta L}{\Delta W}$:

$$L = -\frac{1}{m} \left[Y \log(\hat{Y}) + (I-Y) \log(I-\hat{Y}) \right]$$

$$\frac{\partial L}{\partial W} = -\frac{1}{m}$$

$$\left[\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_n} \right]$$

$$\frac{\partial}{\partial w}(Y \log \hat{Y}) = Y \frac{\partial}{\partial w} \log \hat{Y} = Y \frac{1}{\hat{Y}} \frac{\partial}{\partial w} \hat{Y} = Y \frac{1}{\hat{Y}} \frac{\partial}{\partial w} \sigma(xw)$$

$$= \frac{Y}{\hat{Y}} \sigma'(xw) [I - \sigma(xw)] \frac{\partial}{\partial w} xw$$

$$= \frac{Y}{\hat{Y}} \hat{Y} (I - \hat{Y}) X \quad \because \sigma(xw) = \hat{Y}$$

$$\begin{aligned} \frac{\partial}{\partial x} \sigma(x) &= \frac{1}{1+e^{-x}} \\ \sigma'(x) &= \sigma(x)(1-\sigma(x)) \end{aligned}$$

$$\text{Silly. } = Y (I - \hat{Y}) X$$

$$\begin{aligned} \frac{\partial}{\partial w} (I-Y) \log(I-\hat{Y}) &= (I-Y) \frac{\partial}{\partial w} \log(I-\hat{Y}) = \frac{I-Y}{I-\hat{Y}} [-\hat{Y}(I-\hat{Y})] X \\ &= -(I-Y)\hat{Y}X = -\hat{Y}(I-Y)X \end{aligned}$$



$$\begin{aligned}
 \text{Thus } \frac{\partial L}{\partial w} &= -\frac{1}{m} \left[y(I-y)x - \bar{y}(I-\bar{y})x \right] \\
 &= -\frac{1}{m} [y(I-y) - \bar{y}(I-\bar{y})]x \\
 &= -\frac{1}{m} [y - y\bar{y} - \bar{y} + \bar{y}\bar{y}]x \\
 &= -\frac{1}{m} [y - \bar{y}]x
 \end{aligned}$$

Do the derivative of loss fn without the matrix form to find gradient.

Hence:-

$$W_{\text{new}} = W_{\text{old}} + \frac{\eta}{m} (y - \bar{y})x$$

$$\begin{aligned}
 w &= \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix} ; x = \begin{bmatrix} 1 & x_1 & x_2 & \dots & x_n \end{bmatrix} ; y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \\
 y &= \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} ; (n+1)x_1 \quad ; \quad \begin{bmatrix} 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} ; y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \\
 &\quad (m, n+1) \quad m \times 1
 \end{aligned}$$

Jupyter Notebook.

problem

We learnt Regression metrics in ~~Classification~~ Regression, to find which model performs better on a Dataset.

Similarly classification problems have classification metrics which guide us to ki koi classification model kitna accha perform kar raha hai.

There are lot of classification metrics:-

Confusion, precision, Accuracy metric, F1-score, ROC, AUC etc.

Accuracy

Consider: CGPA | IQ | placement

Actual Label	placed or not:-	Log pred.	Dec. Tree pred
1	1	1	1
0	1	0	0
0	0	0	0
0	0	0	1
1	1	0	0
0	0	0	0
0	0	1	1

800 train data
1000 students data \rightarrow 200 test data
we use two classifiers
Algorithm :-

Logistic Reg

↓ Train
↓ Test

Decision Tree

↓ Train
↓ Test

→ On which model is perfor best
on the Test Data.



To find which model is better:- for every Page 67.
 Test data we find Log. Reg. ne galat ki ya nahi &
 same for Decision Tree & then find kisne kam
 galat ki & that model is better.

$$\text{Accuracy score} = \frac{\text{# of correct pred}}{\text{Total Pred}}$$

$$\therefore \text{acc. score for Log. Reg} = \frac{8}{10} = 0.8 = 80\%.$$

$$\text{acc. score for Dec. Tree} = \frac{9}{10} = 0.9 = 90\%.$$

∴ Hence Dec. Tree is performing better on the test data than Log. Reg.

Jupyter Notebook (Kaggle)

Accuracy score of multi class problem:- (output has more than 2 category)

Consider the iris dataset :- The output has three categories:

Setosa, Virginica, Versicolor

Actual Label	Log. Reg. Pred		Dec. Tree Pred
	0	1	
0	0	0	0
0	0	0	0
0	0	0	2
2	2	2	0
0	0	2	2
2	2	2	0
0	0	2	2
2	2	2	1
1	1	1	1

→ in this case

$$\text{accuracy score} = \frac{\text{# of correct pred}}{\text{Total Pred}}$$

$$\therefore \text{acc. score of Log. Reg} = \frac{10}{10} = 100\%.$$

$$\text{acc. score of Dec. Tree pred} = \frac{10}{10} = 100\%.$$

How much accuracy is good? (matlab:- kitna accuracy hoga hai)

Log. Reg. actually depends upon the data

→ 80% accuracy pe shi model ko deploy kar sakte hai
 & 99% pe shi nahi kar sakte.

Meaning: good order karne ya nahi wale model ka 80% accuracy hai
 toh bhi deploy kar sakte hai since agar wrong predict kiya toh
 such khay loss nahi hogा

par:- heart disease hai ya nahi wale model ko 99% pe

bhi deploy nahi kar sakte :- galti kiya toh patient
marega. → galti ka stake Lekhni high hai.

The problem with accuracy Score:-

It gives a number matlab say acc-score = 0.9

matlab 10% galti par rha hai model par acc-score
ye nahi bata rha ki galti ka type kya hai.

say binary classification hai:-
Galti types Actual | Predicted
 1 | 0

Actual | Predicted
0 | 1

Type 2 Error
(False Negative)
Placement hua
that but model ne not placed
predict kiya

Placement nahi hua
that but model
ne placed predict
kiya.

Acc Score ye nahi batayega ki kong
Error kitae baar hua in that
10% galti.

Type 1 Error. (False positive)

To handle this:-
we use

Confusion Matrix.

first see jupyter Notebook how to calculate
Confusion Matrix using scikit learn.

Confusion Matrix: (scikit learn Nomenclature)

		Prediction	
		1	0
Actual Values	1	True Positive	False Negative
	0	False Positive	True Negative

Type 2 error
Sahi tha Galat
predict kar diya.

galat tha
galat predict
kiya.

issee hum accuracy
score bhi nikal sakte
hai

Type 1 error
(Galat tha Sahi
predict kar diya)

This is when
output Haast cat
has only two
classes (binary)

Ex: Heart Disease.

Actual	1	0
1	26	6
0	0	29

- 26 logo ko heart disease tha & model ne bhi predict kiya
- 29 logo ko nahi tha & model ne bhi predict kiya
- 6 logo ko heart disease tha & model ne bhi predict kiya

$$\text{accuracy score} = \frac{\# \text{ of correct pred}}{\text{total pred}}$$

$$= \frac{\text{True pos} + \text{True Neg}}{\text{True pos} + \text{False pos} + \text{True Neg} + \text{False Neg}}$$

- * False positive → is called Type 1 Error.
- & False Negative → is called Type 2 Error.

Confusion Matrix for multiclass classification problem
output mei more than 2 categories hai

		predicted		
		0	1	2
Actual	0	1	1	1
	1	1	1	1
2	1	1	1	1
	1	1	1	1

∴ the Accuracy score = $\frac{\# \text{ of correct pred}}{\text{total pred.}}$

are the Right/ Prediction / correct.

in the MNIST Data set → there are 10 classes (0-9)
& hence the confusion Matrix will be 10×10 order.

When accuracy is misleading? # of elements/data
↳ Imbalanced dataset → i.e. the categories are not equal/almost equal.

ex: Airport passenger hai ya Terrorist hai.
This will be highly imbalanced data.
In such case accuracy gives bad result.

		predicted	
		0	1
Actual	0	0	99999
	1	99999	0

→ acc score = 99.999 %.
↳ almost 100 %.

↳ to miss Kargil par
that I is dangerous
& the stakes are high.

hence we study Precision & Recall & F1-Score

Consider the Ex:-

(Email spam classifier)

		predicted			
		sent to spam	Not sent to spam	sent to spam	Not sent to spam
Actual	spam	100	170 false neg.	100	170 false neg.
	not spam	30 false pos	700	700	30 false pos
Model 1					

		predicted			
		sent to spam	Not sent to spam	sent to spam	Not sent to spam
Actual	spam	100	170 false neg.	100	170 false neg.
	not spam	700	30 false pos	700	30 false pos
Model 2					

- model 1 :-
- 30 not spam are sent to spam
 - 170 spam are not sent to spam

- in model 2 :-
- 10 not spam are sent to spam
 - 170 spam are not sent to spam.

Here: not spam messages are important (as they may contain something imp).

if more non spam are sent to spams \rightarrow it is not good.
Hence model with less False pos mistake is good.
Hence model 2 is good comparatively.

precision ko precision bala jata hai.

Logi is what Proportion of Predicted Positive

is truly positive :-

Higher the precision
Better the model.

$$\therefore \text{precision} = \frac{\text{True Positive}}{\text{True pos} + \text{False pos}}$$

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

in model 1 precision = $\frac{100}{130} = \frac{10}{13}$

in model 2 precision = $\frac{100}{100} = \frac{10}{11} \rightarrow$ this is greater, hence

in this case model 2 is better than model 1.

To understand Recall :-

P10

Cancer detection:

	Detected Cancer	Not detected	
Has cancer	1000	200	False Neg
No cancer	800	8000	

model 1 false pos

Det. Cancer	Not Det.		
Has cancer	1000	500	False Neg
No cancer	500	8000	

model 2. False pos

accuracy score of both is same

In this case, false Negative (has cancer but not detected) is more dangerous. Hence Lesser False Negative the better the model.

in model 1 false neg is less than model 2.

~~false neg~~ is Better. \downarrow This is called Recall.

Defn: Recall :- What proportion of Actual positive is correctly classified (is predicted positive)?

$$\therefore \text{for model 1 :- } \text{Recall} = \frac{1000}{1200} = \frac{10}{12}$$

$$\text{for model 2 :- } \text{Recall} = \frac{1000}{1500} = \frac{10}{15}$$

formula:- $\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negatives.}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$

Lesser the Recall \rightarrow Better the Model.

Hence:- if Type 1 Error is dangerous \rightarrow then High precision model is preferred/better

& if Type 2 Error is dangerous \rightarrow then Low Recall model is better.

F1 SCORE

Sometimes we cannot say in a classification problem why error Type 1 or Type 2 is more dangerous. Ex: cat, dog image classification (binary classification).

There: Type I error \rightarrow cat ko dog bol diya \rightarrow which is dangerous
 Type II error \rightarrow dog ko cat bol diya \rightarrow now we don't know
 In this case both precision & recall has to be handled — which is difficult as there is a tradeoff b/w them (they are kind of inv. proportional).
 In this case we use F1-Score metric.

It is a combination of precision & recall
 (actually a Harmonic Mean of both).

$$F1\text{-Score} = \frac{2PR}{P+R} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

\downarrow F1-Score $\rightarrow [0, 1]$
 best

A higher F1-Score is generally better.
 worst

It is still able to give true model performance when dataset is imbalanced.

Jupyter Notebook

Precision, Recall & F1-Score for multi class classification.

Consider Image classifier

confusion matrix:

		predicted			Actual	Total
		Dog	Cat	Rabbit		
Actual	Dog	25	5	10		
	Cat	0	30	4		
Rabbit	4	10	20			

Predicted Total 29 45 34
 ↓ Predicted no. of Dogs
 ↓ Predicted no. of Cats
 ↓ Predicted no. of Rabbits.

Harmonic Mean
 ex. aisa mean hai jo hamesha lower value ke side inclined rehta hai

$$\text{Ex: } P=0, R=100$$

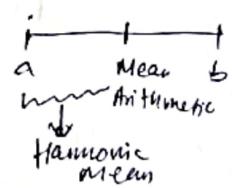
$$AV = \frac{P+R}{2} = 50$$

$$F1\text{-Score} = 0$$

$$\downarrow a, b \quad a > b$$

$$\text{Harmonic mean} = \frac{1/a + 1/b}{2}$$

$\frac{1}{b} > \frac{1}{a} \therefore$ harmonic mean shifts towards $\frac{1}{b}$



40 \rightarrow Actual no. of dogs
 34 \rightarrow Actual no. of ~~cat~~ cat
 34 \rightarrow Actual no. of Rabbit

Calculation for precision = $\frac{\text{Actual pos}}{\text{total pred pos}}$

precision for dog class = $\frac{\text{# of Actual dogs}}{\text{total # of predicted dogs}} = \frac{25}{39} = 0.64$

2nd col \rightarrow n Cat class = $\frac{\# \text{ of Actual cats}}{\text{total # of predicted cats}} = \frac{30}{45} = 0.66$

3rd col \rightarrow n Rabbit class = $\frac{\# \text{ of Actual Rabbit}}{\text{total # of predicted Rabbit}} = \frac{20}{39} = 0.51$

Combined precision of the model.

Naive precision = Avg. all the precision = $\frac{0.86 + 0.66 + 0.51}{3} = 0.70$
 When classes are balanced

Weighted precision = $\sum (\text{class weight} \times \text{precision}) = (0.86 \times \frac{40}{108}) + (0.66 \times \frac{39}{108}) + (0.51 \times \frac{34}{108})$
 When classes are imbalanced.

Calculation for Recall = $\frac{\text{actual pos}}{\text{total actual pos}}$ = 0.71

for dog \rightarrow Recall of dog class = $\frac{25}{40} = 0.62$

for cat \rightarrow Recall of cat class = $\frac{30}{39} = 0.77$

for rabbit \rightarrow Recall of rabbit class = $\frac{20}{39} = 0.51$

Combined recall of the model

Naive recall = Avg. all Recall = $\frac{0.62 + 0.77 + 0.51}{3} = 0.66$
 When classes are balanced

Weighted Recall = $\sum (\text{class weight} \times \text{recall}) = (0.62 \times \frac{27}{108}) + (0.66 \times \frac{45}{108}) + (0.51 \times \frac{34}{108})$
 When classes are imbalanced

Similarity Calculation of F1-score

$$\text{F1-score of dog class} = \frac{2 P_D R_D}{P_D + R_D}$$

$$\text{" " " cat class} = \frac{2 P_C R_C}{P_C + R_C}$$

$$\text{" " " rabbit class} = \frac{2 P_R R_R}{P_R + R_C}$$

Again, Macro F₁ score = Av. of all F1-score

Weighted F₁-score =

So far the Logistic Regression we studied:-
the output/target col had two classes/categories:
What if there are more than two :- (Binary classification)

Then we use:-

Softmax Regression or Multiclass/Multinomial Logistic Regression

Consider dataset:-

Cgpa	Io	placement
7.1	7.1	0
8.5	8.5	1
9.5	9.7	2

The classes here are Placed 1
Not placed 2
Opt out 3

We need to a Logistic Reg. classifier that predicts output given the necessary input.

Logistic Regression (for binary output) is a special case of Softmax Regression.

What is Softmax function:-

; k = # of classes in the output col.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$

placed.

$$\sigma(z)_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} ; \sigma(z)_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}} ; \sigma(z)_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

Prob of placed class

↓
Prob of not placed class

↓
Prob of opt out class.

Note :- $0 < \sigma < 1$

$$f \sum_{i=1}^K \sigma(z_i) = 1$$

If softmax for two classes for output is sigmoid fn.

Training intuition of softmax Reg. (just an example, nahi exactly scikit learn mein nahi hota hai)

		$\{ Y = 0 \}$		
		$\{ N = 1 \}$		
		$0 \rightarrow 1$		
cgpa	1.8	Placement		
1	0			
2	1	one-hot encoding of output cat		
3	2			

class not placed			class 1 placed	class 2 placed	class 3 placed
			opt out		
cgpa	0	1	1	0	
1	0	0	0	1	
2	0	0	0	0	1

Now we have 3 cols.

Now we split it into three (No. of categories) datasets.

As there are three output cols & they are binary classified.

		class 1
		D ₁
cgpa	1.8	not placed
1	;	;
2	;	;

		D ₂
		D ₃
cgpa	3.0	placed=1 : class 2
1	;	;
2	;	;

		D ₃
		D ₃
cgpa	7.0	opt out: 3 : class 3
1	;	;
2	;	;

Now we train this on

Three Logistic Reg model.

Model 1

This will return
3 coeff (of 2 input
data)
 $w_1^{(0)}, w_2^{(0)}, w_0^{(0)}$

Model 2

3 coeff
 $w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$

Model 3

3 coeff
 $w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$

Actually scikit learn mein there different
hotai hai. (loss fn + gradient descent
use ho raha hai).

Now training is done:- Let's focus on Predictions:-

M1

$w_1^{(0)}, w_2^{(0)}, w_0^{(0)}$

Given a query point

This model predicts
placement hua ya nahi

placement hua :- pos class

nahi :- Baaki do cases
opt out ya not placed.

M2

$w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$

Given a query point
it predicts

1 : placement hua
(pos class)

0 : baaki dono class
not placement hua

ya opt out

M3

$w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$

Given a query point it
predicts :-

1 : opt out (pos class)

0 : Baaki dono class

placed ya not placed.

Given a student :- CGPA = 7, IQ = 70 \rightarrow To predict (placed, not placed) or Opt out

not placed

M1 :-

placed

M2.

Opt out

M3.

$$w_1^{(0)}, w_2^{(0)}, w_3^{(0)}$$

$$z_1 = 7w_1^{(0)} + 70w_2^{(0)} \\ + w_3^{(0)}$$

$\xrightarrow{70}$ \leftarrow -ve side
means either
not placed or
placed or
out but

Now we apply the

$$\sigma(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

! Prob of not placed || 0.35 (say)

$$w_1^{(1)}, w_2^{(1)}, w_3^{(1)}$$

$$z_2 = 7w_1^{(1)} + 70w_2^{(1)} \\ + w_3^{(1)}$$

$$w_1^{(2)}, w_2^{(2)}, w_3^{(2)}$$

$$z_3 = 7w_1^{(2)} + 70w_2^{(2)} \\ + w_3^{(2)}$$

\rightarrow say in those two cases
also.

softmax fn:-

$$\sigma(z_2) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$$\sigma(\text{placed}) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

prob of placed say = 0.90

$$\sigma(z_3) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$\sigma(\text{opt out})$

= prob of opt out

0.25

(say)

Hence going by this logic prob of placed is highest
 \therefore the student will get placed.

To #q Logistic Reg = #q classes

Hence if there are more no. of classes ; then
will be more no. of logistic Regression to train
the process will be quite slow

\downarrow so we use diff approach

we change the loss fn of the Logistic Regression
using that changed loss fn \rightarrow using a single model
we can train the above three dataset & get all
the coefficients for each model.

And for this reason this is
not implemented
in scikit learn

This is implemented in scikit learn

Say we choose degree 2.

Page 72

Loss function

$$\text{Logistic Regression Loss fn} = -\frac{1}{m} \sum_{i=1}^m Y_i \log(\hat{Y}_i) + (1-Y_i) \log(1-\hat{Y}_i)$$

Softmax Reg / Multinomial Logistic Regression Loss fn =

$$-\frac{1}{m} \sum_{j=1}^m \sum_{i=1}^k Y_j^{(i)} \log(\hat{Y}_j^{(i)})$$

for rows \downarrow $\sum_{j=1}^m$ narrow rectangle $\#$ of classes k summation bar.

	x_1	x_2	y
	x_{11}	x_{12}	1
	x_{21}	x_{22}	2
	x_{31}	x_{32}	3

one hot encoding.

x_1	x_2	$y_{k=1}$	$y_{k=2}$	$y_{k=3}$
x_{11}	x_{12}	1	0	0
x_{21}	x_{22}	0	1	0
x_{31}	x_{32}	0	0	1

$$\text{Loss fn} = -\frac{1}{3} \left[Y_1^{(1)} \log(\hat{Y}_1^{(1)}) + Y_2^{(1)} \log(\hat{Y}_2^{(1)}) + Y_3^{(1)} \log(\hat{Y}_3^{(1)}) + Y_1^{(2)} \log(\hat{Y}_1^{(2)}) + Y_2^{(2)} \log(\hat{Y}_2^{(2)}) + Y_3^{(2)} \log(\hat{Y}_3^{(2)}) + Y_1^{(3)} \log(\hat{Y}_1^{(3)}) + Y_2^{(3)} \log(\hat{Y}_2^{(3)}) + Y_3^{(3)} \log(\hat{Y}_3^{(3)}) \right]$$

$m=3$ \leftarrow for model 1
for class 1 $i=1$
 $k=1, 2, 3$

\leftarrow for model 2
for class 2 $i=2$
 $k=1, 2, 3$

\leftarrow for model 3
for class 3 $i=3$
 $k=1, 2, 3$

for the above Data :-

$$\text{Loss fn} = -\frac{1}{3} \left[Y_1^{(1)} \log(\hat{Y}_1^{(1)}) + 0 + 0 + Y_2^{(2)} \log(\hat{Y}_2^{(2)}) + 0 + 0 + 0 + Y_3^{(3)} \log(\hat{Y}_3^{(3)}) \right] \\ = -\frac{1}{3} \left[Y_1^{(1)} \log(\hat{Y}_1^{(1)}) + Y_2^{(2)} \log(\hat{Y}_2^{(2)}) + Y_3^{(3)} \log(\hat{Y}_3^{(3)}) \right]$$

Now to find: $\hat{Y}_1^{(1)}, \hat{Y}_2^{(2)}, \hat{Y}_3^{(3)}$:-

$$\hat{Y}_1^{(1)} = \sigma \left(w_1^{(1)} x_{11} + w_2^{(1)} x_{12} + w_0^{(1)} \right) \rightarrow \text{where } w_1^{(1)}, w_2^{(1)}, w_0^{(1)} \text{ are weights / coeff wrt col } Y_{k=1}$$

$$\hat{Y}_2^{(2)} = \sigma \left(w_1^{(2)} x_{21} + w_2^{(2)} x_{22} + w_0^{(2)} \right) \rightarrow w_1^{(2)}, w_2^{(2)}, w_0^{(2)} \text{ are coeff wrt col } Y_{k=2}$$

$$\hat{Y}_3^{(3)} = \sigma \left(w_1^{(3)} x_{31} + w_2^{(3)} x_{32} + w_0^{(3)} \right) \rightarrow w_1^{(3)}, w_2^{(3)}, w_0^{(3)} \text{ are coeff wrt col } Y_{k=3}$$



so the coeffs are:- $\begin{bmatrix} w_1^{(1)} & w_2^{(1)} & w_0^{(1)} \\ w_1^{(2)} & w_2^{(2)} & w_0^{(2)} \\ w_1^{(3)} & w_2^{(3)} & w_0^{(3)} \end{bmatrix}$ → are in the loss fn

How to find the gradient of loss fn wrt them
in gradient descent we need to calculate:

$$\frac{\partial L}{\partial w_1^{(1)}}, \frac{\partial L}{\partial w_2^{(1)}}, \frac{\partial L}{\partial w_0^{(1)}}, \frac{\partial L}{\partial w_1^{(2)}}, \frac{\partial L}{\partial w_2^{(2)}}, \frac{\partial L}{\partial w_0^{(2)}}, \frac{\partial L}{\partial w_1^{(3)}}, \frac{\partial L}{\partial w_2^{(3)}}, \frac{\partial L}{\partial w_0^{(3)}}$$

∴ In gradient descent → we initialize with random
q values q the above matrix q then in epochs
we update for each q q coefficient.

$$w_1^{(1)} = w_1^{(1)} - \eta \frac{\partial L}{\partial w_1^{(1)}} \\ w_2^{(1)} = w_2^{(1)} - \eta \frac{\partial L}{\partial w_2^{(1)}} \\ \vdots \\ \text{and so on.}$$

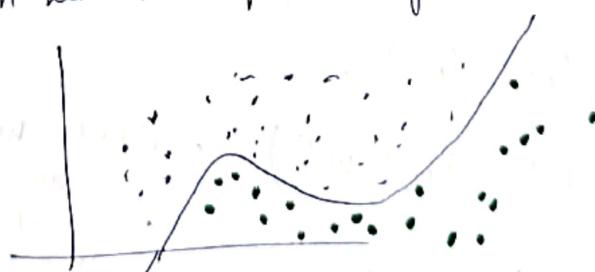
jupyter Notebook

Polynomial features in Logistic Regression

Non-linear logistic Regression

we know logistic regression works when the data is
linearly separable.

But if the data is non linearly separable then we can
use Non linear logistic Regression..



How do we do:- For every feature (input col) we convert it to
polynomial features of any degree we want/need.

Say we choose degree 2.

Page 73.

x_1	x_2	y	\rightarrow	x_1^0	x_1^1	x_1^2	x_2^0	x_2^1	x_2^2	y
:	:	:								

& Now we

use Logistic Regression
jupyter Notebook.

Hyperparameters in Logistic Regression

for

- Penalty: {'l1', 'l2', 'elasticnet', 'none'}: \rightarrow Regularization
default 'l2'
- tol (default = $1e^{-4}$) \rightarrow tolerance \rightarrow it is the stopping criteria for gradient descent.
- C (default = 1.0) \rightarrow it is the inverse of regularization strength,
i.e. $C = \frac{1}{\lambda}$ \rightarrow smaller the value of C stronger the regularization.
- fit_intercept (default = True) \rightarrow w_0 wala value chahiye ya nahi.
- class_weight: (default=None); 'balanced' is used if the data has unbalanced classes in output (like 95% one class & only 5% another class)
- solver: {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default
 \hookrightarrow we made our own Logistic Reg. class using = 'lbfgs'. gradient descent sly apply other optimization techniques on the loss fn, a Logistic Reg class can be made & the above are them. (see the documentation:- which to use when).
- random_state: int; default=None
 \hookrightarrow used when solver = 'sag', 'saga' or 'liblinear' to shuffle data.
- max_iter: int; default = 100
 \hookrightarrow # of epochs/iterations taken for the solver to converge.
- when the output on has more multi-class: {'auto', 'ovr', 'multinomial'}, default = 'auto'.
then of ovr \rightarrow for every class a logistic Reg model is made & coeff are found categories. like we discussed
multinomial \rightarrow Loss fn is minimized & coeff are found.

- warm_start :- default: False → har baar Cnn se training hoga.
when set to True, jis point pe Training stop kar rhe ho, waise baad firse training start karoge then jis point pe stop kiya tha wahi se start hoga.

- n-jobs : int, default: None.
No. of CPU cores used when parallelizing over classes if multi-class = 'ovr'
→ means using all processors → multi processing.
- l1_ratio :-($0 \leq l1_ratio \leq 1$) → only used if penalty = 'elasticnet'.
 $l1_ratio = 0 \rightarrow$ means penalty = ' l_2 '
 $l1_ratio = 1 \rightarrow$ means penalty = ' l_1 '.

DECISION TREES

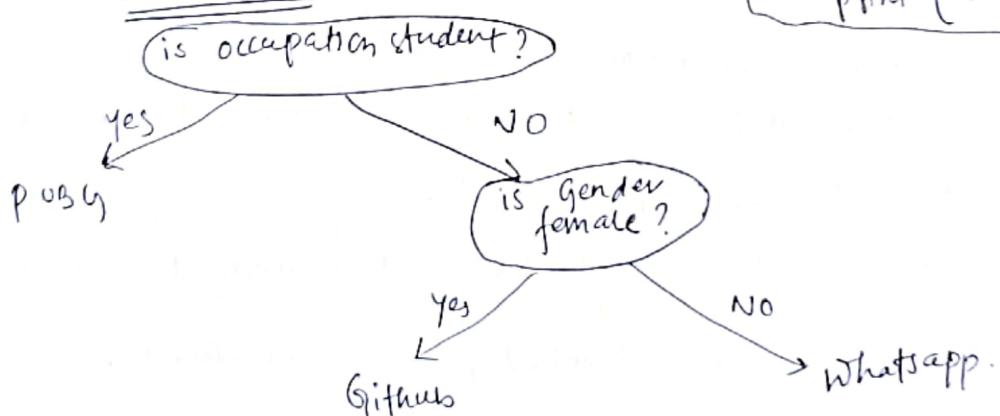
Consider the Data:-

Gender	occupation	suggestion
F	student	PUBG
F	programmer	Github
M	"	Whatsapp
F	"	Github
M	student	PUBG
M	"	"

Based on this Data, we can definitely write a code (no ML) that can predict suggestion for a new input. Since from the data we observe:
Female + student or Male + student = PUBG
Female + programmer = Github
& male + programmer = Whatsapp.

This is a decision tree.
(Decision trees are Nested if, else conditions).

The tree:-



```

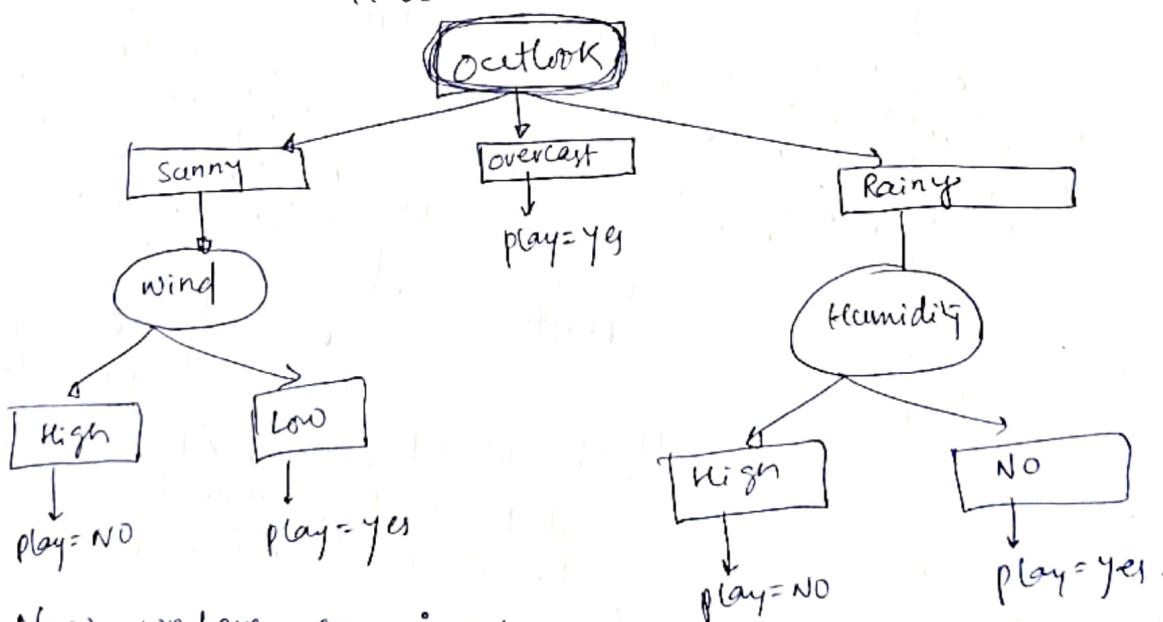
if occupation == 'student'
    print (PUBG)
else if gender == 'female'
    print (Github)
else
    print (Whatsapp)
  
```

Consider the Data:

Page 74.

Day	outlook	Temp	Humid	wind	play?
1	Sunny	Hot	High	weak	NO
2	Sunny	Hot	High	Strong	NO
3	overcast	Hot	High	weak	Yes
4	Rain	Mild	High	weak	Yes
5	Rain	Cool	Normal	weak	Yes
6	Rain	Cool	Normal	Strong	NO
7	overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	weak	NO
9	Sunny	Cool	Normal	weak	Yes
10	Rain	Mild	Normal	weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	overcast	Mild	High	Strong	Yes
13	overcast	Hot	Normal	weak	Yes
14	Rain	Mild	High	Strong	NO

Looking on the weather conditions → we need to predict the player will play or not.
Here the decision tree will be: (say)



Now we have an input query → [Rainy, mild, High, strong]

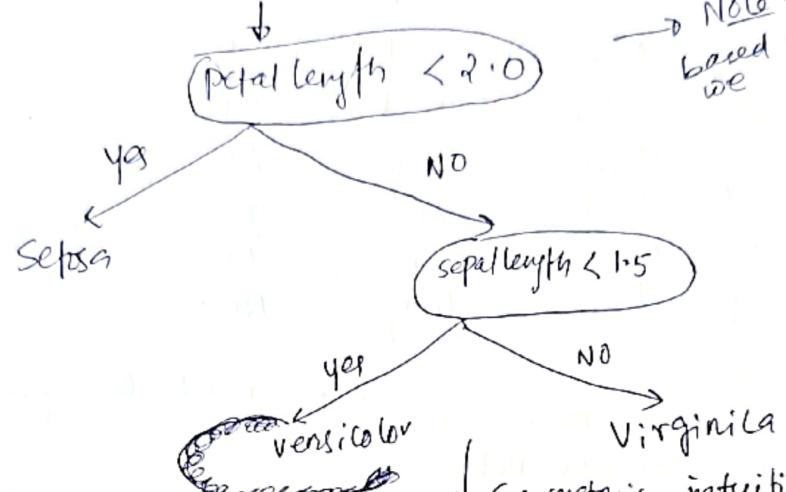
we go to Rainy & then check Humidity instead of Temp & Wind
here predict play = No.

Here, we only checked Outlook & Humidity even though other cond's were there since our decision tree is made that way.

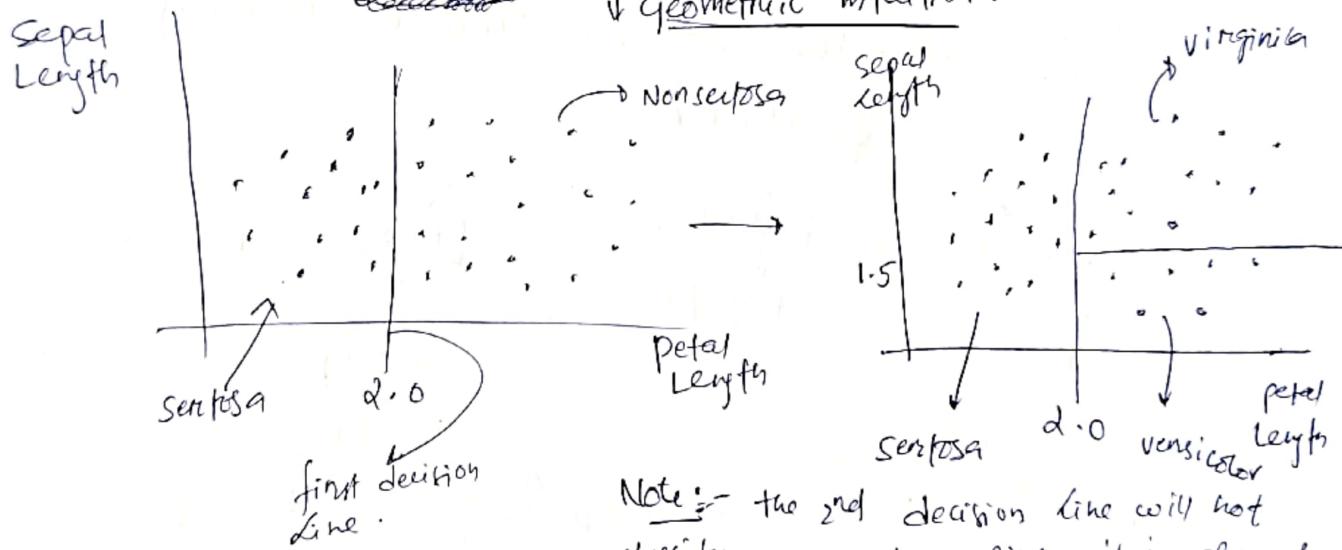
What if we have Numerical Data?

petal-length	sepal-length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.62	Setosa

Note :- first we classify based on one col & then other cols.
we include



Geometric intuition:



Note:- The lines we make are parallel to coordinate axes.

Note:- the 2nd decision line will not classify the 'Setosa', since it is already classified - it classifies the remaining part.

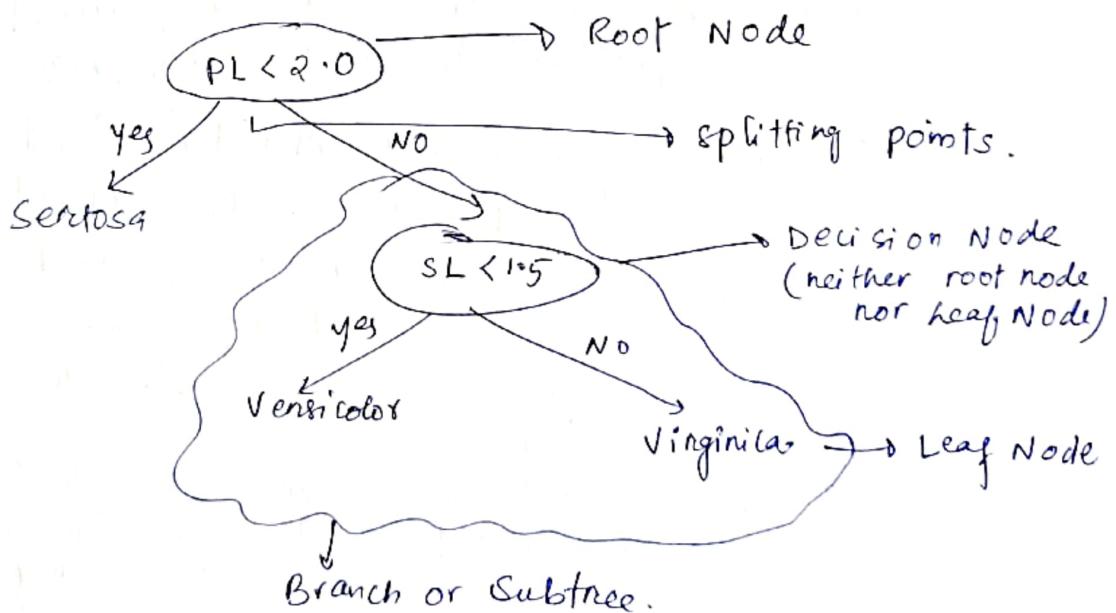
↓ In higher Dimensions the lines will be planes or hyperplanes.

Pseudo code :- for decision Tree

- Begin with your training dataset, which should have some feature variables and classification or regression output.
- determine the "best feature/attr" in the dataset to split the data on (we will see how we define best feature)
- split the data into subsets -that contains the correct values for this best feature. This splitting basically defines a node on a tree i.e. each node is a splitting point based on a certain feature from our data we choose the next best feature
- ^{Noo} Recursively generate new tree nodes by using the subset of data created from step 3.

Conclusion:

- programmatically speaking, Decision trees are nothing but a giant structure of nested if else condⁿ.
- mathematically speaking, decision trees uses hyperplanes which run parallelly to any one of the axes to cut your coordinate system into hyper cuboids.

Terminology:

Some questions:-

- How to decide which column should be considered as root node? (i.e. which col/feature is the best feature).
- How to select subsequent decision nodes? (Next best features/cols)
- How to decide the splitting criteria in case of Numerical columns?

Advantages

- Intuitive & easy to understand
- minimal data preparation is reqd.

Disadvantages

- There is a great chance of overfitting
- prone to errors for imbalanced dataset

The cost of using the tree for inference is logarithmic in the number of data points used to train the tree i.e. Time complexity is $O(\log(\# \text{data points}))$

* Decision Tree is also called CART → classification and Regression trees.

Since it is mostly used for classification problems but the logic of Decision Trees can be applied to regression problems as well, hence named CART.

* There is a game called Akinator :- where we think of a character & the Akinator will ask us questions :- Is he Asian? Is he Indian? Is he an actor? Does he sing etc... & for each answer provided, he will divide the data he has to reach to a conclusion

↳ This is purely based on Decision Tree where the data is divided based on the answer provided (Decision Nodes)

To Learn how the decision Tree works:-
we need to learn :-

+ Entropy

+ Information Gain

Entropy

what is entropy?

In the most layman terms, Entropy is nothing but the measure of disorder. Or you can also call it the measure of purity/impurity.

Let's see an example:-

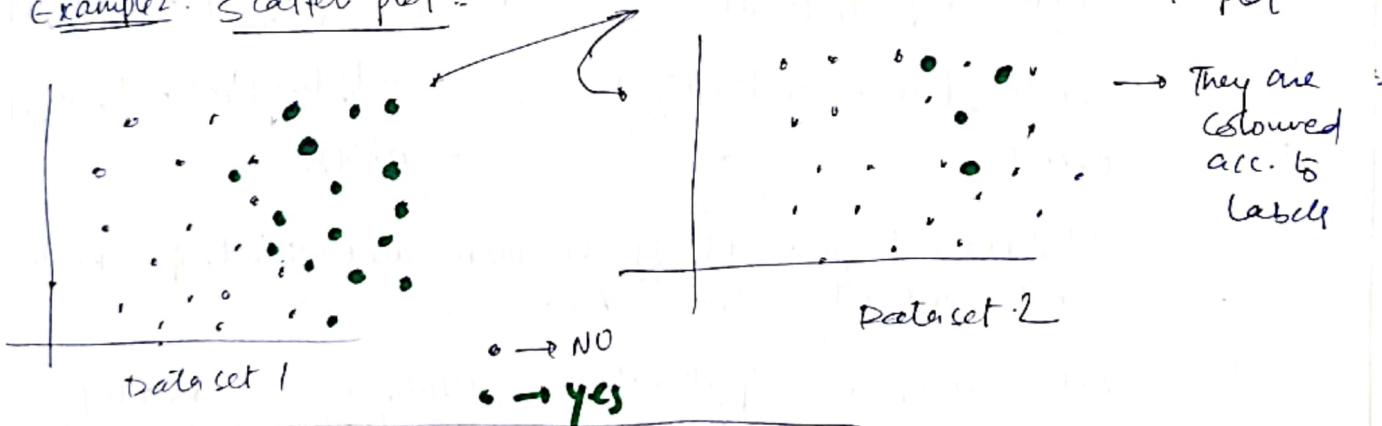
Water has three states:-

ice, water, vapour → ice is solid, water is liquid
vapour is gas → the molecules are most loosely packed.

Hence gas(vapour) has the highest entropy of the three.

Example 2. Scatter plot:

These two are the same scatter plot.



(More knowledge less entropy.)

∴ The dataset jiske hum more confidently predict kar sake hei has less entropy.

Here ∵ Dataset 1 has higher entropy & Dataset 2 has lower entropy compared to Dataset 1.

Quantifying Entropy (How to calculate Entropy)

The mathematical formula for Entropy is:-

$$E(S) = \sum_{i=1}^c -P_i \log_2 P_i$$

Where P_i is simply the frequentist probability of an element/class 'i' in our data.

For e.g.: if our data has only two class labels Yes & No,
 in the output data

$$E(D) = -P_{\text{Yes}} \log_2(P_{\text{Yes}}) - P_{\text{No}} \log_2(P_{\text{No}})$$

Q3. Consider two datasets:

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

If has two classes:-

Yes & No.

$$P_Y = \frac{2}{5}, P_N = \frac{3}{5}$$

$$\therefore E(D_1) = -P_Y \log_2(P_Y) - P_N \log_2(P_N)$$

$$E(D_1) = -\frac{2}{5} \log_2(\frac{2}{5}) - \frac{3}{5} \log_2(\frac{3}{5})$$

$$E(D_1) = 0.97$$



Logit has higher entropy so more uncertainty & hence less certainty than D2.

D3. all entries in output col = No (then we will be most certain & hence it should have least entropy)

$$E(D_3) = -P_{\text{Yes}} \log_2(P_{\text{Yes}}) - P_N \log_2(P_N)$$

$$= -0 \log_2(0) - 1 \log_2(1)$$

$$= 0$$

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

Logit also has two classes
Yes & No

$$(P_Y = \frac{1}{5}; P_N = \frac{4}{5})$$

$$E(D_2) = -P_Y \log_2(P_Y) - P_N \log_2(P_N)$$

$$= -\frac{1}{5} \log_2(\frac{1}{5}) - \frac{4}{5} \log_2(\frac{4}{5})$$

$$= 0.72$$



* Calculating Entropy of a 3 class problem

Page 77

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
30000	30	Maybe
12000	18	No
40000	40	Maybe
20000	20	Maybe

classes : Yes, No, Maybe

$$E(\text{dataset}) = -P_Y \log_2(P_Y) - P_N \log_2(P_N) - P_M \log_2(P_M)$$

$$P_Y = 2/8, P_N = 3/8, P_M = 3/8$$

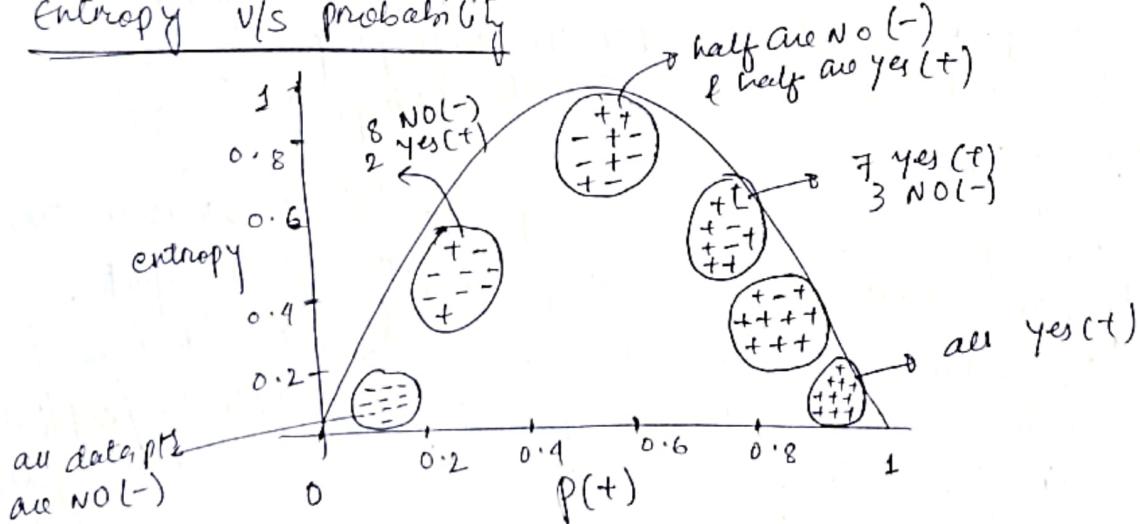
$$\therefore E(\text{dataset}) = 1.56$$

Observations: more the uncertainty more is the Entropy of lesser is the knowledge about that dataset.

- For a 2 class problem [output with only two categories] the min entropy is 0 (if all the datapoints are from one class only) & the max entropy is 1. (if both classes have same number of datapoints)
- For more than 2 classes the min entropy is 0 but the max entropy can be greater than 1.
- Both \log_2 or \log_e can be used to calculate Entropy : we only want to know ki kis dataset mei entropy/jyada hai.

chart-making

Entropy v/s probability



Hence Entropy is highest if $P(\text{Class}) = 0.5$

Q1. if all the class has equal probabilities for 2 class then the entropy is the highest (which is obvious also \rightarrow then we are least certain about a given datapoint that it is of a particular class).

Entropy for Continuous Variables :-

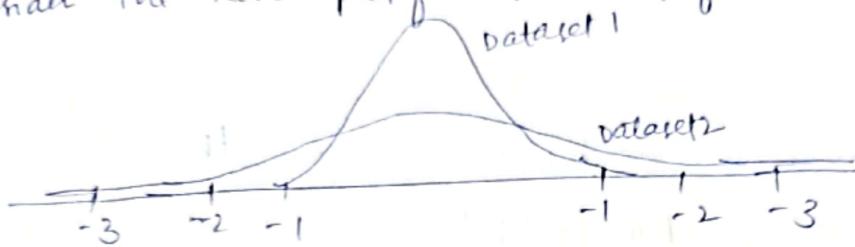
dataset 1.

↳ output col is Numerical
dataset 2.

Area	Builtin	Price
1200	1999	3.5
1800	2011	5.6
1400	2000	7.3
...

Area	Builtin	Price
2200	1989	4.6
800	2018	6.5
1100	2005	12.8
...

Now which of the above Datasets has higher Entropy?
Consider the KDE plot of output col for both the Datasets.



We know jis bhi Dataset ke baare mei jyada knowledge hoga wiska entropy kam hoga. → kam area mei jyada point

Dataset 1 ke jyadatari points $(-1, 1)$ ⚡ hai & hence we have more knowledge about Dataset 1

~~Dataset 2~~ knowledge about Dataset 2 ⚡ hai & hence we have less knowledge about Dataset 2 ∵ datapoint spread hai

Hence Dataset 1 has lesser Entropy compared to Dataset 2.

Hence for cont's variables :- plot the KDE (Histogram → smooth)
higher the peak of KDE → Less the Entropy f
smaller the peak of KDE → higher the Entropy.

Information Gain:

Information Gain is a metric used to train Decision Trees.
Specifically, this metric measures the quality of a split.

The information gain is based on the decrease in entropy after a data-set is split on an attribute/col/feature. Constructing a decision tree is all about finding the attribute/col/feature that returns the highest information gain.

formula for Information Gain for a particular col:-

$$\text{Info Gain} = E(\text{parent}) - \left\{ \text{weighted Average} \right\} * E(\text{children})$$

↓
Complete dataset before split.

↳ after split the weight average of entropy of all the children

underfitting :-

~~Reas~~ Consider the Dataset with cols: outlook, Temp, Humidity, wind, play..
let's calculate the info gain for the outlook col

Page 78.

Step1: Entropy of parent

$$E(P) = -P_Y \log_2(P_Y) - P_N \log_2(P_N) = \frac{3}{14} \log_2(\frac{3}{14}) - \frac{5}{14} \log_2(\frac{5}{14})$$

Step2: we'll divide the dataset based on the ~~outlook~~ ^{= 0.97} ~~outlook~~ col :- (Group by outlook col)

day	outlook	humidity	wind	play	outlook				
					D	O	H	W	P
1	sunny	high	weak	no	3	overcast	high	weak	yes
2	"	high	strong	no	7	"	normal	strong	yes
3	"	high	weak	no	12	"	high	strong	yes
4	"	normal	weak	yes	13	"	normal	weak	yes
5	"	normal	strong	yes					
6									
7									
8									
9									
10									
11									
12									
13									
14									

calculate the entropy of all these children datasets.

$$E(\text{sunny}) = 0.97$$

$$P_Y = \frac{3}{5}, P_N = \frac{2}{5}$$

$$E(\text{overcast}) = 0$$

$$P_Y = 1, P_N = 0$$

$$E(\text{Rain}) = 0.97$$

$$P_Y = \frac{3}{5}, P_N = \frac{2}{5}$$

Step3: Calculated weighted Entropy of children

$$\begin{aligned} \text{weighted entropy} &= \text{prob}(\text{sunny}) \times E(\text{sunny}) + \text{p}(\text{overcast}) \times E(\text{overcast}) + \text{p}(\text{Rain}) \times \\ &= \frac{3}{14} \times 0.97 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.97 \end{aligned}$$

$$W.E. (\text{children}) = 0.69$$

Note:- overcast is a leaf node as its Entropy is zero(0) \rightarrow this data will have no more splitting.

Step4:- Calculate Information Gain:-

$$\begin{aligned} \text{Info Gain for } \text{outlook column} &= E(\text{parent}) - W.E. (\text{children}) = 0.97 - 0.69 \\ &= 0.28 \end{aligned}$$

so the info gain(or decrease in Entropy / impurity) when you split this data on the basis of outlook condition/col is 0.28

Step5:- Calculate the Info Gain for all the cols in the similar manner & whichever column has the highest information Gain (max decrease in entropy) the algorithm will select that col to split the data.

Step6:- Find info gain Recursively.
~~Reas~~ Decision Tree then applies a recursive greedy search algorithm in top bottom fashion to find info gain at every level of the tree. Once a leaf node ($\text{Entropy} = 0$), no more splitting is done on that particular dataset.

Gini Impurity
parameter called

In sklearn DecisionTreeClassifier there is a criterion = {'gini', 'entropy'} & default value is gini

Just like Entropy :- Gini Impurity also tells us about the impurity of the child dataset after split.

formula for a 2 class- Entropy = $-P_Y \log_2(P_Y) - P_N \log_2(P_N)$
 $G_I = 1 - (P_Y^2 + P_N^2)$

Ex:- we have the same Salary | Age | Purchase with two datasets:-

for Dataset 1:

$$G_I = 1 - (P_Y^2 + P_N^2) = 1 - (9/25 + 16/25) = 0.98$$

for Dataset 2:

$$G_I = 1 - (P_Y^2 + P_N^2) = 1 - (1/25 + 16/25) = 0.32$$

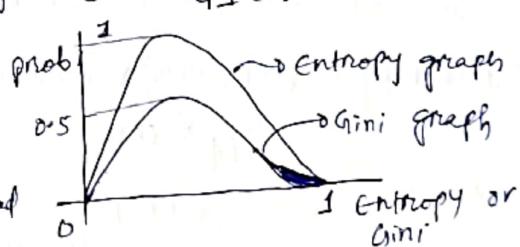
Hence Dataset 2 is more pure than Dataset 1.

Lesser the G_I more pure data is
More knowledge we have - more confidently we can predict & lesser the entropy

Note:- jo kaam hum info Gain ke liye Entropy ke through karte hoo wohi same kaam hum Gini Impurity ke through karte hoo hei. The formula will be Info Gain = $G_I(\text{parent}) - \text{weighted } G_I(\text{child})$

If: $P_Y = 1$ } $E(D) = 0$ & if $P_Y = 0.5$ } $E(D) = 1$
 $P_N = 0$ } $G_I(D) = 0$ $P_N = 0.5$ } $G_I(D) = 0.5$

∴ Gini V/S Prob Graph will be:-



* Default value in sklearn is

Gini ∵ it is computationally fast
(∵ we only need to compute squares instead of logs)

* ∵ for large datasets → we generally use Gini impurity instead of Entropy.
* Entropy is used because the tree by split is sometimes more balanced in Entropy case compared to Gini. (Hence always do hyperparameter tuning for both Gini & Entropy to get the best result).

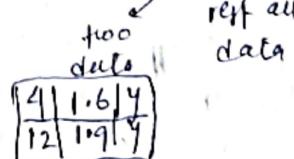
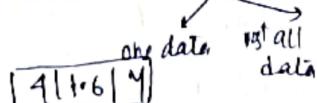
HANDLING Numerical input col.

→ Here if we Groupby 'user rating'
data['user-rating'].nunique = 12 → all unique values.

Now splitting on 12 data we will get lot of subtrees & hence calculation will be difficult
Solution:-

Step 1: Sort the data on the basis of the numerical col. (Here user-rating) → 1.6

Step 2: Split the entire data on the basis of rating > 1.6 or rating ≤ 1.6 again. rating ≤ 1.9 & soon for all

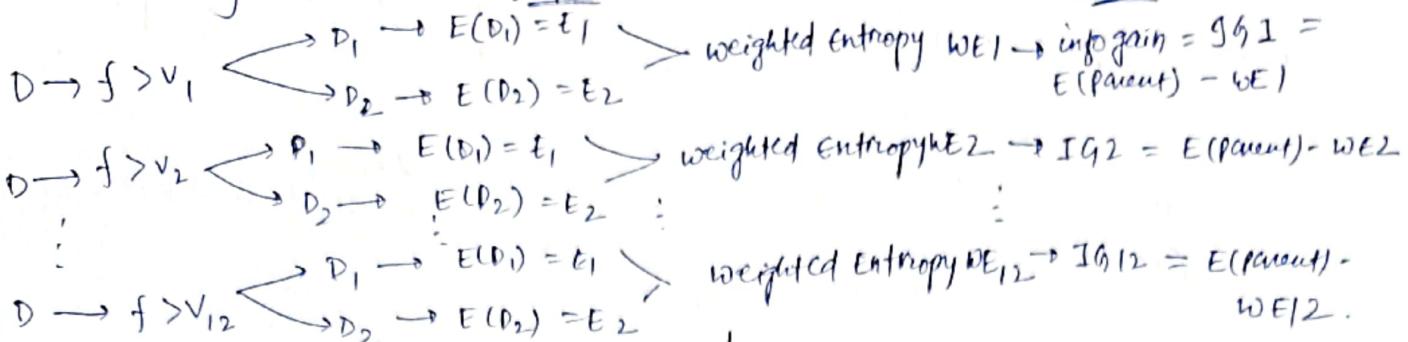


the rows after sorting.

S.NO.	UserRating	Downloaded
1	3.5	
2	4.6	Y
3	2.2	N
4	1.6	Y
5	4.1	N
6	3.9	N
7	3.2	N
8	2.9	Y
9	4.8	Y
10	3.3	N
11	2.5	Y
12	1.9	



So we have :- for every particular value of user rating we split the dataset.



Where: $D \rightarrow$ dataset
 f is feature/col user-rating
 $\& v_i$'s are the values of the various rows of User-Rating.

Step 6: Do this recursively until you get all the leaf nodes.

AFTER this:-

Step 5: Max $\{IG_1, IG_2, \dots, IG_{12}\}$
 say IG_3 is the max

then $f > v_2$ will be the splitting criteria for col user-rating.



Question:- The entire thing looks computationally too expensive. Is it the right way of finding the splitting criteria for Numerical input cols.

Ans:- Yes. Because we are doing this only once & that too on the train data set... we don't do this on the test dataset or on deployment level when a new query point is given. ∴ The train time complexity is still higher but the test time complexity is $\log(n)$ (Tree based).

Hence:- Decision Tree is more apt for categorical Data; if the Data has many Numerical cols then we mostly avoid Decision Tree.

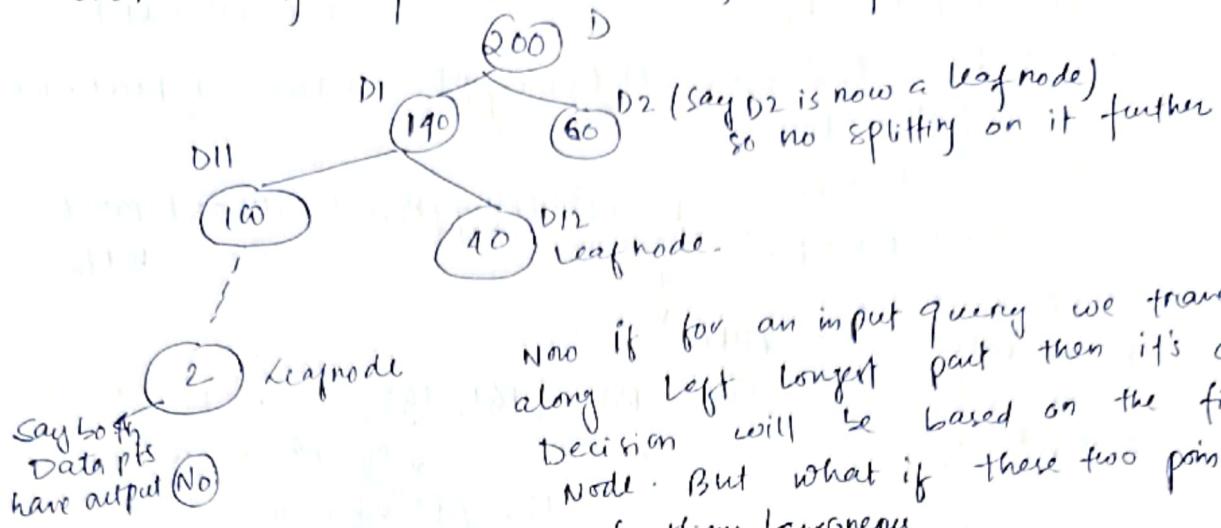
HYPERPARAMETERS (Overfitting & under fitting)

Logit has a tendency to overfit (training data pe acche results deti hain par test data pe nahi).

- Depth of Tree → is the main thing jiske wajah se overfitting ya under fitting hota hai.

Oversplitting → performs well on Training data
performs bad on Test data.

Let's say we had 200 rows in a dataset & on the basis of splitting criteria of a particular column, we split it



Hence: all those ~~else if~~ if else to reach actions → then the system algo will fail.

→ This system will work good on training data (obviously) but on test data it may give error or outlier output even though it is not.

→ This is a clearly cut case of oversplitting.

So: if the parameter max_depth = None (by default) then the Decision Tree classifier runs till it gets all the Leafnodes & in such cases there is a huge chance of oversplitting.

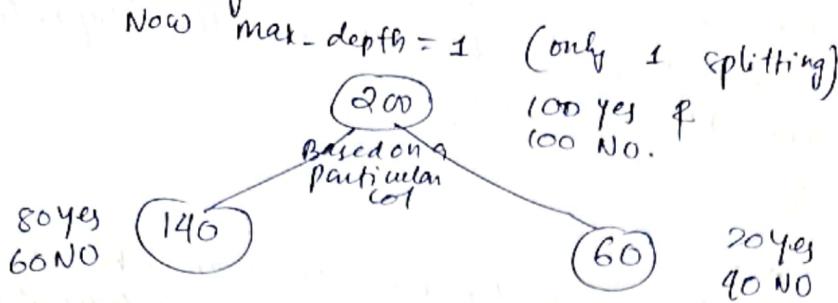
Geometric Intuition of oversplitting:-



→ This is case of oversplitting.

underfitting :-

page 81
overfitting problem



Page 80.

Now we have a new query input & it goes into (140) node
then since it has more yes than NO ∴ the output of the query will be predicted Yes (Even though it may have output some No.)
So we don't want both under fitting (high or None value for max_depth) & over fitting (low value of max_depth) of hyperparameter NoteBook.

for more hyperparameter Tuning:- see again the video Decision Trees - Hyperparameters from 12:00 minutes.
imp. hyperparameters are:- criterion (Gini, entropy)

by default splitter = None → Max-depth (✓)
Min samples split
Min samples Leaf
Max Features
Max LeafNodes
Min impurity Decrease } See the documentation of decision Tree Classifier & Learn about these more.

splitter: for numerical input col → we saw how we choose the splitting criteria → the one with best information gain but sometimes we can keep it random to split the data based on a col Randomly → random reduces overfitting

randomly select karte hoi for training data pe reta acca fit mali kar pata algorithm.

min_samples_split := 10 (say)

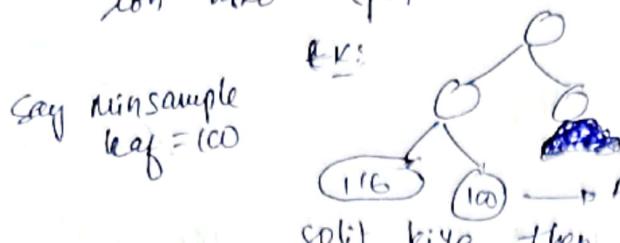
It means suppose we are splitting Dataset now
D1 will be further splitted iff min 10 rows are available in D1 & similarly on D2 & if there is further splitting then each ke child datasets ka splitting shi tab hi hogा if they have atleast 10 (rows/dataset) hai unme otherwise 2 child dataset ka splitting run jaiga.

This is also other way of trimming our Decision

Higher its value → Higher chance of underfitting
Lower its value → Higher chance of overfitting (think).



- Min Sample Leaf :- It is the same as above ~~ki agar~~ ^{ki agar}
min sample leaf \neq ~~will not~~ atleast maki hue in the leaf node
Jab tak split karne hi maki doge.



split kiya then the leaf nodes will have ~~less~~ less than 100 rows. but min sample leaf = 100 says that the leaf nodes must have atleast 100 rows.

Higher value \rightarrow underfit

Smaller value \rightarrow overfit.

used in case of high dimensional data & there is overfitting

- Max features :- we can choose ki kyun Decision Tree ko kitne col done chahte hai to work at every node.
say max features = 3 \rightarrow then it will split ~~any~~ based on the ~~any~~ 3 cols. at every node.

(not the best one)

\therefore if value = 1 then at every node a random col is selected & based on it split is done.

Lo This is done to introduce Randomness \rightarrow hence reduce overfitting.

- Max-leaf Nodes :- How many Leaf Nodes we want in the final decision tree. \rightarrow Higher value \rightarrow overfitting Lower value \rightarrow underfitting

- Min Impurity decrease = α (say)
at every node by splitting we decrease error / impurity / entropy
 \hookrightarrow It means ki agar split karne pe based on a col agar min α impurity ghalga jab hi split hoga wahan mali hoga.

Higher value \rightarrow underfitting

Lower value \rightarrow overfitting.

Decision Tree is imp because it is used in

+ Random forest

+ Bagging

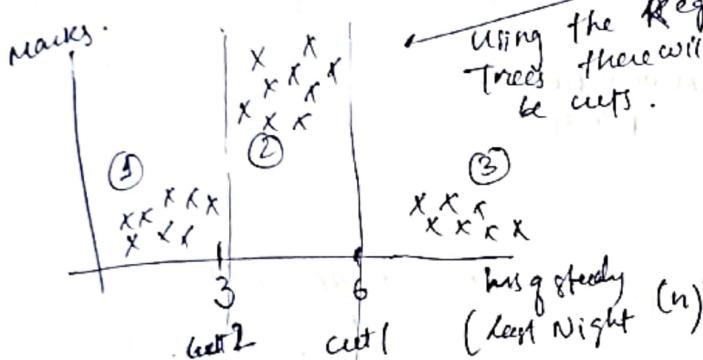
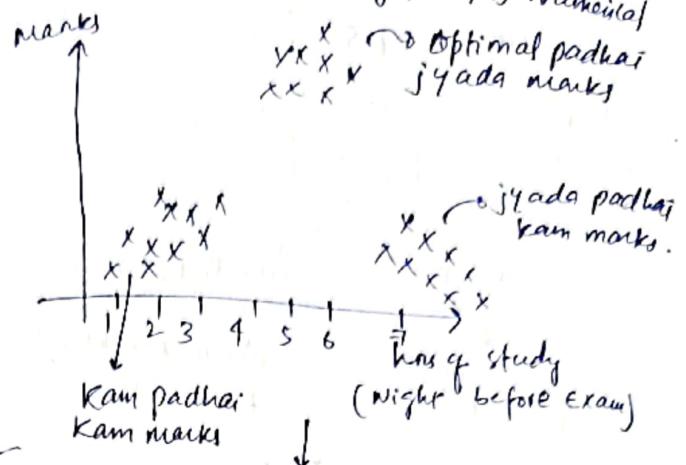
+ Gradient Boosting

{ do we need to know clearly
ki decision Tree ke kaise
hyperparameters ko kaise tune
kare to get desired result.

Regression Trees (Decision Trees on Regression Problem)

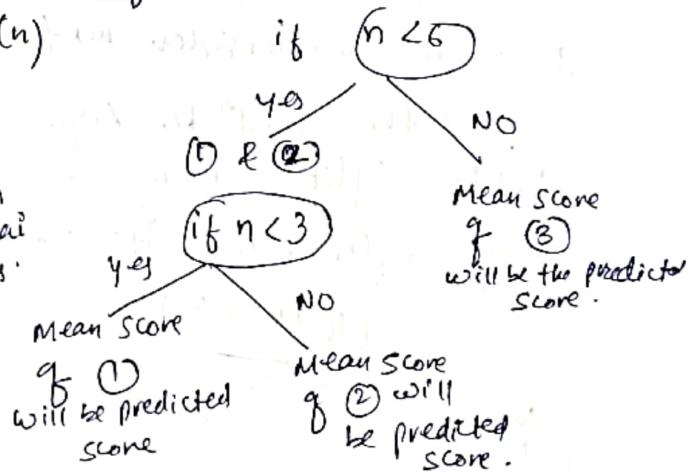
page 81

Where is it applicable?



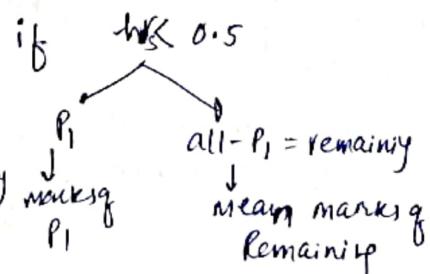
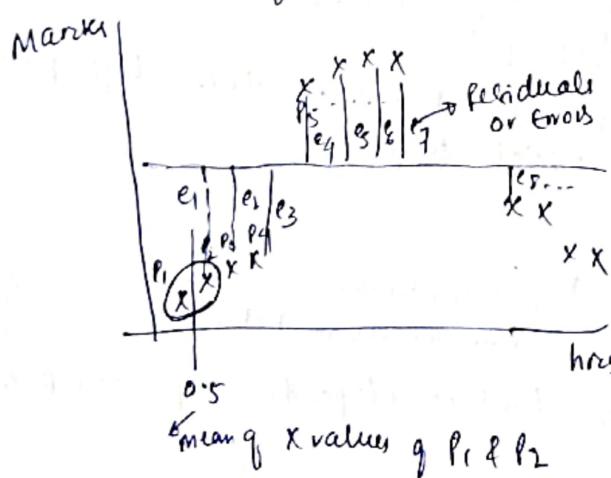
Here Linear Regression will not help. Hence when we use Regression Trees there is a non linear Data.

code will be:



Then splitting criteria visible nahi hai \rightarrow Regression Tree handles

So how Regression Tree will work (step:- select 1st two)



$$calculate: SSE_1 = e_1^2 + \dots + e_n^2$$

sum of squared errors

Now select another two

$P_2 \& P_3 \rightarrow$ mean of their

x -values = 1.5 (say)

& proceed similarly.

Here:

$hrs < 1.5$

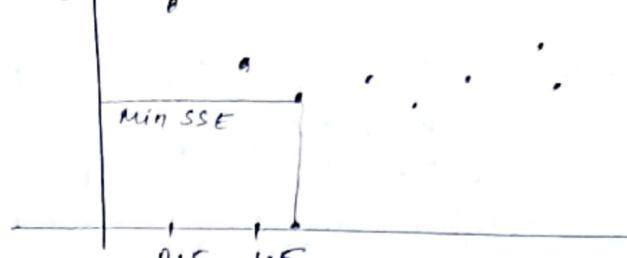
P_1, P_2

mean marks
of P_1, P_2

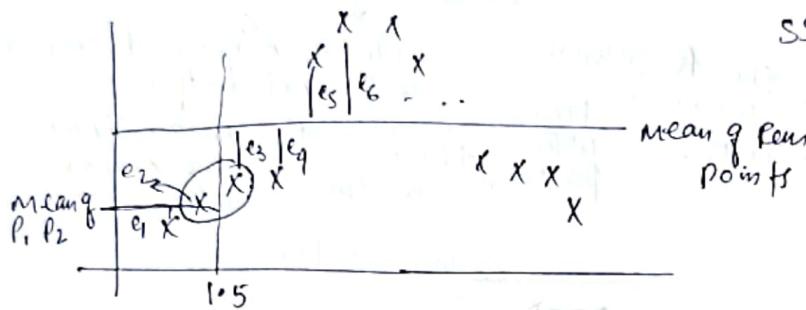
$\text{all } - P_1, P_2 = \text{rem}$

mean marks of
Remaining.

Error
(SSE)



mean of
Selected
two hours



$$SSE_2 = e_1^2 + e_2^2 + e_3^2 + \dots + e_6^2$$

& do on we consider next two points & proceed.

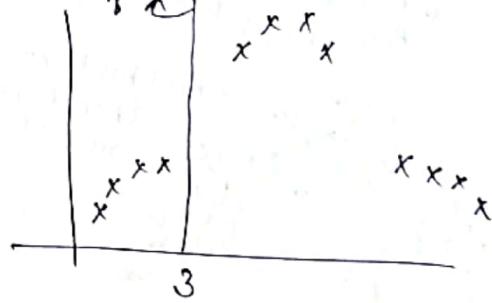
∴ jis bhi cond ke liye SSE minimum hai woh hamara
Pehla split banjega.

so say after first iteration

if $hrs < 3$

Now in both
parts repeat the
process

minimum hai woh hamara
if $hrs < 3$



Note:- if we keep on continuing then we will split each point.
(overfitting) → we only need to split in clusters.

for this if a child has $\leq \underset{\text{threshold}}{x}$ # of elements we won't further split.

Say in above case we set it 4 & hence the left part
of the subtree won't be splitted further.

if $hrs < 3$

=
Marks
Mean
of left part

split continue.

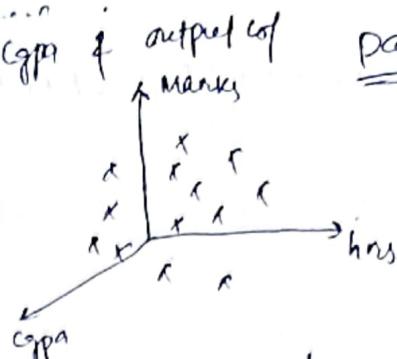
using the above method.

generally the threshold is 20-25 but it depends upon the data.

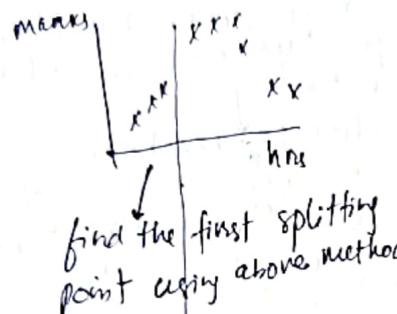
* what if we have input cols hours/cgpa & output col marks

Page 82.

we use plane to cut/split
& in further higher dimension
we use hyperplanes.



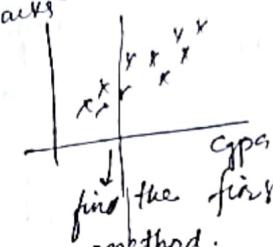
we do independent analysis! - hours/marks & cgpa/marks.



find the first splitting point using above method

if we have min sum of squared Error = SSE_{hr}

Now there are two cases:-



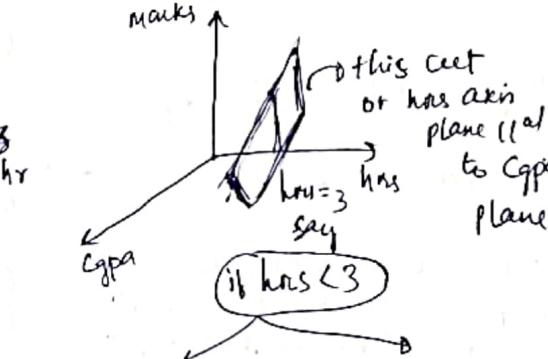
find the first splitting point using the above method.

if we have min sum of squared Error = SSE_{cgpa} .

& $SSE_{hr} > SSE_{cgpa}$

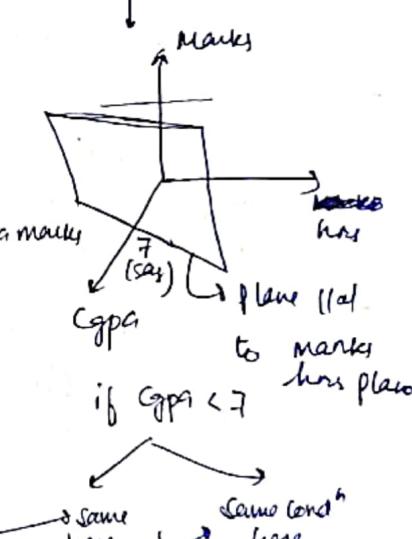


in this case



if here we have more than we split or stop

same here



if $cgpa < 7$ plane ||| to marks hrs plane

same here same cond here

In Decision Tree Regressor

there are some changes in the hyperparameter

criterion we have for splitting is MSE (Mean Squared Error) & MAE (Mean Absolute Error).

Rest are same as Decision Tree classifier & Friedman-MSE

ENSEMBLE LEARNING

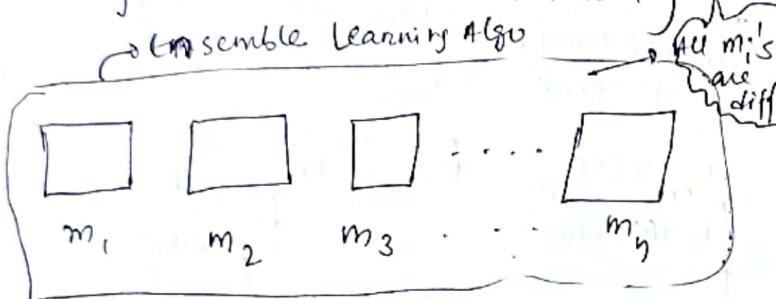
→ it signifies a collection of different multiple ML algos to make a new/better model.

Wisdom of the Crowd: → in KBC → audience poll, almost all times the audience knew the answer.

Core idea of Ensemble Learning:

Prediction stage
in Ensemble Learning

Say we have an Ensemble Learning Algorithm (Collection of smaller ML Algos / base models)



→ Product review \Rightarrow humne reliability milta hai ki humne product kharidne chahiye ya nahi in online Marketing.

Bahot logon ne kharida but bas ek ne 4.5 rating diya (we'll not trust much)

But agar bahot logo ne 4.5 rating diya toh we'll trust it & buy it.

Similarly IMDB Rating for movies.

→ Democracy (jyadatar log agar ek party ko support kar rhe hai toh woh sahi hi hogi).

→ Experiment done → Animal on stage → its weight is asked by people everyone guessed it (no one was accurate) but the mean of everyone's guess was almost the same.

↓
Somehow crowd ko answer pata hota hai.

↓
Jissi cheez pe based hai.

Ensemble Learning

↓
i.e. crowd ko single et jyada pata hota hai.

↓
Crowd should be mixed (not all g. Some fysx like software engineer)

Say: $m_1 \rightarrow$ Linear Reg (all different)

$m_2 \rightarrow$ SVM

$m_3 \rightarrow$ Decision Tree & so on.

Or: $m_1 \rightarrow$ Linear Reg but during the training they were given different data
 $m_2 \rightarrow$ Linear Reg & $m_3 \rightarrow$ Linear Reg & so on & hence variety is created.
Or we can use combination of above two.

Now suppose we have a new query input: we have CGPA & IQ if have to predict placed or not. (Classification problem)

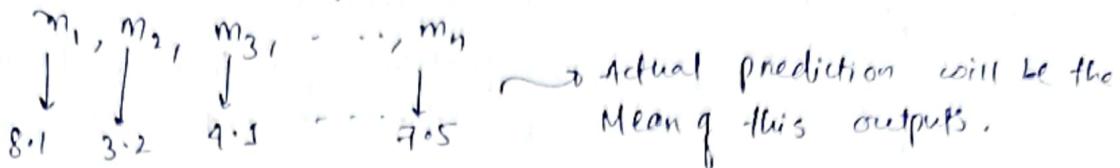
This query point is given to every model is

Predict	$m_1, m_2, m_3, \dots, m_n$	→	Ab Democracy wala logic lagega (majority count) of wohi actual prediction
Y	N N Y Y	→	



for a regression input query : (placement in books) Page 83.

the input is again given to all the models:-



Types of Ensemble Learning

4 types that are used most are

Voting Ensemble

In this the base models (m_i 's) are different Algo

e.g. $m_1 = \text{SVM}$

$m_2 = \text{Logistic Reg}$

$m_3 = \text{Decision Tree}$.

Now the same training data set is given to all to train & an input query point is given to all the models & then the majority count (for classification) or Mean (for regression) is predicted.

Yaha \rightarrow han Algo (models ko Bonakar value diya jaa rha hai)

Say each of them will have 500 datapoints then by Bootstrapping we randomly choose 500 datapoints from Data & make it D_1 , again out of data (1000 rows), 500 random Datapoints $\rightarrow D_2$ & again out of random Datapoints $\rightarrow D_3$ & Now for training the ensemble model: $D_1 \rightarrow m_1$; $D_2 \rightarrow m_2$; $D_3 \rightarrow m_3$

Note: Bahot high prob hai ki D_1, D_2 &

D_3 are different.

Now a new query point is given to all m_1, m_2 & m_3 & majority count or mean is predicted finally.

Random forest is special case of Bagging where the base models are decision trees

Bagging

stands for Bootstrap Aggregation

Random forest

Here the models are the same but different training data is given to each say we have m_1, m_2, m_3 say SVM.

Now in Data we have 100 rows & now we create variety of Data, by Bootstrapping D_1, D_2, D_3

Boosting

AdaBoost

Adaptive Boosting

Gradient Boosting

XG Boost

(Extreme Gradient Boosting)

in Boosting

again the models are same say

$m_1 = m_2 = m_3 =$

Linear Reg.

Here:- the Data

is given to m_1 ,

& m_1 keeps track

g kaha kaha wne

galtiya ki p

yahi data with the

galtiya m_2 ko explain

karke pass kar de

hai & yeh m_2 galtiya

par ushi results

dene lagta hai pan

ise kuchki galtiya

lohi hai & wo m_3

ko balya jata hai

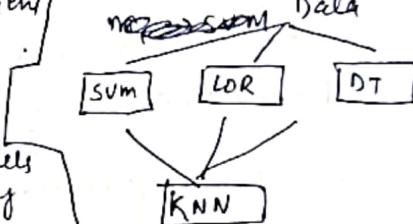
This is Boosting

hence finally the model

makes lesser mistake.

Stacking

It is similar to Voting Ensemble. But yaha har model ko kuch weights assign honge by a different ML Algo & hence har model ke value in prediction will not be same



SVM, LOR (Log. Reg.) & DT will be trained on the Data & KNN will be trained on

sum	LOR	DT	Average / Final Prediction
1	0	1	1
0	1	0	0
-	-	-	-
-	-	-	-

↓
This KNN algo will give weights to the above models.

Isse ye patha chalta hai ki kongu model yada accha hai in prediction.

Ensemble Learning in classification problem pick prediction from multiple models & creates a better model by removing the noise in their prediction using majority count.

Same thing happens in Regression problem - just that the noise in individual predictions is removed by taking mean & getting a better model.

Benefits of Ensemble Learning :-

- Improvement in Performance
- Bias, Variance Ko reduce karta hai
 (we need Low Bias & Low variance)

Say Decision Tree has Low bias but High variance then use Random forest → it will give Low bias + Low variance.
- if model has:- High bias + Low variance → use ensemble learning we'll get Low bias + Low variance.
- Robustness — Data mei bahot jyada change shi kardoge toh toh kisi performance jyada change / fluctuate nahi hota.

Disadvantage is :-

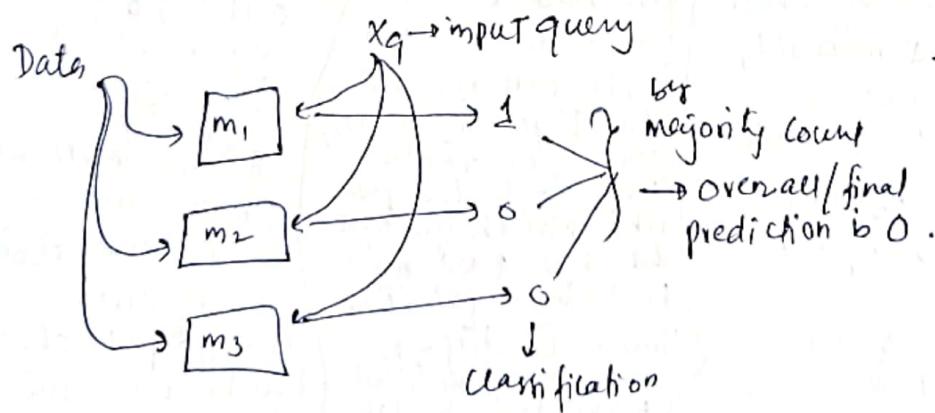
we are training multiple models & hence the computational complexity increases.

* When to use Ensemble Learning?

Ans:- Always.

VOTING ENSEMBLE

- + core idea / intuition
- + Voting classifier (for classification problems)
- + Voting Regressor. (for Regression problems)



Agar Regression problem hota toh jeenka Mean return karta as final prediction.

Why Voting works?

Page 84.

Say we have 3 models in Voting Ensemble $m_1, m_2 \& m_3$

Now say accuracy of them are :- 0.75, 0.6, 0.55 resp.

then how do they combine in voting Ensemble to give an overall accuracy that is better than all the individual accuracy (say 0.88) ↓

This we need to discuss

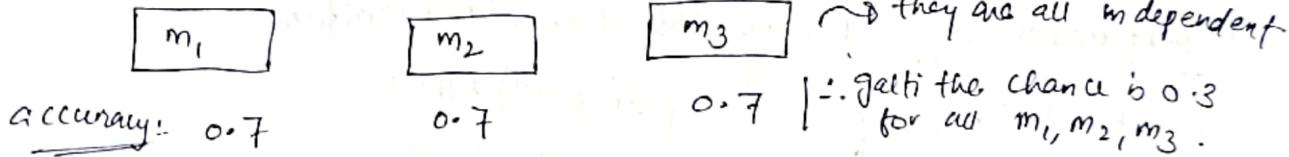
before that let's discuss the assumptions behind Voting Ensemble:-

- All the models are independent (kind of different)
- Same individual models ka minimum accuracy should be $\geq 50\%$.
nahi toh overall accuracy $\cdot \min\{m_1, m_2, m_3\}$ \Rightarrow koi karna hoga.

Why voting works is :- When we have a crowd then all the individuals are from different backgrounds & have different knowledge of hence ek ki kamai ~~ko~~ dusra chupi leta hai & overall performance is better.

These are called
~~base~~ model.

↓ Mathematically proving this (probability)



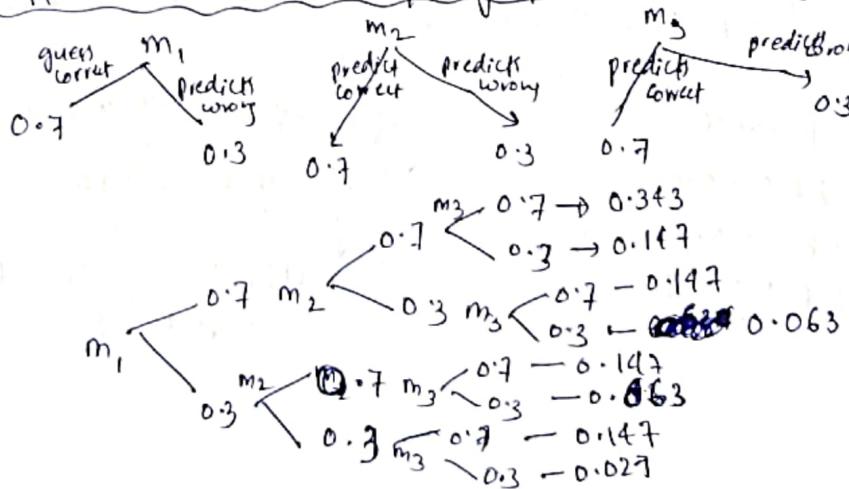
\therefore jaldi the chance is 0.3 for all m_1, m_2, m_3 .

Now ab agar iska voting classifier banaye then majority jo bollega rochi final prediction hoga.

To show accuracy of voting classifier ≥ 0.7 .

Suppose we have a new query point:-

→ since they are independent



Now:- Voting classifier will predict yes if there are atleast two yes (Majority count). Hence the prob of that is :-

$$= \frac{YYY + YYN + YNY + NYY}{16} = 0.343 + 0.147 + 0.147 + 0.147 = 0.789$$

Hence the prob to predict yes by the voting classifier is 0.789

Also it can be proven that if individual accuracy is ≤ 0.5 then the overall performance will be worse than the $\min(m_1, m_2, m_3)$ accuracies. Take accuracy of m_1, m_2 & $m_3 = 0.3$ & check using the same above method.

Classification Problem using Voting Ensemble

Voting Classifier

voting classifier using

we have discussed the core idea of majority count before. see this video again from 3 minutes to understand & visualise again.

hyperparameters of sklearn.ensemble.VotingClassifier

→ estimators : list of models ('fit', model) type.

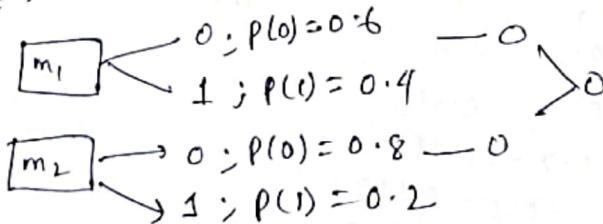
→ voting : { 'hard', 'soft' } ; default : 'hard'

if hard → uses predicted class labels for majority rule

if soft → predicts the class label based on the

argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

Example:-



Hard:

Soft

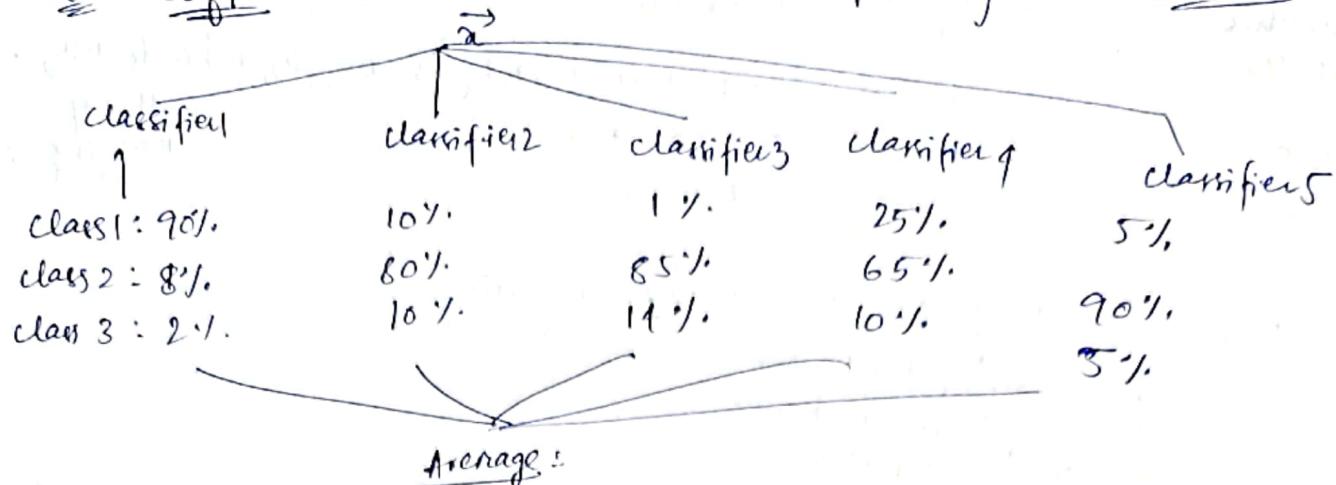
$$\text{avg prob of } 0 = \frac{0.6 + 0.8}{2} = 0.7$$

$$\text{avg prob of } 1 = \frac{0.4 + 0.2}{2} = 0.3$$

\therefore output = 0 ••• avg prob of 0 > avg prob of 1.

Ex:- soft :- (multi class classification problem)

Page 85



Generally speaking softvoting ka result better aata hai
hard voting & But run both & see kiene better aata hai.

* for concentric Dataset:-



→ using Log. Reg.,
Gaussian Naive Bayes,
& Random Forest
with soft voting gives better
result than
using the same models
with hard voting.

in soft voting the decision
boundary ka smoothening
hota hai & hence sometimes
better result de deta hai.

the hyperparameters
one

estimators list
& weights

Jupyter Notebook

Voting Regressor (For Regression Problem)

→ we already know how it works.

the final output is the mean of the each individual output

Jupyter Notebook

in this
all the
base model
are same
Algorithms.

Bagging Ensemble

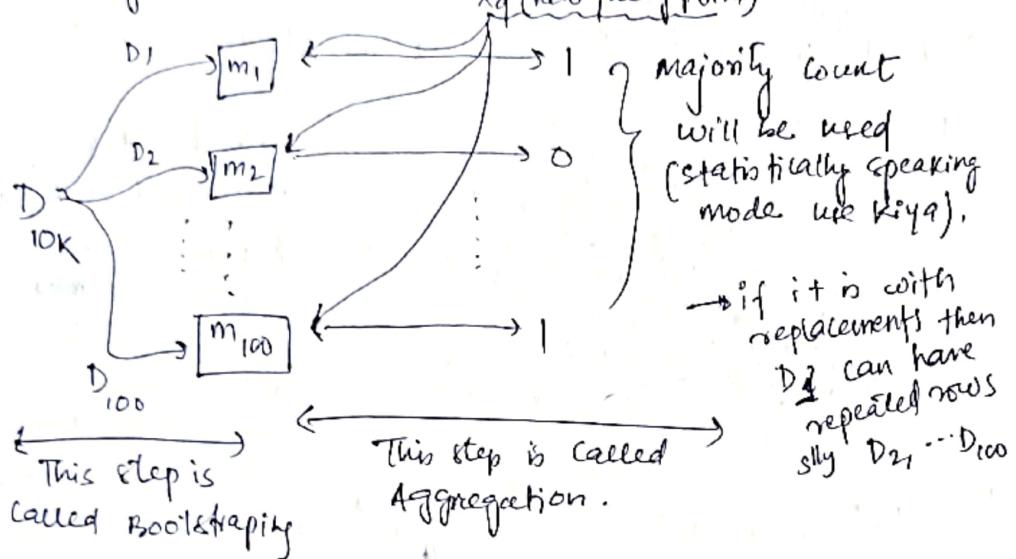
- ↳ Core idea / intuition, why use Bagging, when to use,
- ↳ Bagging classifier
- ↳ Bagging Regressor

Types of
Bagging.

Bagging \rightarrow Bootstrapping + Aggregation.

Given population m_{100}
Samples drawn from
randomly (Sampling)

Suppose we have a dataset with 10000 rows & we want to perform classification problem on it using Bagging.
say we have 100 base models :- train model along Data set
Tree in stages.
(Say 1000 rows per)



majority count
will be used
(statistically speaking
mode like Kya).

→ if it is with
replacements then
 D_i can have
repeated rows
like D_1, \dots, D_{100}

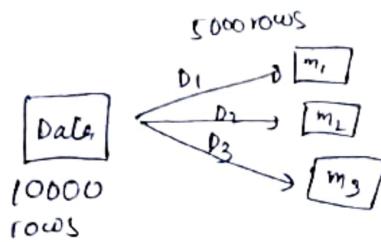
This is done by
Bootstrapping :-
randomly selecting
1000 rows from
 D (sometimes
it can be with
replacement &
sometimes without
replacement).

* Why use Bagging? Why it works?

→ It is based on Bias variance tradeoff (we need Low Bias & Low variance)
But they are inversely prop.

In Bagging we take Algos (base modes) that are Low bias +
high variance (overfitting) ((Ex. Decision Tree with max_depth = n-1))

Now Bagging will reduce Variance



Now if we say change 500 rows & again train then these 500 rows in the distributed Randomly. Again each Ex. like m_1 , but due to random distribution among all three models, the change in model is considerable & similarly. On these new set of data also the model performs completely diff from original with low neighbour value.



Hence jo resultant model hai utka variance is

Page 86.

low (low variance Matlab :- Agar data mai changes kardo
tak bhi ~~model~~ ke output/performance mai jyada changes nahi
aale i.e. model consistent Result data hai)

& how bias (accuracy) doo wo ths hi hence overall
Resultant model low Bias + low Variance ban jata hai.
Hence we use Bagging.

→ When to use Bagging?

Always, whenever you are dealing with low Bias + High variance
model.

↳ implementation of Bagging is using as
Decision Trees, base models →
Random Forest

for Classification
problem.

Jupyter Notebook

(General Base models are:-
Decision Trees, KNN & sometimes SVM)

Bagging classifier

→ Base estimator/model (default is Decision Tree)

→ # of estimators/models

Sampling

→ max_samples → How many rows dena chahie hai.

Not sampling
of rows
with replacement

→ max_samples → Mean Sampling with replacement

→ Bootstrap_sampler = True
ie D_i can have repeated rows.

→ max_features → Kitne features ko use karne chahie hai.

→ Bootstrap_features → Kitne features use karne chahie hai.

These are hyperparameters
that we can tune.

isfit = False

mean sampling

without replacement

means 'PASTING'.

D_i 's will not handle
repeated rows.

RANDOM SUBSPACES

→ max_samples = all rows

Bootstrap_sampler = False

max_feature = not all cols
(say 2 if there are 5 cols)

Bootstrap_features = True.

↓ sampling on columns

This is done when we have
large no. of rows like on
MNIST dataset.

RANDOM PATCHES

(both Row & col sampling)

Say max_samples = 50 (50 rows for each model)

Bootstrap_samples = True (with replacement)

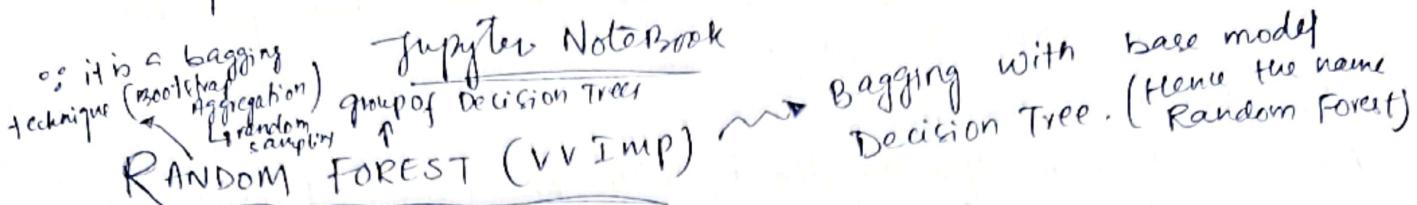
max_features = 1 (say there were 3 cols)

Bootstrap_features = True.

Jupyter Notebook.

Bagging Regressor (for a Regression Problem)

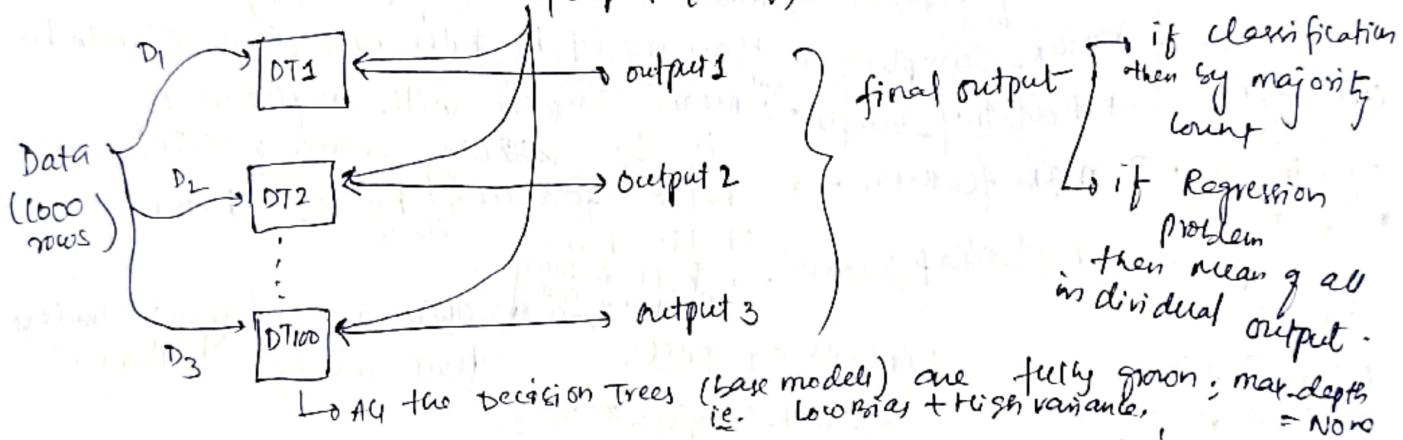
→ In the Aggregation step → the final output is the mean of all the individual outputs & the Bootstrap step is the same as Bagging classifier.



- This along with Gradient Boosting & XG Boosting ka performance in almost any problem is very good.
- It can be applied to both classification & regression problems

→ Random forest is such an Algorithm jisko out of the Both - bohot jyada fine karte ki jarurat nahi hai. i.e. hyperparameters bohot jyada change nahi kriye kaise fab bli iksa performance kafi accha rehta hai.

x_q (input query)

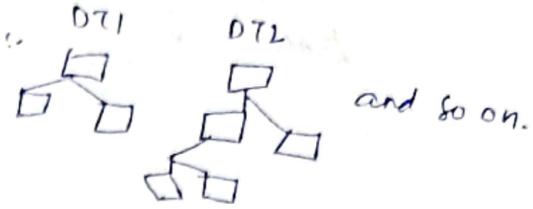


D_i 's are created either by Row Sampling, or by Column Sampling or by a combination of both! with or without replacement just like we did in Bagging

→ Bootstrapping

→ Note: say we are making D_i 's by row Sampling with Replacement:-
Say for D_1 :- we need 500 random rows:- Row 1 is selected then put back, again Row 237 is selected & put back & again Row 1 is selected & put back & so on... till D_1 has 500 rows & like this D_i can have repeated Rows due to Replacement only other D_i 's can have repeated rows.

Since base decision tree (base models) ko alag- Page 89.
alag data diya ja raha hai toh ho ~~89~~ kila hai
ki har decision tree alag alag different.



Jupyter Notebook.

Bias-Variance of Random Forest:-

for any ML model, we want to keep Low Bias & Low Variance.
But the problem is that they are inversely proportional.
(It is called Bias-Variance tradeoff)

most ML Algos are :-

Low Bias + High variance

- Fully grown Decision tree
- SVM
- KNN (When # of neighbours is very low)

High Bias + Low variance

- Linear Regression
- Logistic Reg. (sometimes)

* Random forest converts low bias + high variance Algo to
low bias + low variance.

* we saw earlier how it happens i.e even if we change some data points in original data, set them (unrelated) due to bootstrapping the new data points get's distributed into (overfit) all the base models & not go to only one base model & hence the performance is not affected much, hence on a new dataset too, the performance is same, hence the overall model becomes low variance. since the impact of the noisy data introduced is distributed among the base models (Decision Trees)

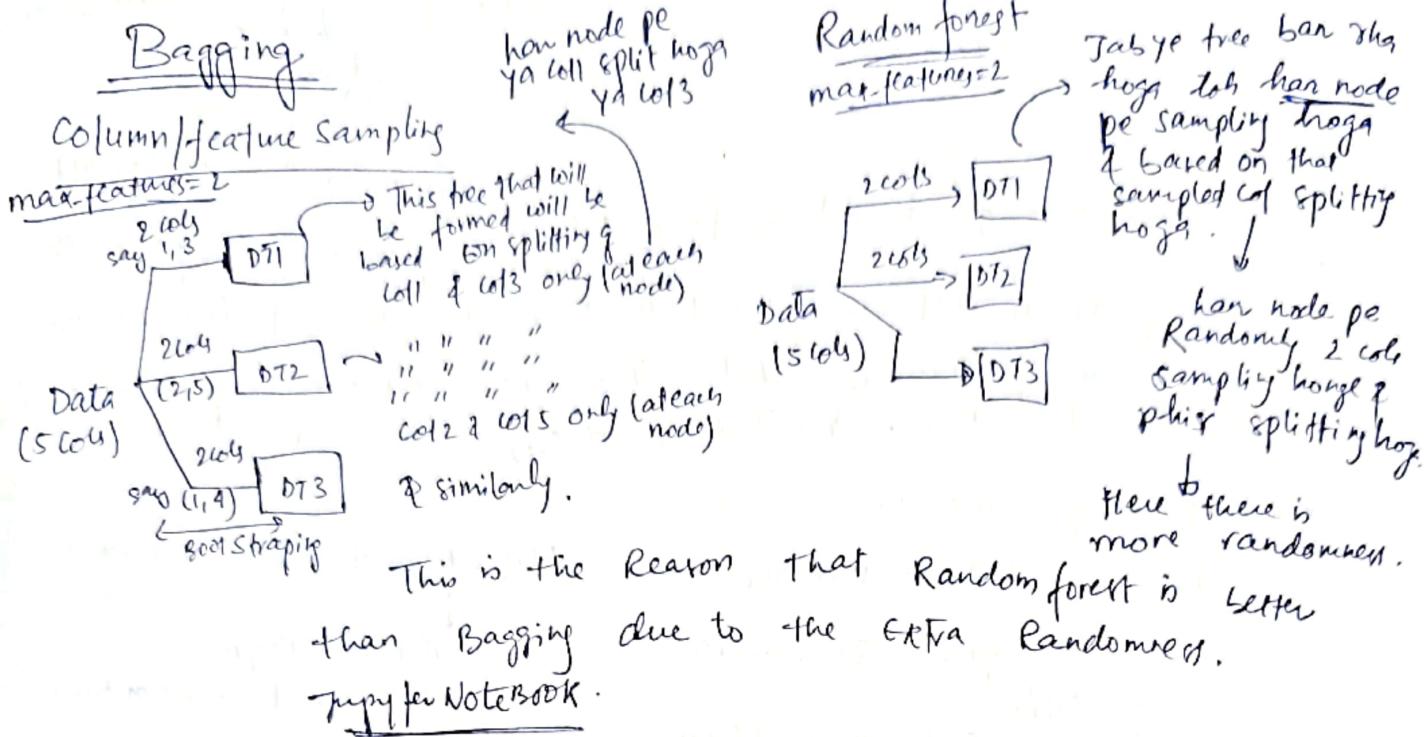
Jupyter Notebook.

Bagging V/s Random Forest:-

The Differences:-

- In Bagging the Base models/estimators can be Decision Tree, KNN, SVM (or some other also). But in Random forest the Base models is Decision Tree. There is a separate class for Random forest in sklearn & different class for Bagging.

Ques :- Bagging with base models — Decision Tree → Is it a random forest → Ans: No.
 The difference is in feature Sampling (max-features parameter)



Hyperparameters in Random Forest Classifier :-

IP & Similar hyperparameters in Random Forest Regressor.

Hyperparameters to tune Random Forest :-

- number of estimators :- decides the no. of decision trees to be used to make the random forest. default = 100
- max-features :- # of columns we want to use for every node creation (splitting)
 Read this in Documentation also (There are more).
 say there are total 5 cols & we choose max-feat = 2 then at every node creation two cols will be randomly selected and then from that two a col will be selected for splitting.
- bootstrap : if we want to select the rows/columns randomly or not - (default: True); if false the whole dataset is used for each decision tree.
- max_samples : # of rows we want to give to each decision tree - (if bootstrap is True)
 - none (default) then use X.shape[0] samples
 - if int then jo diya hai vto use karo
 - if float (0,1) then 0.7 means 70% of rows use karo.

Generally
50% to 75%

- criterion :- gini (default) or entropy
- max_depth : int, None (default)
↳ fully grown
- already discussed earlier
min_samples_split → if the samples agar hai node mei tohi further split hoga.
- min_samples_leaf → split karne ke baad agar leaf moi atleast itne samples takeye toh hi split hoga.
- max_leaf_nodes → kitne leaf nodes wo want to keep in each Decision Tree.
- min_impurity_decrease :- agar node ko split karne pe atleast itna impurity ke hogi toh hi split hoga.
int, None (default)
- n_estimators :- (multi processing), the # of jobs to run in parallel. fit, predict, decision_path & apply are all parallelized over the trees.
- None means : 1
→ means : all processors.
- random_state :- controls both the randomness of the bootstrapping sample (rows) used when building trees (if bootstrap = True) and the sampling of the features to consider when looking for the best split at each node (if max_features < n_features)
↳ total no. of cols
- warm_start (default = False)
used when stages mei model ko train karne hai (when set to True, reuse the state of the previous call to fit & add more estimators to the ensemble, if false -- just fit a whole new forest).
- class_weight is used :- when you have imbalanced dataset.

In Random Forest Regression:- the changes in hyperparameters are -
Criterion: {'mse' (default), 'mae'} ; Rest all are same as Classifier.

Jupyter Notebook.

OOB Evaluation :-

To out of Bag

in sampling with replacement (so repeated rows are possible)

in Random Forest :- While Bootstrapping there might be some rows which don't go to any decision trees (base models). These unselected rows are called out of Bag Samples.

and this out of Bag samples can be used for testing the performance.
model once it is trained.

statistically speaking: ~37% rows are OOB samples in Bagging or Random Forest.

Jupyter Notebook.

Feature Importance using Random Forest & Decision Trees ... how is Feature Importance calculated?

Feature importance:- Konsa feature/kojii jyada important hai output predict karne ke liye. in a particular algorithm. & then we keep the most important feature & remove the rest.

Jupyter Note Book.

How is feature importance calculated.

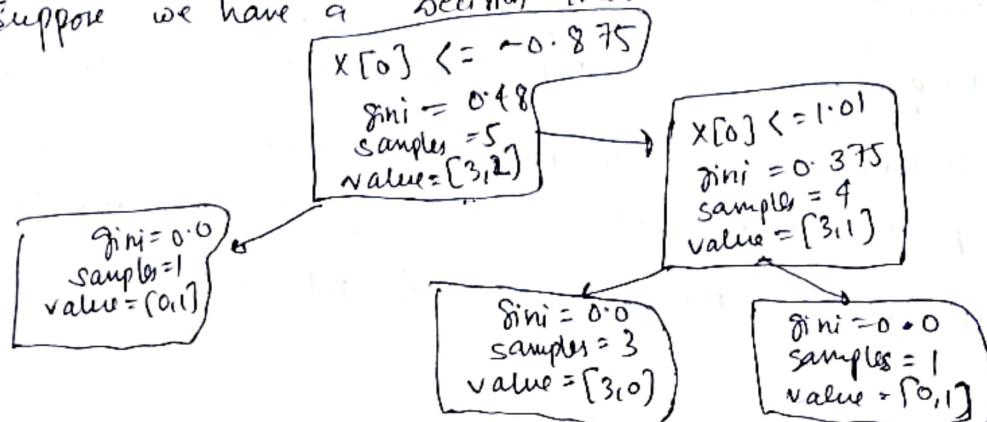
+ to understand how it is calculated in Random Forest we need to know how it is calculated in Decision Tree.

In documentation of DecisionTree classifier:- there is an attribute:- feature_importances_

- + Returns the feature importances
- + The importance of feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as Gini importance.

Jupyter Note Book.

Suppose we have a decision tree:-



Now:- kisi bhi coefficient ka importance

Page 91.

Calculate karne hai using Decision trees.

~~Hg rows~~ in node n_i of that node (Gini value) \rightarrow grows in right node g.v.

$$n_i = \frac{N_{\text{tot}}}{N_{\text{node}}} \left[\text{Impurity} - \left(\frac{N_{\text{left}}}{N_{\text{tot}}} \times \text{right-impurity} \right) - \left(\frac{N_{\text{right}}}{N_{\text{tot}}} \times \text{left-impurity} \right) \right]$$

Lenode & Raimportance picarein data & Hg rows in node n_i . Gini value of right node of n_i

$$\text{fI}_k = \frac{\sum n_i}{\sum n_i}$$

feature imp of k^{th} col
i.e. nodes on which splitting occurred.

In the diagram we have two cols $X[0]$ & $X[1]$.

Now to find feature importance of $X[0]$, we need to find those nodes in the diagrams/tree jispe splitting $X[0]$ cat ke wajah

et hui hai. $\rightarrow X[0] \leq 1.01 \rightarrow$ only 1 node.

& iss node the importance find karne pehle

$$X[0] \leq 1.01$$

$$\text{gini} = 0.375$$

$$\text{Samples} = 4$$

$$\text{value} = [3, 1]$$

Hence then feature importance of $X[0]$ = $\frac{\text{sum of all node importance of those nodes jispe splitting } X[0] \text{ ke wajah}}{\text{sum of all the node importance jispe splitting those nodes jispe splitting } X[0] \text{ ke wajah}}$

$\sum \text{node importance of those nodes jispe splitting } X[0] \text{ ke wajah}$

$\sum \text{node importance of those nodes jispe splitting } X[0] \text{ ke wajah}$

* node importance of $X[1] \leq -0.875$

$$= \frac{5}{5} \left[0.48 - \frac{4}{5} \times 0.375 - \frac{1}{5} \times 0 \right] = 0.18$$

* node importance of $X[0] \leq 1.01$

$$= \frac{4}{5} \left[0.375 - \frac{1}{5} \times 0 - \frac{3}{5} \times 0 \right] = \frac{4}{5} \times 0.375 = 0.30$$

Now:- feature importance of $X[0] = \frac{0.30}{0.30 + 0.18} = 0.625$

$$\& " " " X[1] = \frac{0.18}{0.30 + 0.18} = 0.375$$

Now calculating feature importance in Random Forest is not that difficult.

In random forest :- flat decision Tree (there are multiple), har col/feature ka feature importance calculate karta hai

cup
ish
ki
ab

Suppose there are 10 Decision Trees in a Random Forest, then each decision tree (col 1) has feature importance (col 2) considering the splits made by each tree. Then actual/final feature importance of col 1 = Average of feature importance in each decision tree.

Warning: impurity based feature importance (above) can be misleading for high cardinality features (many unique values). For such features we use `sklearn.inspection.permutation_importance` as an alternative to calculate feature importance.

Boosting Ensemble

- + Adaboost
- + Gradient Boosting
- + XG Boost

AdaBoost: opposite :- strong learners - jiska accuracy badi hai 95%, 99% etc.

- Weak learners :- Aise ML Algo jiska accuracy 50% just thora sa jyada hota hai.

In Adaboost :- we combine/multiply multiple weak learners to make a bigger & better model.

- Decision Stumps :- It is a type of weak learners.

It is a decision tree whose max-depth = 1. In Adaboost although we can use any weak learner, but 99% of the time we will use decision stumps.

- +1 and -1 :- Generally in Binary classification problem the classes are 1 & 0; here in adaboost the classes are 1 (for pos points) & -1 (for neg points).

↳ pts in +ve Region.

↳ pts in -ve Region

The Big Idea of AdaBoost:

^{It is a} stagewise additive method.

page 92

- Not placed

+ placed

29	<table border="1"> <tr> <td></td><td>+</td><td>+</td><td>-</td></tr> <tr> <td>+</td><td>-</td><td></td><td>+</td></tr> <tr> <td>+</td><td>+</td><td>-</td><td></td></tr> </table>		+	+	-	+	-		+	+	+	-		cgsa 29 placed
	+	+	-											
+	-		+											
+	+	-												

Say we have 3 base models (Decision Stumps)

Step 1:- first decision stump will be done :- & the splitting will be done based on that column jisse entropy for representing the nodes will be jyada hoga

splitting	<table border="1"> <tr> <td></td><td>+</td><td>+</td><td>-</td></tr> <tr> <td>+</td><td>-</td><td></td><td>+</td></tr> <tr> <td>+</td><td>-</td><td>-</td><td>+</td></tr> </table>		+	+	-	+	-		+	+	-	-	+
	+	+	-										
+	-		+										
+	-	-	+										



This splitting made some mistakes & when passing this data to

the next decision stump, we will do upsampling i.e. we will ask the next decision stump to value/focus more on the mistakes of this decision stump (like three + are in - region). And also based on the performance of this split a weight (α_1) is assigned to it. We will use α_1 in prediction of final output k_1 is decision stump ka contribution weight kaha hogega in final output.

Step 2 - 2nd Decision Stump:- focussing more on first decision stump mistakes makes a split such that the entropy is

possible	<table border="1"> <tr> <td>+</td><td>-</td><td>+</td><td>+</td><td>-</td></tr> <tr> <td>+</td><td>+</td><td>-</td><td>+</td><td></td></tr> </table>	+	-	+	+	-	+	+	-	+	
+	-	+	+	-							
+	+	-	+								



is ke performance ka weight iska weightage calculate hoga. Lekin we see isne decision stump 1 ke mistakes toh

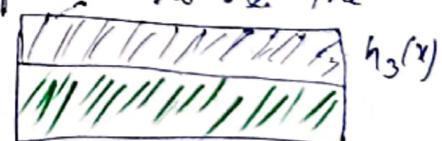
thik kar diye but iske ab khudke mistakes hain \rightarrow (-ve wale in the side)

Step 3:- 3rd decision stump (the wale in the regions) are upsampled & are focused on more here & again split is performed to ke the highest entropy possible

suppose α_3 hai

isne ~~extra~~ durr wale or ki galti thik kardi but ab khudki galti kar diya.

<table border="1"> <tr> <td></td><td>+</td><td>+</td></tr> <tr> <td>+</td><td>-</td><td>+</td></tr> <tr> <td>+</td><td>-</td><td>-</td></tr> </table>		+	+	+	-	+	+	-	-
	+	+							
+	-	+							
+	-	-							

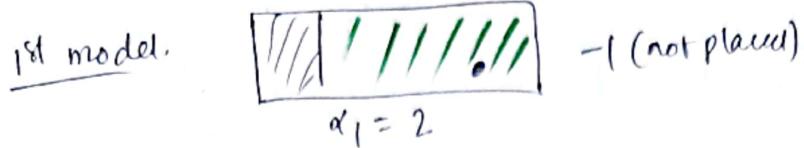


Now we have three Base models (on the right, diagram) & their weights (α_i); Now we add the three models

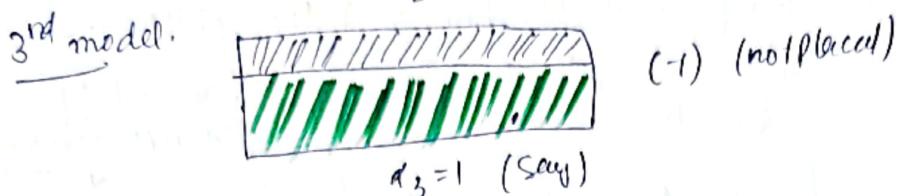
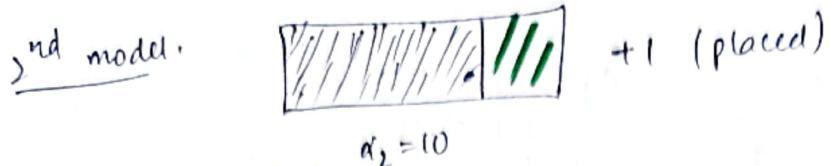
to AdaBoost model $A = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$ if pos A = +1

$$A = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)) \quad \text{if } -ve \ A = -1$$

Now let's do a prediction :- CGPA = 8.5, IQ = 81



- Not placed
- placed



$\therefore \text{final prediction} :=$

$$A = \text{sign} (\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3)$$

$$= \text{sign} (2(+1) + 10(+1) + 1(+1)) = \text{sign}(7) = +1$$

\therefore the student will be placed.

final decision boundary ke liye teeno model ke decision boundary ko ek ke rup par ek rakhe. we get:-



$n = 5 = \# \text{ rows}$. AdaBoost step by step

x_1	x_2	y	weight
3	7	1	
2	9	0	
1	4	1	
9	8	0	
3	7	0	

→ initially hum har apne row ko ek weight assign karte hai (2 equal value assign karte hai)

→ generally the weight assigned is $w_i = 1/5 = 0.2$
so there is now a weight col:-

weight
0.2
0.2
0.2
0.2
0.2

Stage 1 :-

1) we train a Decision stump (split with max entropy loss or max information gain).

Now iss m₁ ka performance checkge on the data set & we find y-pred.

x_1	x_2	y	$y\text{-pred}$	weight (initial)
3	7	1	0 (x)	0.2
2	9	0	0 (x)	0.2
1	4	1	0	0.2
9	8	0	0	0.2
3	7	0	0	0.2

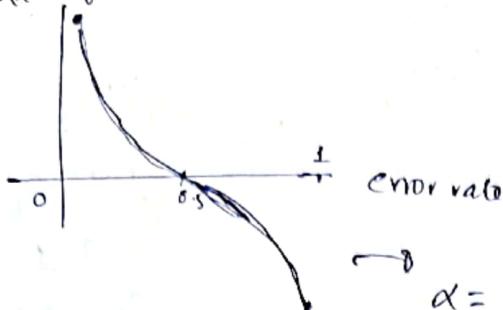
→ wrong prediction.

Now we find α_1 = weight of m_1 on final prediction.

Page 93.

It depends upon error rate (m_1 ne data pe kitni error rate jitna jyada, model utna bhejna galti ki) utna hi kam. $\therefore \alpha_1 \propto \frac{1}{\text{error rate}}$ generally error rate (error rate)

\therefore we want a fan with graph



(lowest error rate) accuracy high $\therefore \alpha = \text{high}$
(highest error rate) accuracy high in negative $\therefore \alpha = \text{high in -ve}$
 $\alpha = 0$
$$\alpha = \frac{1}{2} \ln \left(\frac{1 - \text{error}}{\text{error}} \right)$$

this is the fan.

To calculate error of a Decision Stump:-

from the Table find all the rows that are misclassified by m_1 (different Y-pred than actual Y) & then un rows ka initial weight ka sum:- So for m_1 ; error = 0.2 + 0.2

$$\therefore \alpha_1 = \frac{1}{2} \ln \left(\frac{1 - 0.9}{0.4} \right) = \frac{1}{2} \ln \left(\frac{0.6}{0.4} \right) = 0.4$$

$$\therefore \boxed{\alpha_1 = 0.20}$$

Ab m_1 ke galtiyon ko final prediction.

Stage 2 (m_2) ko upsampling karke (importance badha kya)

^(misclassified) To increase the importance of mistakes of m_1 :- Galtiya wali rows ka weights ko increase (Boost) kar denge & classified wale points ka weight isekar denge.
Updating weights:-

For misclassified points (galtiyo ka)

$$\text{new_wt} = \text{curr_wt} \times e^{\alpha_1}$$

For correctly classified points

$$\text{new_wt} = \text{curr_wt} \times e^{-\alpha_1}$$

for m_1 : for miscl. pts: $\text{wt} = 0.2 \times e^{0.2} = 0.24$

for clas. pts $\text{wt} = 0.2 \times e^{-0.2} = 0.16$

new-table-				X ₁	X ₂	Y	Y _{Pred}	weight	updated weights	normalized divide by sum=0.96
3	7	1	1					0.2	0.16	
2	9	0	1					0.2	0.24	0.25
1	4	1	0					0.2	0.24	0.25
9	8	0	0					0.2	0.16	0.166
3	9	0	0					0.2	0.16	0.166

n=5.

X ₁	X ₂	Y	new-wt	range	cumulative sum	upper value	lower value
3	7	1	0.166	0 - 0.166			
2	9	0	0.25	0.166 - 0.416	0.166 + 0.25		
1	4	1	0.25	0.416 - 0.666	0.416 + 0.25		
9	8	0	0.166	0.666 - 0.832	0.666 + 0.166		
3	9	0	0.166	0.832 - 1.0	0.832 + 0.166		

now we make a dataset with only these rows.

∴ dataset for stage 2.

Row1

Row3

Row3

Row3

Row4

→ this will go to the second Decision stump & everything will be repeated.

Jupyter Notebook

Since wt g of Row2 & Row3 are more therefore Row2 & Row3 have higher possibility than Row1 & Row4.

that range is higher since

AdaBoost Hyperparameters:

→ this gives the best result.

Decision Tree with max_depth=1)

Otherwise we can use

- base_estimator (default:- Decision Tree with max_depth=1)
- n_estimators = # of base models (logistic Regression, SVM) we want to use (default = 50)

→ in case of perfect fit (Decision stump with no mistakes), the learning procedure is stopped early.

- learning_rate:- default=1, float:- weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier. There is a trade-off b/w learning rate & n_estimators parameter.

- algorithm (SAMME, SAMME.R) (default:- SAMME.R)

Jupyter Notebook

Bagging v/s Boosting

Page 94.

Type of model used :-

Target is to achieve Low Bias + Low variance in both

in Bagging: the base models have Low Bias + High Variance \rightarrow like fully grown Decision Trees.

in Boosting: the base models are

High Bias + Low variance (like: shallow tree whose depth is low) (or KNN with local neighbour query)

Sequential v/s parallel Learning

in Bagging the dataset is split

subdataset & given to different base models. } Parallel Learning
in Boosting: Data \rightarrow model₁ then \rightarrow model₂ ... (same models Saath mei
Sequential Learning (Models "and" train hole hain)
ek ke baad ek train ho rhe hain).

Weightage of base learners

in Bagging the base models/learners have weightage = 1

in Boosting: han model ka alpha (Sath ka vote same hain) (by default)

jiska weight jyada weightage (α_i) hota hai.

jiska weight jyada uska contribution hota hai.

jyada in final output.

K-MEANS CLUSTERING

Clustering is an unsupervised ML technique where we put similar things in a group.

Internal working

But ye agar higher dim mei dekhna nahi payenge cluster kar denge & bol cluster dekhna hum dekhkar hi 2D mei Dikh rakte hain.

Steps: 1. Decide n clusters & then we use

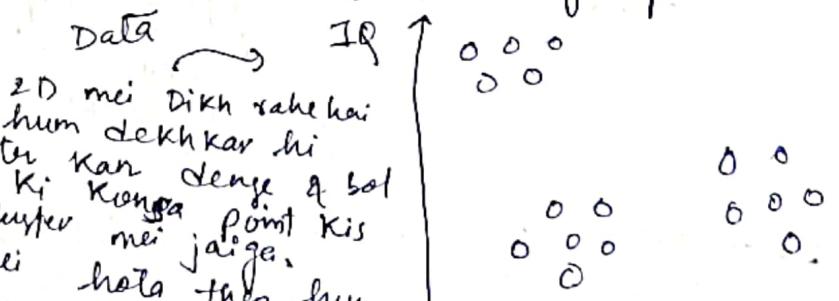
2. Initialize centroids

3. Assign Cluster

4. Move Centroids

5. Finish.

Data \rightarrow

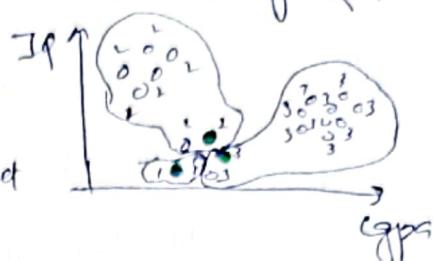


K-means clustering



1/ Decide n clusters \Rightarrow we'll have to tell the Algo how many clusters we want (wicnd) \rightarrow It is denoted by K .

Say $K=3$.



2/ Initialize centroids: Algorithm from dataset

Randomly kisi bhi 3 points ko
Centroid man lega

3/ Assign cluster: Ab humne har point ko ek cluster assign karna hai on the basis of the 3 centroids chosen.

Ek ~~one~~ point ka ~~one~~ cluster centroid & distance nikalo & jis centroid & distance sabse kam hai usko us centroid ke cluster mei rakho.

Aisa hum har point in the dataset ke liye kareng.

4/ Move Centroid:

Ab in teeno clusters ka centroid calculate kareng.

Say cluster 3 ka centroid: (mean of c_{ps} q points in cluster 3, point in cluster 3)

Sty. we will find centroids of cluster 1 & cluster 3.

This need not be a point on the dataset
It will be at the centre of the cluster 3.

* To check we should finish the Algo are not:-

compare the ^{new} centroids to the previous step centroids, if they are the same then no improvement in centroids \rightarrow finish the Algo & if they are different then Algo is improving by moving the centroid \rightarrow continue. Step 3 \rightarrow Step 4

Assign cluster again w.r.t these new centroids.

Do this till finish cond^h is met
Move centroid again using the above formula

→ centroids same rehna ka matlab hai ki points in that cluster are not changing; no new points are added or no points are getting removed i.e. cluster is now fixed.

The only metric we used in this Algo is Euclidean distance which works in higher dimension also $\&$ \therefore K-means clustering is such a powerful Algo.

* How do we guess the no. of clusters (k) at the beginning of the K-means clustering Algo?
(Elbow method)

Page 95-
 $\sum_{\text{within clusters}} \text{squared distance}$
 WCSS (inertia)

In this we plot a graph called elbow graph/curve.
To find $\text{WCSS}_{\text{for } k \text{ cluster}}$ do ari wtsa $\text{centroid ka Euclidean distance vs cluster ke hon points } z_i \text{ Nikalo}$
then Sab ka square karke, sum kardo
if there are multiple clusters then WCSS of big cluster
= sum of individual WCSS .

* to form the ~~elbow~~ elbow curve:-

pehle maant ki pure dataset ek hi cluster hai:-
wtsa = WCSS_1 nikalo & plot kardo $(1, \text{WCSS}_1)$

Now consider two clusters in the Data set,

Total $\text{WCSS} = \text{sum of individual } \text{WCSS}$
plot $(2, \text{Total WCSS}_2)$

Next:- consider 3 clusters in the Dataset,

Total $\text{WCSS} = \text{sum of individual } \text{WCSS}$

plot $(3, \text{Total WCSS}_3)$

and so on do it till 10, 15, or 20 (at most in general)

It is obvious that:-

(think :- the new centroids are getting closer to the points as the # clusters are increased & hence WCSS decreases)

$\text{WCSS}_1 > \text{WCSS}_2 > \text{WCSS}_3 > \dots > \text{WCSS}_n$

if all the points are centroids i.e.

clusters = # points in dataset

then individual $\text{WCSS} = 0 \therefore \text{Total WCSS} = 0$

Now in the graph we need to find the elbow point:-

The point in the graph where the decrease seems to stabilize.

& that elbow point is the # clusters.

(steepness/slope becomes stable at that point).

Jupyter Notebook

on salary then m1.



GRADIENT BOOSTING

For Regression

Consider the toy Dataset:

IQ	GPA	Salary
90	8	3
100	7	4
110	6	8
120	9	6
80	5	3

Mean
4.8

Step 1.

IQ	GPA	Salary	Pred 1 (by m_1)
90	8	3	4.8
100	7	4	4.8
110	6	8	4.8
120	9	6	4.8
80	5	3	4.8

→ Sequential stage wise addition

matlab data ek ke baad ek base

models pe train hoga.

& haan ~~Model~~ base model ko apne
pichle wale model ka galtiyan ko
pata hoga & wo un galtiyon ka
nahi karega.

Let's say we want to use three base
models :-



Now for Regression problem:- m_1 is fixed
in Gradient Boosting & it returns the
Mean of the output col is 4.8.
i.e. give any input col it returns that
the Salary will be 4.8 / predict

Step 2. Now we need a loss fn that tells
ki ~~Model~~ model ne galti kitni ki & wohi
galti pichle hum next model ~~ko~~ batainge.

Here loss fn = pseudo residual = actual - predicted
 res_1 & Now is galti ko
 -1.8 -0.8 3.2 1.2 -1.8 humne somehow model 2
new col of residual of m_1 bhejna hai

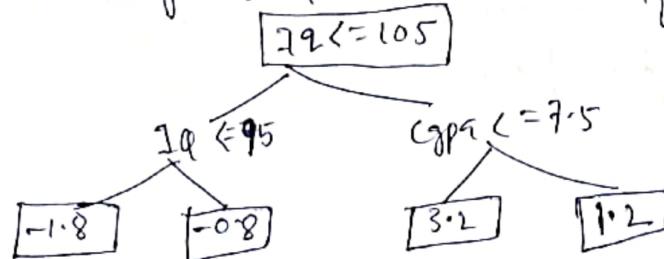
take bhejna hai

Our model 2 generally is Decision Tree (Gives best Result).

Note:- To model 2 we give: $\begin{matrix} \text{IQ} & \text{GPA} & \text{res}_1 \\ \vdots & \vdots & \vdots \end{matrix}$ output i.e. by model 2
we want to predict the res_1 (mistakes of m_1) & jitna accurately

m_2 res₁ pred. karega hum utna accurately hum actual
output (Salary) predict kar payenge.
Say we trained m_2 (Decision Tree) & we get:-

Using Decision Tree
Regressor



Now we predict on the data using m_2 → page 96.

IQ CGPA	Res1	Pred2	
90 8	-1.8	-1.8	
100 7	-0.8	-0.8	
110 6	3.2	3.2	
120 9	3.2	3.2	
80 5	1.2	1.2	
	-1.8	-1.8	

Overfitting (since pure training data perfectly same predict kar diya).

Now we have 3 base models but say agar 2 hi hota
then actual prediction kaise hota :-

of m_2 on salary \leftarrow Actual pred = m_1 pred + m_2 pred (which is obvious also)
Ex: for input query: $\frac{90}{8}$ To get pred. 'kaifi accha hone chahiye

$$m_1 \text{ pred} = 4.8 \text{ (mean)} \quad \therefore \text{Actual pred} = 4.8 - 1.8$$

$$m_2 \text{ pred} = -1.8 \text{ (mistake of } m_1) \quad = 3$$

→ Yaha overfitting ho toh rahi hai (since actual prediction ho raha hai hence naye data pe acche prediction mati hoga)

So to avoid this we use learning rate:-

of m_2 on salary \leftarrow Actual pred = m_1 pred + $\alpha r \times m_2$ pred

Now for input: $\frac{90}{8}$ generally $\alpha r = 0.1$ To get se hamara step chota bortha hai towards the right direction (actual Ans)

$$\begin{aligned} \text{Actual pred} &= 4.8 + 0.1(-1.8) \\ &= 4.7 \end{aligned}$$

Now it is not accurate enough

So now we use more models to improve it:-

IQ (CGPA)	Salary	by m_1 only			by m_2 only	
		Pred1	Res1	Pred2	Res2	
90 8	3	4.8	-1.8	-1.8		
100 7	4	4.8	-0.8	-0.8		
110 6	8	4.8	3.2	3.2		
120 9	6	4.8	1.2	1.2		
80 5	3	4.8	-1.8	-1.8		

$$\begin{aligned} \text{Now we find residual 2} \\ = \text{res2} \\ = \text{salary} - (\text{pred1} + \alpha r \times \text{pred2}) \end{aligned}$$

res2
-1.62
-0.72
2.88
1.08
-1.62

hence the mistakes are being shown m_2 did lesser mistakes on salary than m_1 .

Till here: (Actual pred of m_2 on salary) \leftarrow Salary = Pred1 + αr + Pred2

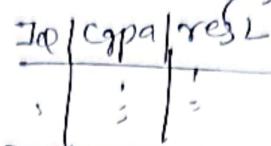


Ideally we want the residual = 0 of all the base models per the residual loss.

So we want the residual to be 0 or almost zero.

m_3 will again be decision tree

Now the data will be



i.e. m_3 will predict the mistakes of m_2 .

& more accurately if predicts the mistakes of m_2 the better it will predict the actual output (Salary).

model 3 (Decision Tree)

$19 \leq 105$

$19 \leq 95$

$cgpa \leq 7.5$

-1.62

-0.72

2.88
1.08

Pred₃ Pred of m_3 on res_2 using

-1.62 Now Actual Pred of m_3 on Salary

-0.72

$$= \text{Pred}_1 + \alpha_1 \times \text{Pred}_2 + \alpha_2 \times \text{Pred}_3$$

2.88
 $\alpha_1 = 1.08$

-1.62

Say input col = $19 | cgpa$

$$\begin{aligned} \text{then: Actual Pred} &= 4.8 \times 0.1 + (-1.8) + 0.1 \times (-1.62) \\ &= 4.8 - 0.18 - 0.162 \\ &= 4.458 \end{aligned}$$

$\alpha_2 = 0.1$
same for
all base
models.

But this is how Gradient Boosting works.
So we need more base models

Jupyter Note Book.

AdaBoost v/s Gradient Boost

1) Max_Leaf nodes :- in AdaBoost we use Decision Stumps i.e. decision tree with max depth = 1. \therefore max_leaf_nodes = 2 (fixed)

in Gradient Boost max_leaf nodes ranges from 8 to 32

2) Learning Rate :- in AdaBoost :- Every model (base) gets a weight α_i for small dataset for big dataset. they can be different & tells us ki konge model jyada important ho. In Gradient Boost we use learning rate & it is same for each base model.

math for Gradient Boosting classification

Page 97

Concept of Additive modelling

What ML models do? $\frac{x}{\text{input}} \frac{y}{\text{output}}$ then M₂ models a func/nets
 $y = f(x)$.

If data is: $x_1 | x_2 | x_3 | y$ then $y = f(x_1, x_2, x_3)$
if the ~~relationships~~ are simpler then the net is found easily (Linear, polynomial) but if the relationship is not additive then we find the bigger func y or $F(x)$ up.

So in additive modelling we find the sum of smaller func: $y = F(x) = f_0(x) + f_1(x) + \dots + f_n(x)$.
in Ensemble Learning
these func are given by Base models.

Additive modelling is a statistical technique for modelling complex relationships between variables by breaking them down into a sum of simpler relationships. The idea is to add up the simpler of the predictors to model the response, rather than attempting to model the response as a complicated func of the predictors.

Ex: Let's say temp, rainfall & crop yield. Instead of trying to model the relationship b/w these variables as a single complex eq, an additive model would break it down into three separate, simple relationships, the relationship b/w temp & crop yield, rainfall & crop yield & temp & rainfall. These simple relationships can be added together to give a final model of the relationship b/w temp, rainfall & crop yield.

Let's understand the math with example in

jupyter Notebook:

We'll do the math/Algo here & apply it in the Jupyter Notebook.

Input :- training set: $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss func

$$L(y, F(x)) = L(y, g)$$

x_i is input data (it can have multiple cov)
 y_i is output data
 $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\}$
and number of iteration = M

in our fn in Jupyter Notebook.

x_i : R&D spend, Adminis, Marketing spend

y_i : Profit & we have only x_1, x_2, x_3
& y_1, y_2, y_3

↓ L model ka predicted output

the loss func that we will use here is Least square just $\frac{1}{2} \sum$ multiply for mathematical convenience

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



Step 1: Initialize $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$.

We have to find $\gamma = f(x)$

& since this is a boosting algo $\therefore L(f(x)) = f_0(x) + f_1(x) + \dots + f_n(x)$

Now to find $f_0(x)$,

$$\text{use } L_2 \sum_{i=1}^N (y_i - \gamma_i)^2 = L(y_i, \gamma_i)$$

& diff w.r.t γ & make it $\equiv 0$ to find the value of γ for which it is min.

we find that γ is the Mean of the output col.

hence we use $f_0(x)$ as mean of the output col.

is $f_0(x)$ is not a Decision tree.

Note:- if we use a diff Loss fn then γ will not necessarily be a Mean.



KNN. (K Nearest Neighbours)

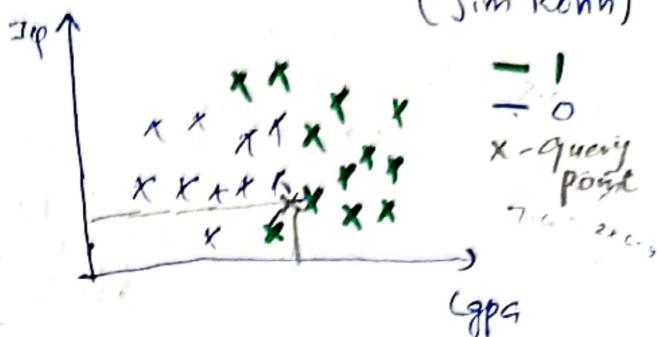
* You are the average of
the five people you spend
the most time with.

(Jim Rohn)

Consider 100 students Dataset

GPA	IQ	placement	plot the
8	80	1	
7	70	0	
6	60		
5	50		
4	40		
3	30		
2	20		
1	10		
0	0		

input output



The logic of KNN is Ki aap apne neighbours ki tarah hote ho.

we first check/choose the value of K (say = 3)

Now say we have a query point (see diagram, pencil)

Now we find the distance of this query point from all the points in the training dataset. (generally Euclidean distances). Now we sort all the distances in ascending order & we find the (K=3) closest points & then we apply the rule Majority Count (Democracy) of the

Now we see Green(1) are in majority of $K=3$ closer points hence the query point will also be green i.e. placement 1.

Note: KNN works for any dimensional Dataset.

Jupyter Notebook.

How to select K? It actually depends upon the data.

Heuristic approach

$K = \sqrt{n}$ ($n = \# \text{ of observations} = \# \text{ of rows in the training data set}$)

Experimentation (Better)

To use cross Validation for different KNN models with different K value & see the accuracy of each & compare them & find the best K value.

Generally, we avoid even value for K since majority count mei dikkat hoga.

Decision Boundary / Surface used in

- for 1D, 2D or 3D Data Classification
- SVM, Log. reg
- Dec. Tree.
- Neural Networks.

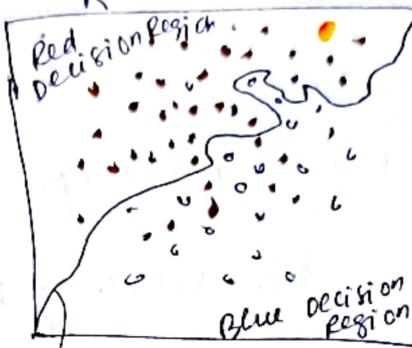
Consider 2D Data

Imagine the Data.

cpt2 | 2D | Placement

Binary classification problem (output has two classes)

K=5



query point

then it is in the
red Decision Region
hence veka placement
Red walo ke tarah
hi hoga.

decision Boundary

cpt4

Steps:

- (1) plot the training Data.

(2) find x & y range.

(3) for this generate a
numpy meshgrid

that is generate many points in this Region

(4) train the kNN model on the training set

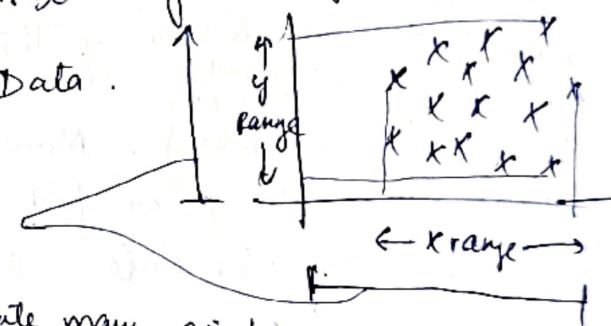
(5) send the meshgrid points generated to the kNN model
if kNN predict 0 → Blue & encode Kano us meshgrid
if kNN predict 1 → Red & Point Ko

& Aiss ha meshgrid points ke liye Kano.

& then show this points as pixels on an image

& we get the Blue & Red Decision Regions

* we have a library called mlxtend → need to plot
Decision Surfaces.

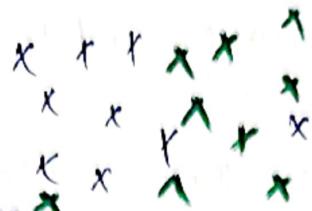


Oversfitting & Underfitting

Page 100.

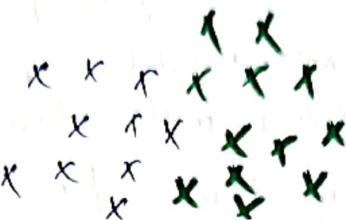
Say $K=1$.
(Low value)

Query point
It will be declared
Green although it should have been Blue.



$K=n = \text{Hg rows}$.

(High value)



This is a case for underfitting.

The model is unable to understand the pattern of the dataset.

In this case

- the decision Region will be divided into many parts

→ This is Overfitting

minor changes
ko thi capture
kar tha hai model.
(high variance)
alag alag dataset pe
alag alag prediction
karna.)

→ here the output will be that class which is dominantly i.e. jo class mei sabhi jyada elements hai wohi dominate kar dega & Decision Region thi bas ek hi class ka banega.

Hence output of any query point is the majority class

Aise ML algo jisme SaavKam prediction
mei hi hote hai (distance calculation).

→ KNN is a lazy learning technique

Training me toh b as data plot hota
iske distance nikalne
mei hi hote hai (distance calculation).
Sort → use majority count

① we would not use KNN on large datasets like $n=100$ (say)
 $\# \text{ of rows/observations} = \# \text{ of features/cols} = 100$ (say)
∴ ~~query~~ point ka han point ki distance nikalne
hai & that will take a lot of time. Hence
the predictions will be very slow.

→ (5 lakhs distances calculate → sort → use majority count)

② High dimensional data ($\# \text{ of cols} = \text{say } 500$) then
Curse of Dimensionality aajata hai i.e. in such
high dimensions the distance concept is not
reliable & hence KNN ke predictions thi reliable nahi
hote.

③ Cannot handle outliers nicely
Ke nearest neighbours ko dekhkar to predict kar dega.
for lower values of K.
which is obvious - KNN outliers

4) Non-homogeneous feature scales

↳ Taki standard scaling karte hai before using KNN.

- 5) Imbalanced datasets :- Those datasets in which the categories/classes in the output col are highly different say Yes or 1 wale points 98% hai } considering & No or 0 " " 2% hai } binary classification
In this case the KNN model will be biased towards the Yes or 1 (98%) wale class.

- 6) When we are using KNN for inference & not for predictions then KNN fails.

i.e. if we have a query point the KNN can predict ki wo koni class mei fall karega but it cannot tell ki ye prediction Karne mei CGPA ka jyada contribution tha ya IL KA. (cannot tell ki koni ka feature jyada imp hai prediction ke)

i.e. KNN acts like a black box model.
ML models are generally $y = f(x)$, but KNN can tell the contributions of x_1, x_2 components of x .

KNN Hyperparameters

- n-neighbours \rightarrow k value (default=5)
- weights :- (default='uniform')

Other values: all points in each neighbourhood are weighted equally.

\rightarrow distance : weight points by the inverse of their distance.

In this case closer neighbors of a query point will have a greater influence than neighbours which are further away.

\rightarrow callable : a user-defined function which accepts an array of distances & returns an array which accept an array of same shape containing the distances.

- algorithm \rightarrow used to compute the nearest neighbours.

• Ball Tree

• KD-tree

• brute (default)

• auto \rightarrow decides the most appropriate algo based on values passed to fit method.



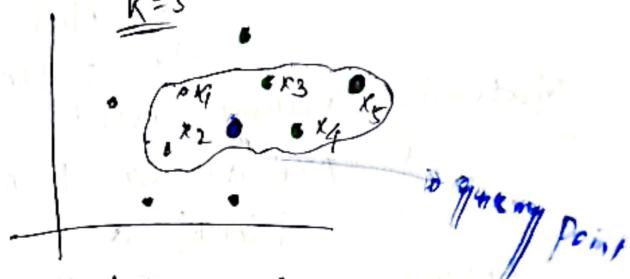
- leafsize \rightarrow default = 30
↳ used only when we are implementing the Ball Tree or Kd Tree algorithm
- p (integer, default value = 2)
↳ the distance metric to use for the tree.
When $p=1 \rightarrow$ manhattan-distance (ℓ_1)
When $p=2 \rightarrow$ euclidean distance (ℓ_2)
- metric :- (default : minkowski)
↳ with $p=2$, it's equivalent to standard Euclidean metric
- n-jobs \rightarrow multi processing; default = 1; if we want to use all the cores of the CPU then use value -1.

Weighted KNN

Lagft is a variant of KNN where we take a simple yet elegant assumption that the impact of nearer neighbours on the query/test points should be more than the farther away points.

5 nearest point	Label	Distance from query point
x_1	B	0.2
x_2	B	0.5
x_3	G	0.7
x_4	G	1.2
x_5	G	1.5

$$\begin{aligned} \text{weight } & \\ 0.2 &= 5 \\ 0.5 &= 2 \\ 0.7 & \\ 1.2 &= 0.8 \\ 1.5 &= 0.6 \end{aligned}$$



Now we calculate weight: Based on weighing fn
Simpler weighting fn
is the inverse fn
 $w_i = \frac{1}{d_i}$
rule \rightarrow Distance goes, weight goes.

$$\begin{aligned} \text{Now weighted sum of blue points} &= 5 + 2 = 7 \\ \text{weighted sum of green points} &= 2 \cdot 8 \end{aligned}$$

Hence the query point will be labelled as Blue.

Note Simple KNN we take for Green labelled kota ye query point (see the above Diag.)

Naive Bayes

conditional prob.

$$P(A|B) = \frac{P(ANB)}{P(B)} ; \text{ provided } P(B) \neq 0$$

Independent events

A, B are called independent if $P(ANB) = P(A) \times P(B)$

ऐसे events जो साथ में तो होकर बाले होंगे तो वे independent होते हैं कि उन्हें कोना काना नहीं पड़ता.

Ex:- Rolling a die twice, पहली बार 3 आया तो दूसरी बार 5 आने की आया prob नहीं ऐसा है $\frac{1}{6}$ क्योंकि पहली बार 3 आया तो दूसरी बार 5 आने की prob नहीं ऐसा है.

In case of indep Events

$$\begin{aligned} P(ANB) &= P(A) \times P(B) \\ \text{& we know } P(A|B) &= \frac{P(ANB)}{P(B)} = \frac{P(A) \times P(B)}{P(B)} \\ \therefore P(A|B) &= P(A) \\ \text{Sly, } P(B|A) &= P(B) \end{aligned}$$

Mutually Exclusive Events:-

A, B are called mutually exclusive events if
 $P(ANB) = 0$

Ex:- एक coin उछाला दिया जाता है। Head & Tail दोनों साथ में आये नहीं सम्भव हैं।

Also, $P(A|B) = \frac{P(ANB)}{P(B)} = 0$

Sly, $P(B|A) = 0$

Bayes' Theorem

A, B are two events then

$$P(A|B) = \frac{\underset{\substack{\rightarrow \text{Likelihood} \\ \downarrow \text{Posterior}}}{P(B|A)} \underset{\substack{\rightarrow \text{Prior} \\ \downarrow \text{Evidence}}}{P(A)}}{\underset{\substack{\rightarrow \text{Evidence} \\ \downarrow \text{Posterior}}}{P(B)}}, \text{ given } P(B) \neq 0$$



Intuition
Consider the Dataset of CSK.

Page 102.

Toss	Venue	Outlook	Result
won	Mumbai	Overcast	won
lost	Chennai	Sunny	won
won	Kolkata	Sunny	won
won	Chennai	Sunny	won
lost	Mumbai	Sunny	lost
won	Chennai	Overcast	lost
won	Kolkata	Overcast	lost
won	Mumbai	Sunny	won

← input ← output →

→ Given input query we have to predict Chennai Super Kings will lose or win the match.

→ Binary classification problem

Suppose input query is :-

{lost, Mumbai, sunny} → Then Naive Bayes will find out $P(\text{win})$ given the above (query) cond's & $P(\text{loss})$ given the above (query) cond's.

Hence we need to find $P(\text{W}/\text{A})$ & $P(\text{L}/\text{A})$

& jiski bhi prob jyada hei wohi answer hai Naive Bayes ko.

Info know:- $P(A/B) = \frac{P(B/A) \times P(A)}{P(B)}$

$$\text{so } P(\text{W}/\text{A}) = \frac{P(\text{A}/\text{W}) \times P(\text{W})}{P(\text{A})} = \frac{\cancel{P(\text{A}/\text{W})}}{\cancel{P(\text{A})}} \times \frac{5/8}{1/8}$$

$$A = \boxed{\text{Loss} \cap \text{Mumbai} \cap \text{Sunny}}$$

<u>A/w.</u>	
won	Mum over
lost	Chennai sunny
won	Kol sunny
won	Chennai sunny
won	Mum sunny

$\therefore P(A/w) = 0$

Now:- Naive Bayes is using a simple assumption on the Bayes theorem & the condition is:-

$$P(\text{A}/\text{w}) = P(\text{Loss, Mumbai, sunny}/\text{w})$$

$$= P(\text{Loss}/\text{w}) P(\text{Mumbai}/\text{w}) P(\text{Sunny}/\text{w}) \quad (\text{why see later})$$

$$= \frac{1}{5} \times \frac{2}{5} \times \frac{4}{5}$$

$$\therefore P(\text{W}/\text{A}) = \frac{\left(\frac{1}{5}\right) \times \frac{2}{5} \times \frac{4}{5}}{\frac{1}{8}} = 0.2$$

$$\text{Sly. } P(\text{L}/\text{A}) = \frac{P(\text{A}/\text{L}) \times P(\text{L})}{P(\text{A})} = \frac{P(\text{Loss, Mumbai, sunny}/\text{L}) \times P(\text{L})}{P(\text{A})}$$

$$\text{using Naive Bayes} \quad = \frac{P(\text{Loss}/\text{L}) \times P(\text{Mumbai}/\text{L}) \times P(\text{Sunny}/\text{L})}{P(\text{A})} \times \frac{3}{8}$$

$$= \frac{\frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{3}{8}}{\frac{1}{8}} = 0.3$$

Hence $0.3 > 0.2$
 \therefore CSK will lose the match



Mathematics behind Naive Bayes

$X = \{x_1, x_2, \dots, x_n\}$ \rightarrow input cols / input query.

$C_K = \{C_1, C_2, \dots, C_K\}$ \rightarrow classes / categories in output col.

C_K multidim classification (K classes).

We need to find :-

$$P(C_i | X) \rightarrow i = 1, 2, \dots, K \quad \text{and jyada hoja}$$

wohi hum answer mein
length.

$$P(C_i | X) = \frac{P(X | C_i) \times P(C_i)}{P(X)} \quad (\text{using Bayes theorem})$$

Note that $\forall i = 1, \dots, K$; the deno for all $P(C_i | X)$ is same $P(X)$
hence it will not help in comparison so we only find the
numerator.

so we only need to find $P(X | C_i) \times P(C_i) \rightarrow i = 1, 2, \dots, K$
we know $P(A \cap B) = \frac{P(A \cap B)}{P(B)} \Rightarrow P(A \cap B) = P(A | B) \times P(B)$

$$\begin{aligned} P(X | C_i) \times P(C_i) &= P(X \cap C_i) = P(X, C_i) \\ &= P(x_1, x_2, \dots, x_n, C_i) = P(x_1, \overbrace{x_2, \dots, x_n, C_i}^{\substack{A \cap B \\ = P(A \cap B) \\ = P(A | B) \times P(B)}}) \\ &= P(x_1 | x_2, x_3, \dots, x_n, C_i) \times P(x_2, \dots, x_n, C_i) \quad \forall i = 1, 2, \dots, K \\ &\quad \vdots \\ &= a_1 \times P(x_2 | x_3, \dots, x_n, C_i) \times P(x_3, x_4, \dots, x_n, C_i) \\ &= a_1 \times a_2 \times P(x_3, x_4, \dots, x_n, C_i) \quad \rightarrow \text{this breaking is called Chain Rule for cond. prob.} \\ &= a_1 a_2 \dots a_{n-1} \times P(x_n | C_i) \times P(C_i) \\ &= P(x_1 | x_2, x_3, \dots, x_n, C_i) \times P(x_2 | x_3, \dots, x_n, C_i) \times \dots \times \\ &\quad P(x_{n-1}, x_n | C_i) \times P(x_n | C_i) \times P(C_i) \end{aligned}$$

Now we take Naive Assumption

i.e. x_j is independent of $x_{j+1}, x_{j+2}, \dots, x_n$, it only depends upon C_i



And the Assumption is called Conditional Independence

Page 103.

Hence :- $P(X_1 | X_2, \dots, X_n, c_i) = P(X_1 | c_i)$

say. $P(X_2 | X_3, \dots, X_n, c_i) = P(X_2 | c_i)$

$P(X_{n-1} | X_n, c_i) = P(X_{n-1} | c_i)$

∴ Num of $P(c_i | x) = P(x_1 | c_i) P(x_2 | c_i) \dots P(x_n | c_i) \times P(c_i)$

for $i=1, \dots, K.$ $= P(c_i) \times \prod_{j=1}^n P(x_j | c_i)$

∴ Actual formula is

for $i=1, 2, \dots, K$

$$P(c_i | x) \propto P(c_i) \prod_{j=1}^n P(x_j | c_i)$$

Now: jis i ke value ke liye $P(c_i | x)$ will be largest x uski c_i class mei jaiga.

So $\hat{y} = \underset{i=1, 2, \dots, K}{\text{arg max}} P(c_i) \prod_{j=1}^n P(x_j | c_i) \rightarrow$ Maximum a posteriori rule.

Jupyter Notebook

Numerical Data in Naive Bayes

Height | weight | Gender

Height	Weight	Gender
172	150	M
180	170	M
165	140	M
190	200	M
139	100	F
145	120	F
160	140	F
172	150	F

input query :- H=185, W=170, G=?

To find $P(M | H=185, W=170) = P(M) \times P(H=185|M) \times P(W=170|M)$

& $P(F | H=185, W=170) = P(F) \times P(H=185|F) \times P(W=170|F)$

So we take Height is Gaussian Assumption that Random variable. using previous method $P(H=185|M) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(H-\mu)^2}{2\sigma^2}}$. $P(H=185|F) = 0$.

$$P(H=185|F) = 0$$



So to find: $P(H=185|M)$ \rightarrow
 & $P(W=170|M)$
Step 1: find μ, σ of this
 in case of Normal dist we know $\text{pdf}(\text{prob}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
 & for weight find μ, σ for 150, 170, 190, 200.
 & so we will do this for Female Category:
 To find $P(H=185|F) \& P(W=170|F)$

\downarrow This method is called Gaussian, Naive Bayes.
 $x = 185$

So what if the dist. is not Normal:-

then we first plot (dist plot) and see ki konge

dist follow kar zya hai & accordingly we use

that dist like:- Gaussian Naive Bayes or

Binomial Naive Bayes

or Multinomial Naive Bayes,

or Poisson Naive Bayes & using

their pdf formula for the probability

Sentiment Analysis

Review	Sentiment
Great movie	positive
again	
The movie was also boring... never	negative
Good movie	positive

We'll have to predict the sentiment of a new incoming Review.

Steps

- 1) Text Preprocessing
- 2) Algo will understand Numerical Data so we will vectorize the Review (using technique Bag of words)
- 3) Train the Algo with this data
- 4) Deployment.

removing html tags,
 removing words like
 and, if, or jinka
 analysis mei koi
 fayda nahi hai

Review

Bag of words

BOW

data





Bag of words

Review column mei jitne bhi unique words hai un sabka ek ek col bana dete hai.

Let's say we have only 4 unique words :-

Review1	"Great"	"awesome"	"poor"	"pathetic"	Label (sentiment)
	3	1	0	0	1 (pos)
Review2	0	1	2	9	0 (-ve)

Now we pick up Review1 and see in charo mei \Rightarrow kongya words aarha hai & kongya nahi in Review1, agar

Review1 mei Great 3 baar aya, awesome 1 baar, poor \rightarrow 0 & pathetic 0 baar then we fill the table above for row Review1 as above.

& so on we do that for each Review. (50000 reviews say)

Bag of words can be done using the class CountVectorizer in the Sklearn library.

Suppose the final dataset is:-

awesome f1	great f2	boring f3	label
3	1	0	pos(1)
0	0	2	-ve(0)
3	3	1	pos(1)

So what Naive Bayes do:-

$$P(\text{label} = \text{pos} | f_1=3, f_2=1, f_3=0)$$

$$\& P(\text{label} = \text{-ve} | f_1=0, f_2=1, f_3=1)$$

we know the formula for calculation of these.

Jupyter Notebook. (Do it on Kaggle, Jupyter Notebook mei slow hoga).

input Query:

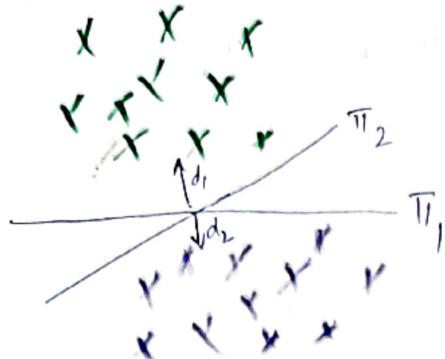
"The movie was great, but slightly boring" $\rightarrow [0, 1, 1]$

How to predict the sentiment.

This is nothing frequency count of the words "awesome", "great", "boring" in the input Review string.

& jo bada hoga uski ans hoga.

Support Vector Machines



→ It improves the idea of ~~Support~~ Logistic Regression

Both the planes / lines π_1 & π_2 are able to classify the dataset with 100% accuracy. But as a classifier which one is better? SVM chooses the better hyperplane for us.

Ans is: π_1

SVM chooses that line to Data to classify both classes such that d_1 & d_2 to maximise d_1 (for binary classification problem)

∴ The hyperplane is called Margin Maximizing hyperplane.

How to select the Margin Maximizing hyperplane:

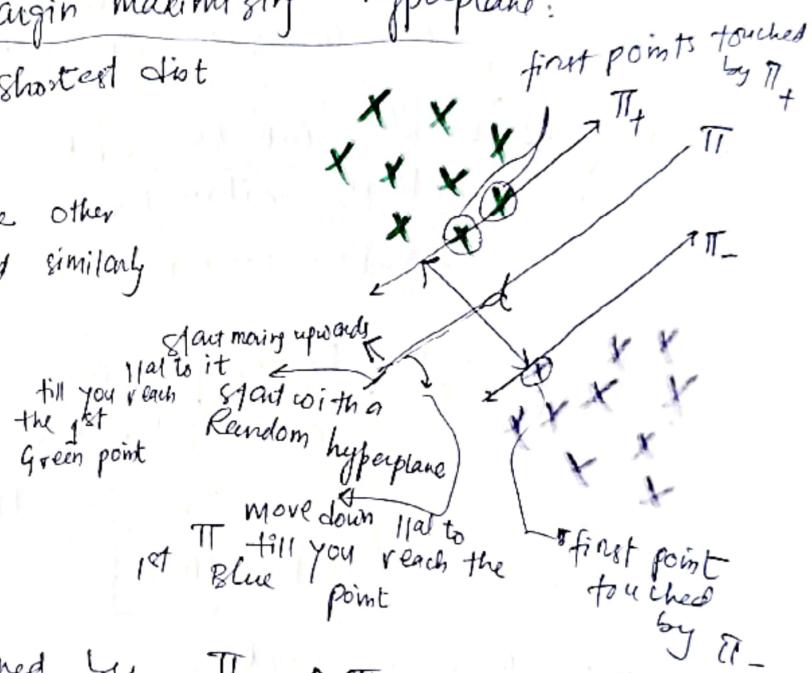
Now we find Margin = shortest dist b/w π_+ & π_- = d_1

Then again we start with some other random hyperplane & proceed similarly

I find d_2 , then $d_3 \dots d_n$
Then finally choose that hyperplane for which d_i is maximum.

Concept of Support vectors

The first points touched by support vectors. (see diagram)



- SVM are very robust to outliers (They handle them quite well)
- It also works when the data is non-linear. (using kernels)
- SVM can be implemented to both classification & regression problems.

Equation of a hyperplane

in n-dimension.

$$\Pi: w_1 x_1 + w_2 x_2 + \dots + w_n x_n + w_0 = 0 \rightsquigarrow w \cdot x + w_0 = 0$$

for 2-D. $w_1 x_1 + w_2 x_2 + w_0 = 0 \rightsquigarrow$ line

for 3-D. $w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 = 0 \rightsquigarrow$ plane.

$$w = [w_1, \dots, w_n]$$

$$x = [x_1, \dots, x_n]$$

$$w^T x + w_0 = 0$$

∴ Assuming ki hamana hyperplane origin
et pass kane, the eqn is

$$w^T x = 0$$

$$w^T = [w_1, \dots, w_n]$$

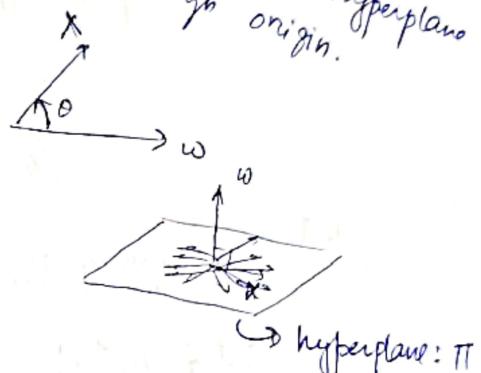
$$x = [x_1, \dots, x_n]$$

Passing through origin.
hyperplane

we know. $w^T x = w \cdot x = \|w\| \|x\| \cos \theta = 0$

when $\theta = 90^\circ$.

w is a vector which is \perp^r to the hyperplane: Π .



Mathematics of SVM

we need to find the eqn of a hyperplane for which the margin d is maximum.

Decision Rule for Prediction

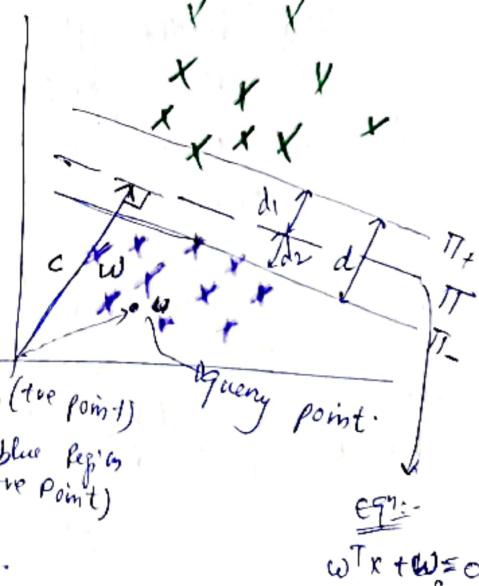
The query point. \vec{u} is a vector
And \vec{w} we know from above
is \perp^r to hyperplane. Now we
take projection of \vec{u} on \vec{w}

$\vec{u} \cdot \vec{w}$ if we check

$\vec{u} \cdot \vec{w} \geq c$ \rightarrow yes query point in green Reg'g (re point)

$\vec{u} \cdot \vec{w} - c > 0$ \rightarrow no query point in blue Reg'g (-ve point)

$$\vec{w} \cdot \vec{u} + w_0 \geq 0 \rightarrow$$
 decision Rule.



In General: Eqn of Π : $w_1 x_1 + \dots + w_n x_n + w_0 = 0$, $w^T x + w_0 = 0$
 $\vec{u} = (u_1, \dots, u_n)$ is query point then we check
 $w_1 u_1 + \dots + w_n u_n + w_0 \geq 0$ or ≤ 0 (simple).
i.e. $w^T u + w_0 \geq 0$ or ≤ 0 .

so for any \vec{x}_i (query point)

$$\hat{y} = \begin{cases} +1 & \text{if } \vec{w} \cdot \vec{x}_i + w_0 \geq 0 \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + w_0 \leq 0 \end{cases}$$

Now we need to find the eqn of hyperplane $\vec{w}^T \vec{x} + w_0 = 0$
Maximizing $d = \text{shortest dist b/w } \Pi_+ \text{ & } \Pi_-$

To find w & w_0 .

Note that: Π should be exactly in b/w Π_+ & Π_-

$\Rightarrow d_1 = d_2$ ($d = d_1 + d_2$) See previous diagram

Let the eqn of Π_+ : $\vec{w}^T \vec{x} + w_0 = 1$ (1, +1 to y_1 5, -5 to y_4 such that Π_+ has same magnitude as Π_-)
& eqn of Π_- : $\vec{w}^T \vec{x} + w_0 = -1$ (Π_+ mei ye ~~will~~ same hi raha ga same to Π hai dono same raha ga \therefore parallel)

Note:

LHS of Π_+ & RHS of Π_- ko

>1 factor \vec{w}
multiply karne ke
d shrink hoga

<1 factor \vec{w} multiply with 2D (line)
karne ke d jse hoga

see in Desmos

graph

Note: There are some constraints on the Data set:

Π_+ is the 1st shift of Π such that it touches the first green point so no green pt can be below it
& Π_- " " " " " " " " " " " "

" blue " " " " " " " " " " " above it.

we have to maximise d keeping this constraint in mind.

So the constraints can be written mathematically

for green points (\vec{x}_i) $\rightarrow \vec{w} \cdot \vec{x}_i + w_0 \geq 1$ (equality holds for support vectors)
& for blue points (\vec{x}_i) $\rightarrow \vec{w} \cdot \vec{x}_i + w_0 \leq -1$ (equality holds for support vectors)

Now we find d maximising it of selecting the correspond. hyperplane.

Converting the two constraints to a single constraint:-

Let $y_i \rightarrow +1$ for pt (green pts)
 $\rightarrow -1$ for -ve (red pts)

The constraint becomes:-

$$y_i (\vec{w} \cdot \vec{x}_i + w_0) \geq 1$$

Equality will hold for support vectors



So given the constraint

Page 106.

$y_i(\vec{w} \cdot \vec{x}_i + w_0) \geq 1$ we need to find eqn of hyperplane $\vec{w} \cdot \vec{x} + w_0 = 0$ with max d.
and for support vectors :- $y_i(\vec{w} \cdot \vec{x}_i + w_0) = 1$.

So projection of $\vec{x}_2 - \vec{x}_1$ vector on a unit vector \perp to π_+ or π_- or π , will give us d.

If unit vector \perp to $\pi = \frac{\vec{w}}{\|\vec{w}\|}$

$$\therefore d = \frac{(\vec{x}_2 - \vec{x}_1) \cdot \vec{w}}{\|\vec{w}\|}$$

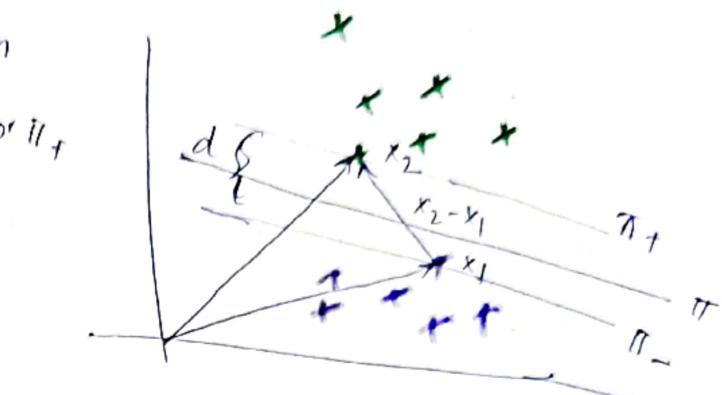
$$d = \frac{\vec{x}_2 \cdot \vec{w} - \vec{x}_1 \cdot \vec{w}}{\|\vec{w}\|}$$

$$d = \frac{(1 - w_0) - (-w_0 - 1)}{\|\vec{w}\|}$$

$$d = \frac{2}{\|\vec{w}\|}$$

So we need to max d

So we need to minimize $\|\vec{w}\|$



$x_2 \rightarrow$ support vector on π_+
 $x_1 \rightarrow$ support vector on π_- .

So for x_2 .

$$y_i(\vec{w} \cdot \vec{x}_2 + w_0) = 1$$

$$1(\vec{w} \cdot \vec{x}_2 + w_0) = 1$$

$$\vec{w} \cdot \vec{x}_2 = 1 - w_0$$

for x_1

$$y_i(\vec{w} \cdot \vec{x}_1 + w_0) = 1$$

$$-1(\vec{w} \cdot \vec{x}_1 + w_0) = 1$$

$$-\vec{w} \cdot \vec{x}_1 = -1 - w_0$$

$$-\vec{w} \cdot \vec{x}_1 = \vec{w} \cdot \vec{v}$$

Note target originally.

$$\pi_+ : \vec{w} \cdot \vec{x} + w_0 = \alpha$$

$$\pi_- : \vec{w} \cdot \vec{x} + w_0 = -\alpha \text{ like hole}$$

$$\text{then } d = \frac{2\alpha}{\|\vec{w}\|} \text{ note .}$$

So $\underset{(\vec{w}, w_0)}{\operatorname{argmax}} \frac{2}{\|\vec{w}\|}$ such that $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$.

was perfect separation b/w the two categories of data. This will not work if the data is mixed up (has impurity)

This type of problem is called Hard Margin SVM.

Soft Margin SVM

What is the data is not perfectly split (ie they are mixed up) Then the problem cannot be solved using Hard Margin SVM. → Generally we will get data such that either both & neither are linearly separable. So we generally use soft margin SVM.

For Hardmargin SVM we had

$$\underset{(\omega^*, w_0^*)}{\text{argmax}} \frac{2}{\|\omega\|} \quad \text{s.t. } y_i (\vec{\omega}, \vec{x}_i + w_0) \geq 1.$$

Also we know max $f(x) \Leftrightarrow \min \frac{1}{f(x)}$

So $\underset{(\omega^*, w_0^*)}{\text{arg min}} \frac{\|\omega\|}{2} \quad \text{s.t. } y_i (\vec{\omega}, \vec{x}_i + w_0) \geq 1$

In Softmargin SVM: we have:-

$$\underset{(\omega^*, w_0^*)}{\text{arg min}} \left(\frac{\|\omega\|}{2} + C \sum_{i=1}^n \xi_i \right)$$

Regularization term [by comp. with Logistic Regression]

Ek pot (green) point galat classified hoi toh uska ξ_i is uska fr dist for Π_+ hyperplane f ex we (blue) point galat classified hoi toh uska ξ_i is uska fr dist for Π_- hyperplane.

and Minimising $\sum_{i=1}^n \xi_i$ means

Minimizing the error (galat classified points)

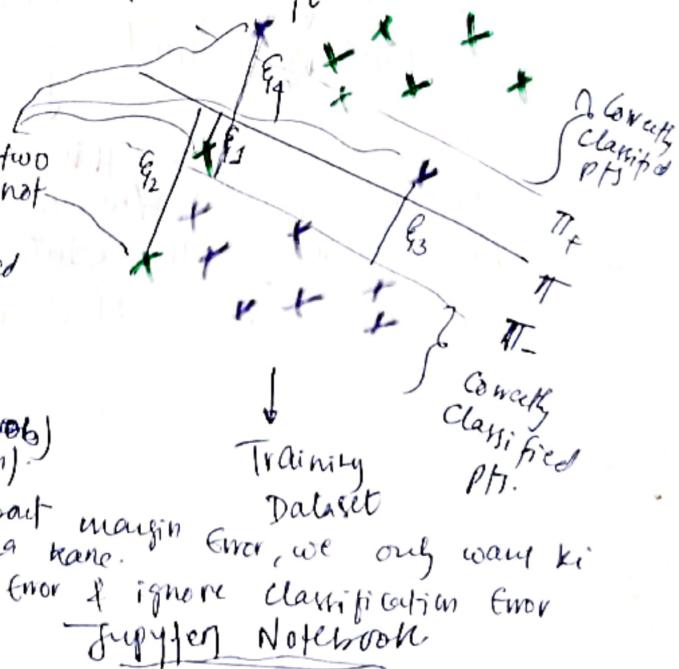
so $\frac{\|\omega\|}{2}$ is called margin error

& $\sum_{i=1}^n \xi_i$ is called classification error.

& C is a hyperparameter (tuning knob) ($\alpha = 1/\lambda$; λ from Logistic Regression).

if C is big → mean we are not caring about margin error, we only want ki Algo koi bhi point ko misclassify na kane. and if C is vr small → focus on margin error & ignore classification error Matlab such misclassification chalega.

These two points are not correctly classified points



Jupyter Notebook