

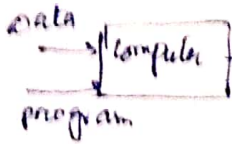
Machine Learning (ML):-

Page NO.: 1

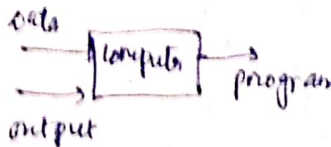
- It is a field of Computer Science that uses statistical techniques to give computer systems the ability to "learn" with data, without being explicitly programmed.

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 numbers, 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 ^{and so on some features} again 3 nos and its sum 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 4 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 advertising company changes these words to Big the ML algo will recognize it (since it has been trained on a large data set) 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 on 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

Kabhi kabhi kuch info in the data bohot hi jyada hidden hote hai; jo sirf visualization se dikh nahi pata.

Bohot jyada hidden jo patterns hote hai 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 inme se agar hum kuch important cheez nikal pa rahe hai apne data ke Baare mei jo Data Analysis se pata nahi Chal paya tha then we call this Data mining.

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

* ML Bohot pehle hi aagya tha but isko recognition bohot late mili since one of the major factor on which ML runs is Data & tab Data jyada tha nahi ya use store nahi kar pate the and agar use store kar bhi le then utne strong hardware/machine nahi tha jo un algos ko itne same data pe run karwa paye But with the evolution of Internet - These problems are solved.



around 1950's

(Alan Turing)

(Symbolic AI)

(Expert Systems)

↓ what is it?

AI (Artificial Intelligence) :-

→ machines mei kisi tarike se intelligence dalna.

This is what we are trying to do AI to do

→ Human intelligence is a very complex thing

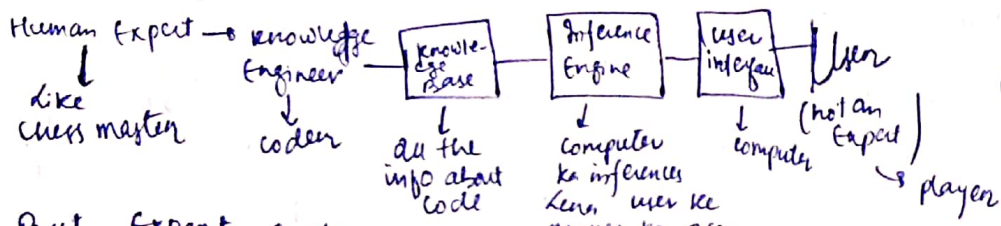
Pattern Recognition (Image)

creativity and Imagination

Emotional Intelligence

These are in Human and are Very complex.

(Since they are very hard to quantify)



But Expert systems are very closed problems based if you give it a fuzzy logic it will fail. (like dog detection in given images)

After Expert systems → we had ML.

ML (Machine Learning) → Already Discussed on page 1

→ Diff b/w ML & symbolic AI is in ML we donot do explicit programming (har case ke corresponding code likhna)

In ML → we have data (input + output). → we feed it to our system (algo) and it's 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 karna system ko — system rules/pattern form karega and based on these patterns/rules system new data pe predictions Karega.

→ It has Algorithms and these Algorithms are inspired from Biology (Neurons) (but DL doernot 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
hoga ya nahi then in ML the features ^(created by us) has to be given
to the ML model when training the model

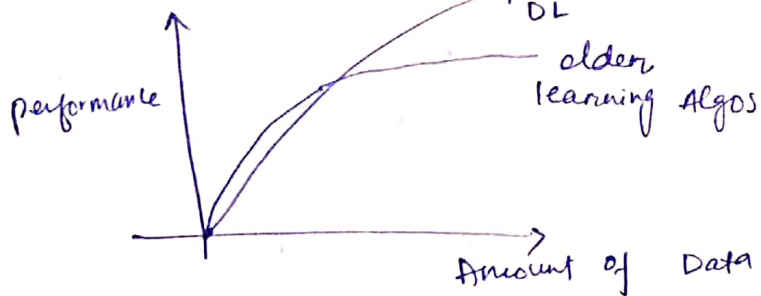
↓
And this features will be
created by the expert
in the field.

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/
prediction power improves. (Since har layer ke saath aur deeper
in the data jo chupa hua hai
wo apan nikal pate hai)

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.

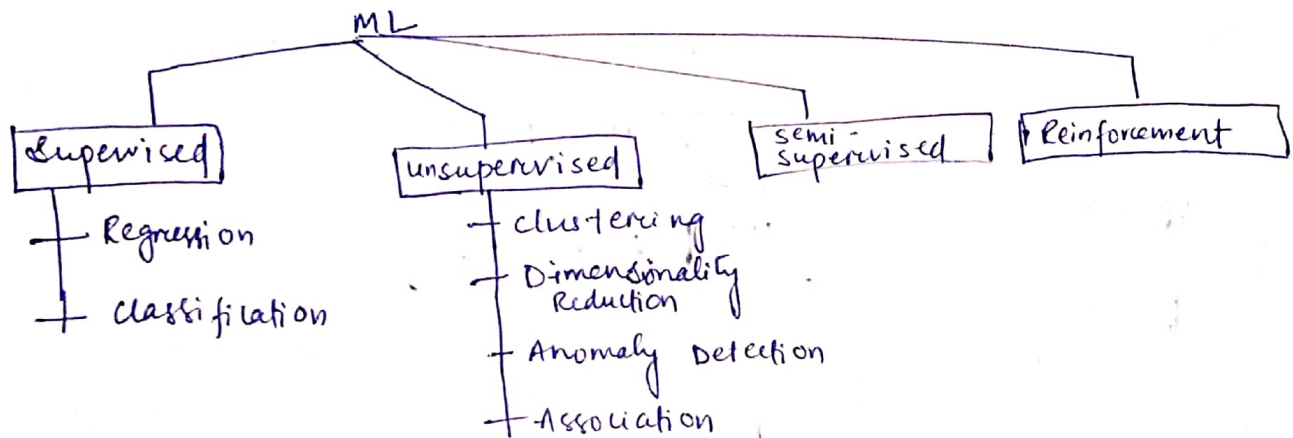
- * Small data pe ML hi better accuracy de dega
- * on large set of data we use DL

TYPES OF ML

— ML ke types alag alag cheezo pe depend karke hai

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. if in Data we have (input + output) and we want to find a relⁿship b/w input & output so that ek naye input ke liye aap output bata sakte hain then this is called Supervised ML

Suppose we have 5000 students Data (input + output)

	IQ	CGPA	Placement
87	7.1	Y	
111	8.9	Y	
75	6.3	N	
⋮	⋮	⋮	

For each we have: IQ | CGPA | Placement →

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

Now we want to draw some mathematical Relⁿship 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 Relⁿship nikalna hai such that new input data ke liye output predict kar sakte hain → 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 kitno ka laga

IQ	CGPA	package
87	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 beane mei bahot kuch bataya hua hai you have to predict mail spam hai ya nahi.

(Classification)
(iii) Saare wheater conditions ko dekhkar predict karne ki aaj barish hogi ya nahi (Classification)

(iv) Alag Alag images ko dekhkar predict karne ki image mei kitne dogs hai - (Regression)

Hence:

Supervised ML

~~Example~~

Input + output data

→ output Numerical → Regression
→ output categorical → Classification

Unsupervised ML

↳ Isse basically bas input data hoti hai output data nahi hoti.

Since output hai hi nahi toh yaha Task prediction toh ho hi nahi sakta since we don't know ki predict kya karana 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 Detation
- + Association Rule Learning.

Clustering:-

we can plot the input data (IQ + CGPA) on the 2-D coord system



Fact Now we can ourselves create a (new col) Label Data

0 →	more IQ + more CGPA category
1 →	more IQ + less CGPA category
2 →	less IQ + more CGPA category.

clustering Algorithm ye detect kar sakta hai ki konse students same group mei hone chahiye Isse mei apne students ko different categories mei divide kar sakta hoon.

Toh isse hum kisi bhi incoming student ko categories mei dhal sakte hai accordingly.

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

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 andhar bhi clustering lagakar clustering karana.

Dimensionality Reduction:-

Sometimes kisi problem me input data me bahot sare columns ho jenge (like 1000 input col's) — in case the algo will run slow and endly expt. ke baad jyada input col dhalne se result me 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 bathrooms
...	...

then Dimensionality Reduction will combine this col's to a new col's 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 Rel'ship that we had in higher Dimensions.

ANOMALY DETECTION.

Kuch gadbad (outliers) hai usko detect karna hai & remove karna hai.

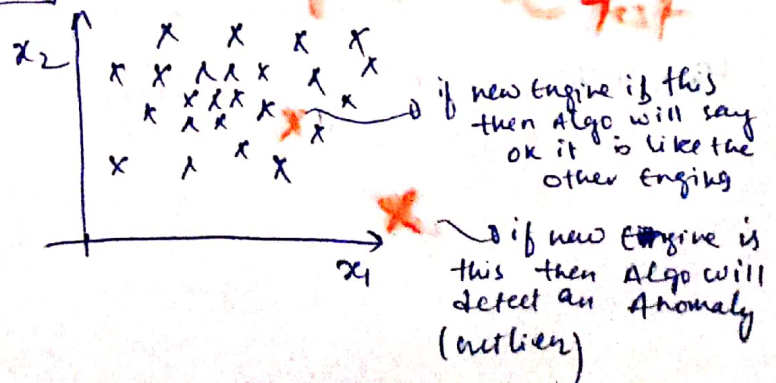
↳ manufacturing me defect, credit card processing defect, loan approval defect, etc.

Data Set : $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

New Engine: x Test

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 konsa product rahega. (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 bhi 80 times milk kharida gaya toh 60 times eggs bhi buy gaye then hum milk and eggs ko ek saath agal bagal rakh denge (since there is a strong association between milk and eggs)

Interesting case study in US:- Beer + Baby Diaper

SEMISUPERVISED → partially supervised & partially unsupervised

The label creation (output creation) create karne mei for the data set on which Algo will be trained, 1 effort lagta hai (like at the end kon batayga is dog hai ki nahi picture mei, and kon batayga ki placement hua ya nahi)

Hence label create karne 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 (cor) 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 koun Data ko manually label karen and baaki ke dataset khud label ho jai is the key idea behind semisupervised ML.

Ex:- Google photos:- jab hum photos click karke hai toh wo ye automatically identify karne lagta hai ki is photo wale aadmi ko mei aur 4,5 photos mei dekha hai toh wo iss photo ko sabse ke saath clustering/group kardega and agar hum EK photo ko label (mummy) kare toh automatically us group ke baaki saare photos par bhi same label (mummy) kardega.

REINFORCEMENT ML

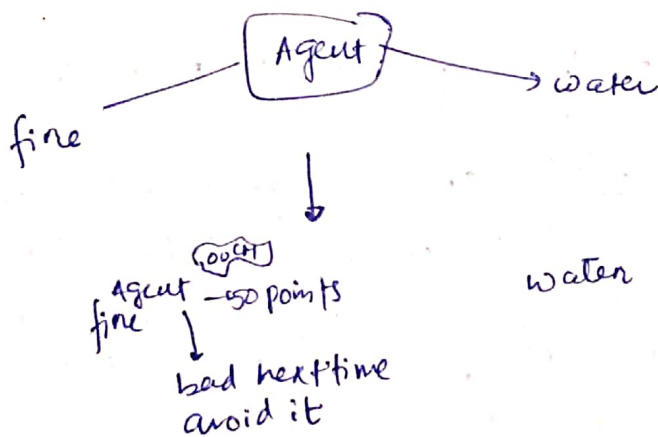
Hum Data dele hi nahi hai - system ke paas data kaha hi nahi hai.

Humara Algo exdum Scratch se start karta hai seekhna q dheere dheere improve karta jata hai (like us human being) Like hum college life mei humare paas pehle se data nahi hai ki kaisa behave karna hai but hum dheere dheere galtiyan karke seekhte hai.

Ex: Self Driving car :-

pehle se koi data nahi hai, wo road pe utar kar gradually learn karta hai ki kaise chalni hai

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



① observe

② select action using

Agent ka Rule Book like it may have go to fine

③ Action

④ Get reward or penalty like fine has penalty

⑤ update policy (learning) like don't 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 of the game straight 4 out of 5 games.

for Go Game (Chinese game more complex than chess)

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

Server ka environment jaha par code run kar raha hote hai use hum production environment bolte hai and jab hum us code ko nahi hote hai apne machine pe toh hum use development environment bolte hai.

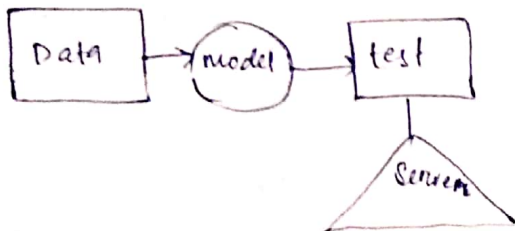
ML on servers behave differently & based on that ML is divided into two categories:

BATCH (OFFLINE) ML VS ONLINE ML

BATCH ML

(there is no incremental learning i.e. chote chote tukano pe train nahi kar sakte)

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)



↳ since the data set is entire (large).

The problem here is the model is static :- Ex baar jo wo sikhi gya wo sikhi gya, model kuch naya nahi sikhi raha hai, wo purane training ko dam pe hi kaam kar raha hai 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 hai but Agar Netflix New movies dhal de DB mei then ye model in naye movies ko recommend nahi kar payega.

Hence, in a way apna ML model ko constantly evolve karne rehna padega Data ke Saath.

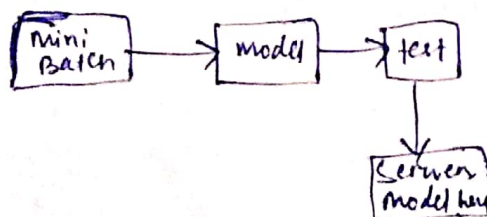
So in Batch ML after New Data is added (in a frequency of 24 hrs or 1 week or 1 month or 6 months or etc.) we pull down the model Retrain it on the updated Data (old + new data) then test it & then deploy it again on the server, periodically...

Disad. of Batch ML

- Lots of Data (may be data bahut fast pr (update) ho raha hai in Batch ML Limitation is ki purane 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 uska instant connectivity nahi hai toh updated data pe periodically retrain karana difficult hogai)
 - Availability (let's say hum ek social networking website par kaam kar rahe hai & humne ek model banaya to newfeed generate karke hai on basis of user interest & we have used Batch ML on 24 hr period ab agar achanak koi breaking news aagayi ya koi baat fail hogi toh hamara model use show hi nahi kariga. Since wo 24 hrs baad new data pe train hone ke baad hi ye newfeed pe show kariga) & taen hamara newfeed us breaking news ka bhar jayga joki ab itna important bhi nahi hai.
- jisme hum interested hai search kar rahe hai hence new data feed kar rahe hai model ko)

ONLINE ML

Online ML is done incrementally matlab jo training hai woh incremental hota hai and hum chote chote batches mei apne model ko data feed karke hai (min batches) sequentially & Har training/batch ke baad model improve krta jata hai and since the mini batches have small chunk of data, ye training hum production server pe bhi karwa sakte hai.



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

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

- Youtube feeds personalize hota 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

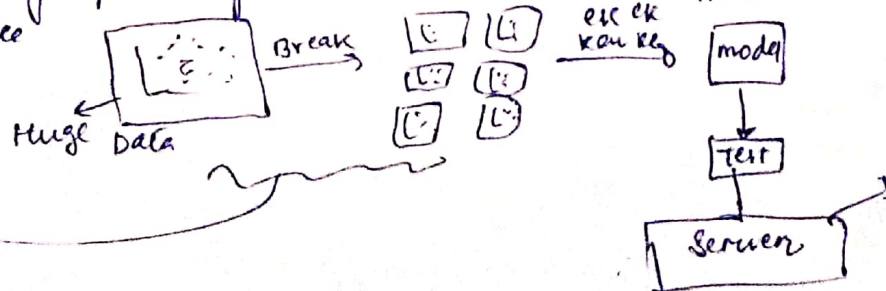
River's ambition is to be the go to ~~ambition~~ library for doing ML on streaming data.

Tricky Thing to set

— Learning Rate: in online ML, Learning Rate ye decide karata hai ki kitna frequently hum model ko new data pe train karenge since new data inflow toh consistent hai humne aisa Learning Rate rakha hota hai ki apna model purane cheeze bhi yaad rakhe & nayi cheeze bhi sikhte 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.

ye kaam hum offline hi karke hai just concept online Learning ke tarah hai



Server pe model ko Batch ya online rakha 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 (∵ naye data ke according mere model ka behaviour change hone lagta hai & agar data ki galat (spam) aane lage toh apna model kafi biased hojaye ga.
solⁿ is ∴ An active monitoring system using maybe some anomaly detection algo like kisi bhi anomaly data ke aate hi system ko offline karde ya us data ko reject karde.

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
Fewer computation, single time batch-based training	Computational power	Continuous data ingestions results in consequent model refinement computation
Easier to implement	Use in product	Difficult to implement & manage
Image classification on anything related to ML - where data pattern 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, Spark MLlib	Tools	Active research / New project tools - eg ∴ MOA, SAMOA, SciKit Multiflow, Stream BM

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

- + Instance Based ML
- + Model Based ML

Learning in ML $\begin{cases} \rightarrow \text{memorizing} \rightarrow \text{Instance Based Learning} \\ \rightarrow \text{generalizing (concept/underlying principle/patterns Samajhna)} \rightarrow \text{Model Based Learning} \end{cases}$

* Future mei jab ML Algo padho then ki wo model Instance Based Learning try to identify kar raha hai ya Model Based Learning kar raha hai.

LAZY LEARNER

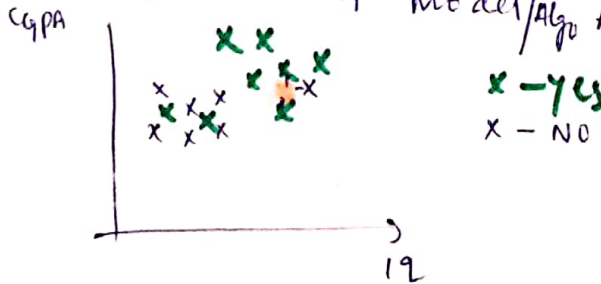
Instance Based Learning

Consider we have Dataset:-

input		output
IQ	cgpa	placement (Y/N)
80	8	Y
70	7	N
...	...	...

→ output is Categorical
∴ Classification Problem.

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



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 input/query point **103, 7.5** as shown above then we look at its nearest neighbours (3, 5 etc depends on us) and we find ~~the~~ query point which ~~is~~ type of neighbours are more (here green) so we say the output of the query point will be like green neighbours is: yes.

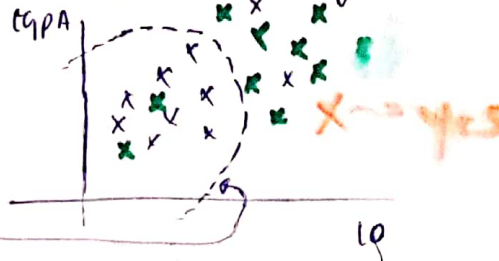
↓
This is the logic of the KNN Algorithm (k nearest neighbour)

Note in Instance Based Learning, jab tak naya point nahi aaye, tab tak apne model ne kuch kiya hi nahi jab uske paas Ek query point (input new) aaya tab uske some 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.

IQ	CGPA	placement (Y/N)
80	8	Y
70	7	N
⋮	⋮	⋮



- Yes
- No

Here 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 ^(between input & output).
Ye apni model ko data ke upar train karke sikha. Called model parameter.

Most of the ML also use this approach: $\rightarrow$ finding a mathematical
 Relⁿ b/w input & output & result is a decision function $\left\{ \begin{array}{l} \text{Exp in classification} \\ \text{problems} \end{array} \right.$

★ Now after getting the Decision fun, even if we remove all the training Data but the Algo will be able to predict an output for a given input Data but this is not the case with Inference 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

MODEL BASED
(usual/conventional ML)

(data preprocessing)

- Prepare the data for model training
- Train model from training data to estimate model parameters i.e. discover patterns

INSTANCE BASED

- prepare the data for model training
(no split here)
- don't train model. pattern discovery postponed until scoring query received.
new input data / query point

- 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
- Storing model generally requires less storage
since bas us for store karne hai

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 observation
- May not have explicit model form
→ pure data set ko store karne hai
- 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 Webscrapping.

Insufficient Data / Labelled Data

Say 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 of data 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 hamare paas bahot jyada data hai toh farak nahi padta ki hum kounsa Algo use kar rahi hai, this is called the unreasonable effectiveness of data.

But we have ^{mostly} small or medium data $\therefore$ Algo matter karta hai.

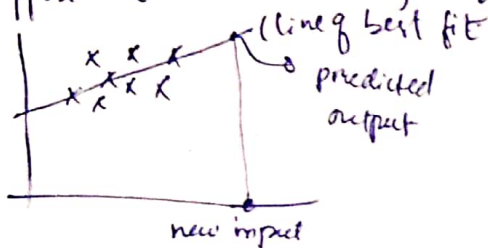
Again: Agar hum data collect kar bhi kr, the labelling it (like cat hai ya nahi in picture) is a tedious job.

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

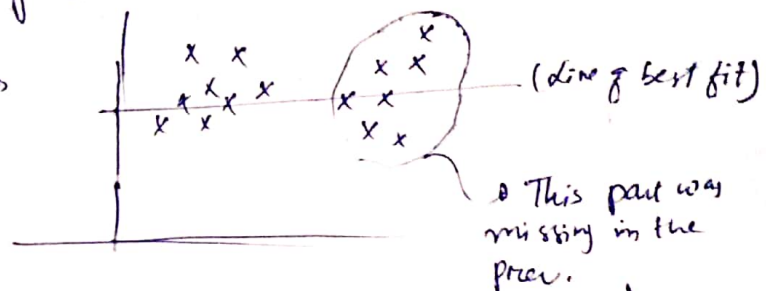
— Non Representative Data:

Suppose we are solving a ML problem but hamare pass jo Data hai wo aadhi kahani hi batata hai Ex:-

Suppose we are solving Linear Regression



But the complete data was



Kabhi kabhi jab hum Data gather karke hai toh Sahi jagah $\hat{e}$ gather na karke kisi ek jagah $\hat{e}$ utha lete hai toh this data problem ka proper Representation nahi deta

Like: we want to survey ki next 120 worldcup konsi team jitegi:-

Agar hum bas indian logo $\hat{e}$ hi puche toh wo definitely India bolenge (biased) toh ye jo data collect hua wo non-Representative data hai. This is called Sampling Noise

valid/proper data ke liye hume survey har us country mei karana chahiye jo team khet rahi hai.

Kabhi kabhi Agar Sab jagah se data le toh bhi dikkat ho jati hai like: Agar hum har country mei bhi puche toh Sabse india bola karega waha bhi mostly indians hi hai — This is called Sampling bias

Sahi representation of data ke liye:- 100 Ind, 100 Aust, 100 Eng ... $\hat{e}$ puchna hoga.

(imp)

Poor Quality Data :-

Data mei bahot saane outliers hai, missing values hai, abrupt values hai, alag alag format mei values hai and in sab cheezo ko sahi karne mei bahot jyada mehnat lagti hai.

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

Irrelevant Features (Columns)

Apne Data mei kuch aise columns hai jo kuch contribute hi nahi kar rahi apne analysis mei - toh apna model accha perform nahi karega

IN ML :- There is a phrase :-

Garbage IN, Garbage OUT :- Agar ML model hi kuchina dhalegi toh output kuchina hi milega.

Suppose we want to Analyse which age group are more fit & data we have
Age | weight | height | Location the Location col will not add anything to our analysis hence we remove it.

Kabhi kabhi hum do features ko mila ke ek single feature bana dete hai (Yaha BMI) jo same analysis dega - And this is called

Feature Engineering

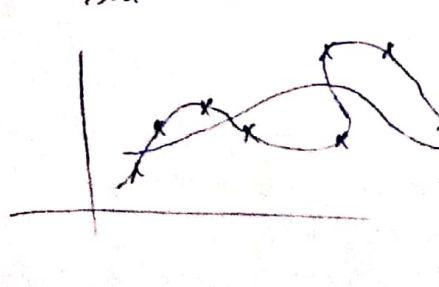
Over-fitting: (Big villain in ML Algos)

Humne ek ML model ko kisi data pe train kiya and us model ne data ko rat liya (matlab concept/pattern nahi sikha) toh agar is model ko hum naya data de toh iska performance accha nahi rahega since model ko bas purane data pe hi kaam करना ata hai.

Ex: Guwahati jga - so movie - ticket - set - we assume Guwahati is costly But aisa nahi hai hum bas movie ticket ke basis par pure city ke prices predict kar rahi hai

slly even ML Algo training data set ko dil di lage lete hai ki jo training data set mei hai wohi sahi hai and baaki sab bhi waisa hi hai

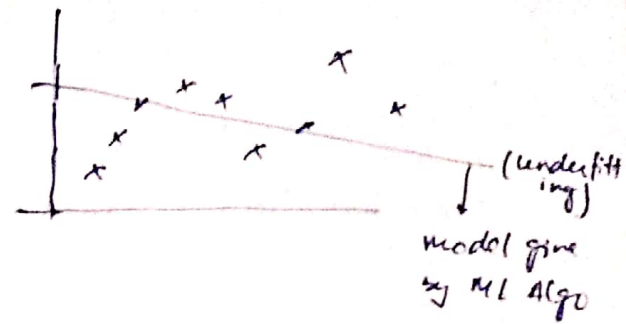
Asa kuch Actually fr samachariye tha.



(overfitting) -> jo humare ML model ne banaya it is trying to touch every data point in the training Data set

— underfitting. (opposite of overfitting)

↳ ye na training data pe achha result dega & nahi new data pe



— Software Integration

End goal of a 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 hai - windows, Android, Linux, some old OS, server OS etc.

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

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

— Offline Learning/Deployment

↳ Batch Learning wali problem (Ek baar model train hokar server pe rhi toh wo update nahi hota - update karne ke liye wapas offline lao - new data pe train & test karo & then phir upload on server) isse deployment mei bahot cost lagta hai & is very difficult

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 ^{kyu} are 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.

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

* 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 increases 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 kounsa 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 kounse product ka stockup karna chahiye & kounse 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. (weird) — Actually using these phone number they do data profiling, ki ye aadmi kounsi cheeze jyada khareidta hai (is he health conscious etc) and then sell it to different companies who do target marketing (Same is Done by Google, facebook, etc.)

* Positions of products in Supermarkets is done by ML (Association Rule Based Learning)

Banking & Finance:

* Loan Babhi ko nahi milta — pehle apna profile submit karna padta hai — iss profile ka Analysis hota hai at two levels $\rightarrow$ ML Algo — Analysis by ML algo — this profile is compared by the past defaulters (jinhone loan nahi and again strong correlation aaya then may be you won't get a loan & your profile will be rejected and if correlation is low then the profile is sent to loan officer who checks the profile manually & loan is sanctioned if good.

* Kaha pe Branches khole, customers ke liye kaha pe kounsa promotion plan start kare ... etc.

* Share market / Trading .

Transportation - OLA

• Din ke kisi particular time pe point A se point B jane ka price bahot high kar diya jata hai jabki wahi price din bhar 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 & those cabs in the red region take the customers but those cab outside of red region, why will they go extra distance to pick customers at the same price? 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 unit due to per unit 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 monitor 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 - its 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 2019 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 apni personality ko hi kind of extrapolate karta hai) (do the tweets are actually valuable)

∴ 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 hai ammoro se lekar)

and these stock brokers un companies mei paisa lagaygi jo already predicted winning party ko support kar rahi hai. Ab agar sach mei predicted party jeetgi toh local public bhi uski company mei paisa lagaygi & stock prices rise and stock brokers ko profit hoga since they already have those shares

→ just like SDLC (Software Dev. Life Cycle)

Machine Learning Development Life Cycle (MLDLC/MLDC)

Page 12.

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 kitna lagaga, 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 hota.

Data

- CSV file
- from some API in json format
- web scraping
- Database
- Big data in tool spark in clusters

 — we make a datawarehouse (ETL Extract Transform Load) — then fetch 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 leker ana chahiye hamara ML algo easily consume kar paye.

— EDA ~~★~~ (jitna time yake spend karo 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.
 then we make lots of experiments with data, extract the relationships hidden inside the data.

- visualization by plotting graphs
- univariate Analysis (basically col ke upar independent analysis)
 mean, std, variance, curve etc
- Bivariate Analysis (2 columns ke beech ka analysis)
- multivariate Analysis (multiple columns ke beech ka analysis)
- outlier detection
- Imbalanced data set — Balanced data set

(like cats & dogs ke # of images same nahi hai for image classification)

★ Feature Engineering & Selection → existing features mei kuch intelligent changes karna so that Analysis easy hojai.

Feature Engg: Features matlab — input columns

Features are imp since output input cols (Features) pe hi depend karta hai
 Idea is kahi kahi kuch naye cols (Feature) hum create karte hai apne hisab se taaki apna analysis easy hojai
 ex: # of rooms + # of bathrooms → combine → sq. feet area for rooms & bathrooms (new input feature)

Feature Selection: kahi kahi data set mei bahot hi jyada features hote hai (100 or 200) then hum ussno features ko lekar aage nahi badh sakte because janani nahi ki har input feature output ko impact kare. So we remove such input col/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

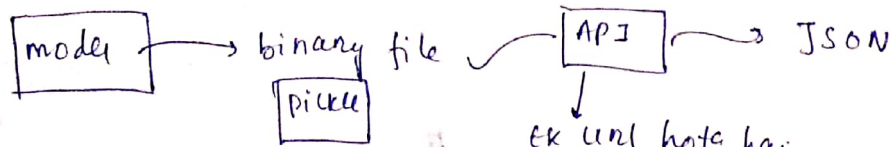
After we have the proper data set, we bring different ML Algos and feed them the data & train (we never actually know koi algo kitna achha niklega kisi particular data ke liye) (nahe bayes perform better on text data but ho sakta hai ki koi aur algo uss achhe perform karjai you never know)
 and ussko Evaluate karte hai ki uska performance kaisa hai using the performance metrics (this tells how good our models are performing)

And then finally we select one or more algo & then tune their best parameters (settings of the Algos) to enhance the performance (hyper parameter tuning)

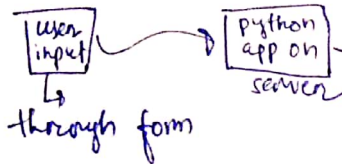
★ Ensemble Learning — multiple ML Algos ko jod ke ek naya powerful method/algo hai

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)



ek url hota hai
jaha pe aap sahi
input do toh wo apko
json data dega



this is sent to the API where the binary file makes the prediction & that prediction in JSON format is given back to the user.

Testing (Beta Testing)

↳ kuch Trusted customers
and ask for feedback

ke paas ya ^{software} model bhjto hai

A/B Testing

↳ isse ye decide karte hai ki
humne jo model banaya wo kitna
sahi se kaam kar raha hai
agar sahi se kaam nahi kar raha
then hum piche wale steps ko
repeat karte hai

Optimize:-

Server pe Launch karne se pehle hum kuch steps karte hai

- + model Backup (✱)
- + Data Backup (✱)
- + Load Balancing (Bohot jyada log aa rahi hai to unko divide karke same ya kam time mei request serve kare)
- + How frequent we need to retrain (Rotting prevent karne ke liye)
 - ↳ This needs to be automated (since overtime data evolve karne lag jata hai)
- + Automation setup ki agar model fat gaya then use roll back karke wapas se chla karna

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 & bhi store kar leta hai
Ex: vectors, matrices are Tensors.

0 dim Tensor is a scalar (ex no.)

1 dim Tensor - List of Nos - vector

2 dim Tensor - Matrices

3 dim Tensor :- ...

General term
Tensor

0 dim

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

(2) (3)

Scalar - 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 andar array

3-D arrays → array ke andar array & uske andar bhi array & so on.

* 1D Tensors:-

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

→ $ndim = 1$

means the dimension

* in Tensors term:- what is Axis:-

say if we have 2-dimensions 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 sakte ho.

↳ arr.ndim = 1

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

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

2 D Tensor - " " " 1D Tensors

3-D Tensors $\rightarrow$ " " " 2D Tensors
 $\rightarrow [[1, 2, 3], [3, 4, 5]]$
 $\rightarrow [[1, 2, 3], [3, 4, 5]]$

* 2 D Tensors / Matrices:-

1D Tensor:- $[1, 2, 3]$ $[4, 5, 6]$ $[7, 8, 9]$

2-D Tensor
 matrix $\rightarrow$
 $\downarrow$
 n-dim = 2
 Row $\rightarrow$ col
 $\begin{bmatrix} 1, 2, 3 \\ 4, 5, 6 \\ 7, 8, 9 \end{bmatrix}$

//
 # of axis
 //
 Rank

creating 2-D Tensor using
numpy:-

mat = np.array([[1, 2, 3],
 $[4, 5, 6]$,
 $[7, 8, 9]$])
 $\downarrow$
 mat.ndim = 2

* 3-D Tensors:-

collection of 2-D Tensors
 3-D
 4 x 3 x 3 Tensor
 Row
 Col
 Depth

3-D
 4 x 3 x 3 Tensor
 $\downarrow$
 3 axis (Row, col, depth)
 //
 3 Rank

slly:- 4-D Tensors $\rightarrow$ collection of 3-D Tensors.

vector (1D Tensor)
 $[1, 2, 3]$
 Hq Axis = 1 $\rightarrow$ rank
 shape = (3,)
 size = 3

IN ML 0 to 5D tak hi kaam ayega

Rank, Axes & Shape, Size of Tensor

Rank = # of Axes = # of dimension = ndim

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

Ex: $\begin{bmatrix} \square & \square \\ \square & \square \\ \square & \square \end{bmatrix} \rightarrow$ 2 dim
 shape = (4, 2)

Ex: $\begin{bmatrix} \square & \square & \square \\ \square & \square & \square \end{bmatrix} \rightarrow$ 2 dim
 shape = (3, 3)
 $\begin{bmatrix} \square & \square & \square \\ \square & \square & \square \end{bmatrix} \rightarrow$ 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
 $a.\text{shape} = (4,)$

But say we have its 2D version:-

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

PRACTICAL EXAMPLES

— Examples of 1D Tensors

Students data set: (10000)

Say we have only two states
 $1 \text{ or } 0$
 $\text{Assam} = 1$
 (Numerical/Label Encoding)

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
 $\rightarrow \text{ndim} = 1$
 Tensor/array

But it is a 3-D vector

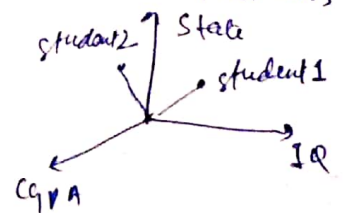
$\therefore$ in terms of vectors it is a 3-D vector

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

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



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

Examples of 2-D Tensors

Students Data (10000 students)

cgpa	iq	state	placement
9.2	101	0	
8.7	97	1	
...	...	...	...
-	-	-	-

input data

student 1 input data

2 dim Tensor

10000th student input data

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

placement: $[1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \dots]$

1-D Tensor or 10000 dim vector

* Examples of 3D Tensors

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

Ex:- Hi Gautam
Hi Neeraj
Hi Jatin

→ This is the text we provide for training to the model & we know ML algo is mathematics (does not understand text) ∴ we convert this text to a vector (called vectorization in NLP)

Set of all unique words

Hi	Gautam	Neeraj	Jatin
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

0 → Hi
0 → Gautam
0 → Neeraj
1 → Jatin

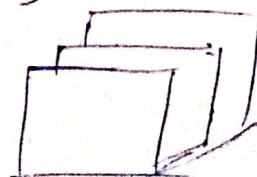
Now:- Hi Gautam → $[1, 0, 0, 0]$, $[0, 1, 0, 0]$

→ 2-D Tensor
∴ collection of 1-D Tensors

Similarly, Hi Neeraj → $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Hi Jatin → $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

∴ Hi Gautam
Hi Neeraj
Hi Jatin



→ ndim = 3 = # of Axes.

→ 3-D Tensor
3 x 2 x 4 → shape

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

Kisi stock ka

Highest / Lowest price

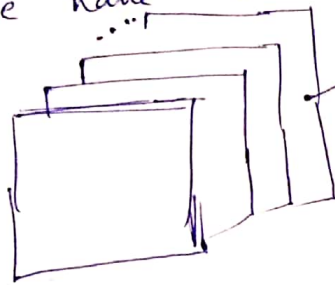
Every day for 365 days.

Day 1 $\begin{bmatrix} - & - \end{bmatrix}$
 Day 2 $\begin{bmatrix} - & - \end{bmatrix}$
 ...
 Day 365 $\begin{bmatrix} - & - \end{bmatrix}$

Aisa data jo EK frequent time period pe collect karte ho

Ex: Sharemarket Data

Now: say we have same of 10-years data



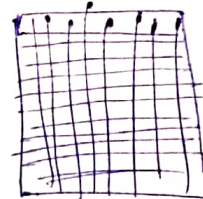
3-D Tensor
 shape = (10, 365, 2)

In Time Series data this is time axis & the data is in 3-D Tensor

Examples of 4-D Tensors

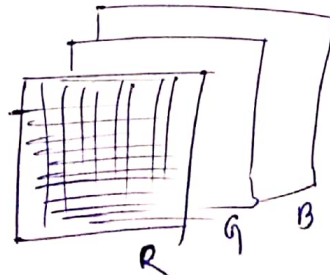
(Collection of Image) data → in Computer vision Domain

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



& har pixel
 wajah se hume

1 Colour Image:-



3-D Tensor
 shape = (3, 1200, 800)

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

Examples of 5D Tensors: Collection of Videos → Collection

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

1 video → 60 sec at 30 fps, 480p (480x720 Resolution)
 1800 images

(1800, 3, 480, 720)

say we have 4 video clips →

(4, 1800, 3, 480, 720) → 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 rahi hai Netflix unko rok le.

Churn rate:- jitne lhi apke platform ke given period (monthly/yearly) ke jaa rahi hai. means churn rate users hai unme kithe customers platform chod

(mostly New customers ka aane ka rate is more than churn rate & then users 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 $\rightarrow$ supervised classification problem. (by offering instant discount on next month subscription)

Again:- kuch log aise honge jisko platform chorne ka man jyada hoga & kuch log aise honge jisko lag raha hoga ki chor dete hai (kam pissed off honge) $\rightarrow$ Hence sabhi ko barabar discount kyu dena

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

$\therefore$ 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.

4. 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 laga raha hai toh apne man pasand ke cheeze nahi mil rahi
- 2/ Search but did not find (mahine mei kitne searches kiye & nahi mile)
- 3/ Content left in the middle → means content is not much tempting
- 4/ Clicked on recommendations (order of recommendations)

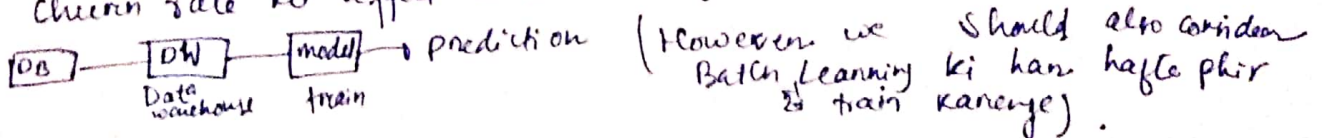
Now Data Engineer Netflix ke Database mei Δ ek Datawarehouse banayga jisme ye saare cols honge

5/ Metrics to Measure : → jo hum kar rahi hai wo sahi hai ki nahi kaise pata karoge ki humne jo model banaya hai wo successful hai ki nahi → humne jitna churn rate predict kiya & jitne log chot kar gaye isme diff bahot kam hai then model sahi hai.

→ jitko predict kiya chot kar jaiga wohi gye — means model sahi hai.

6/ Online vs Batch :-

Here online will be better since bahot saare factors churn rate ko affect karle hai & wo upar niche hote 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)

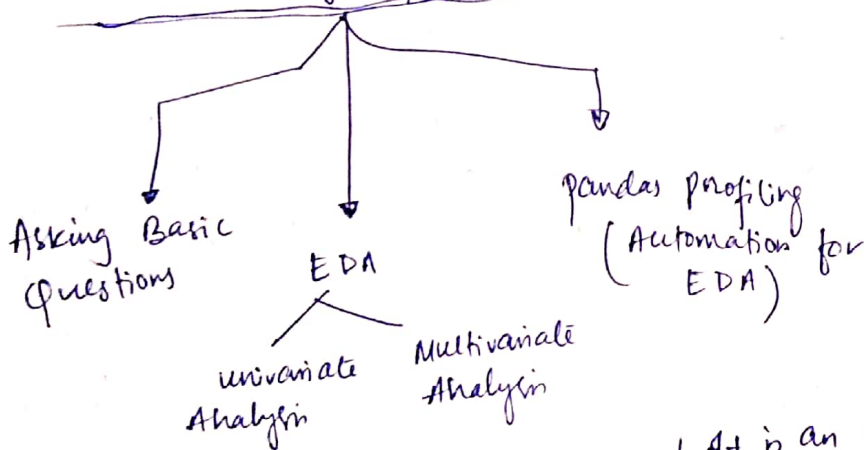
2./ Data Gathering:-

Note:- "Bad Algo with huge Data will work/perform better than Good Algo with less Data".

- + working with CSV file
- + JSON | SQL Data (JSON:- Javascript Object Notation)
 - ↳ API ke through Data lekar aate hai toh woh JSON format mei hota hai.
 - ↳ Database se data nikaloge then it is SQL
- + fetch Data from An API
- + web scraping

Laptop → folder Machine Learning on Desktop - Jupyter Notebook.

3./ understanding your Data:-

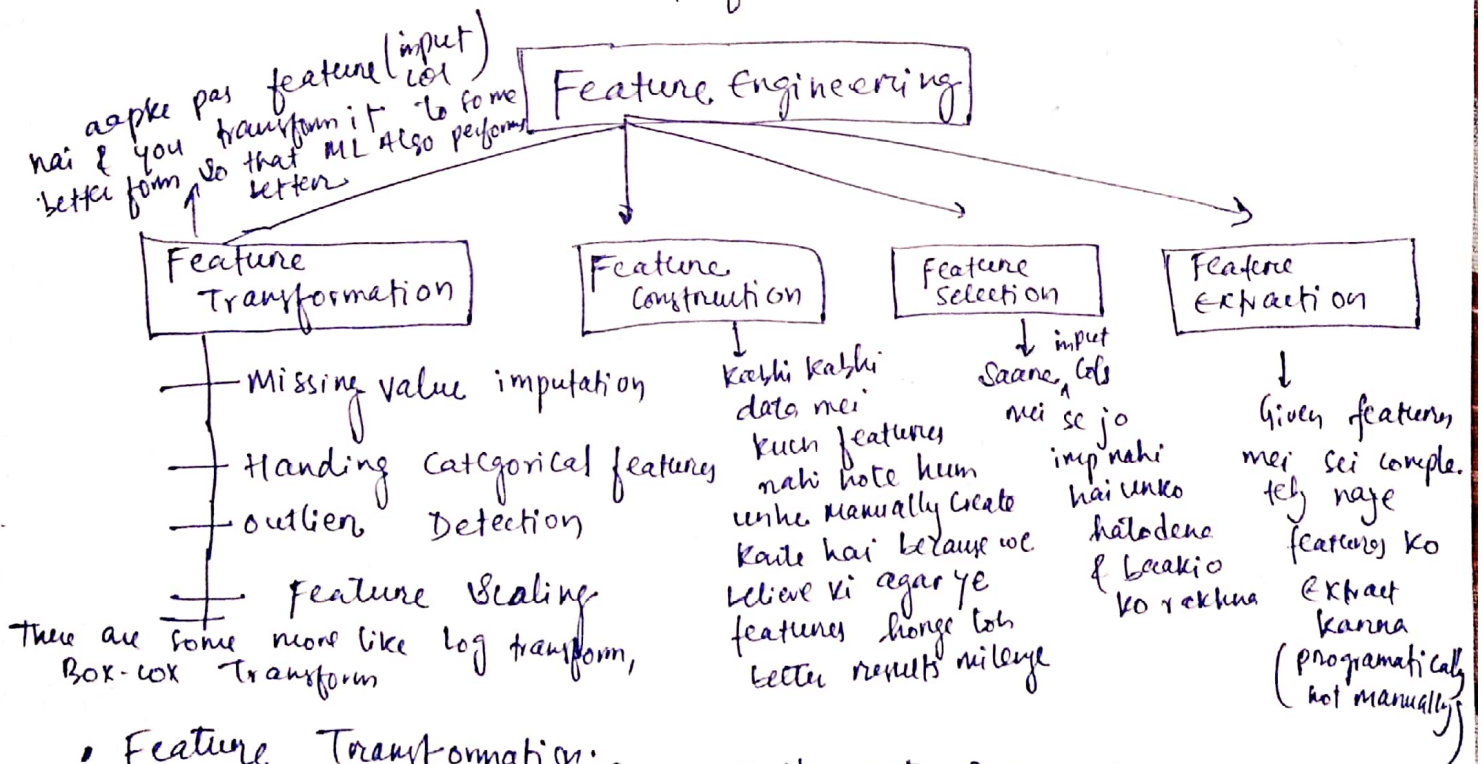
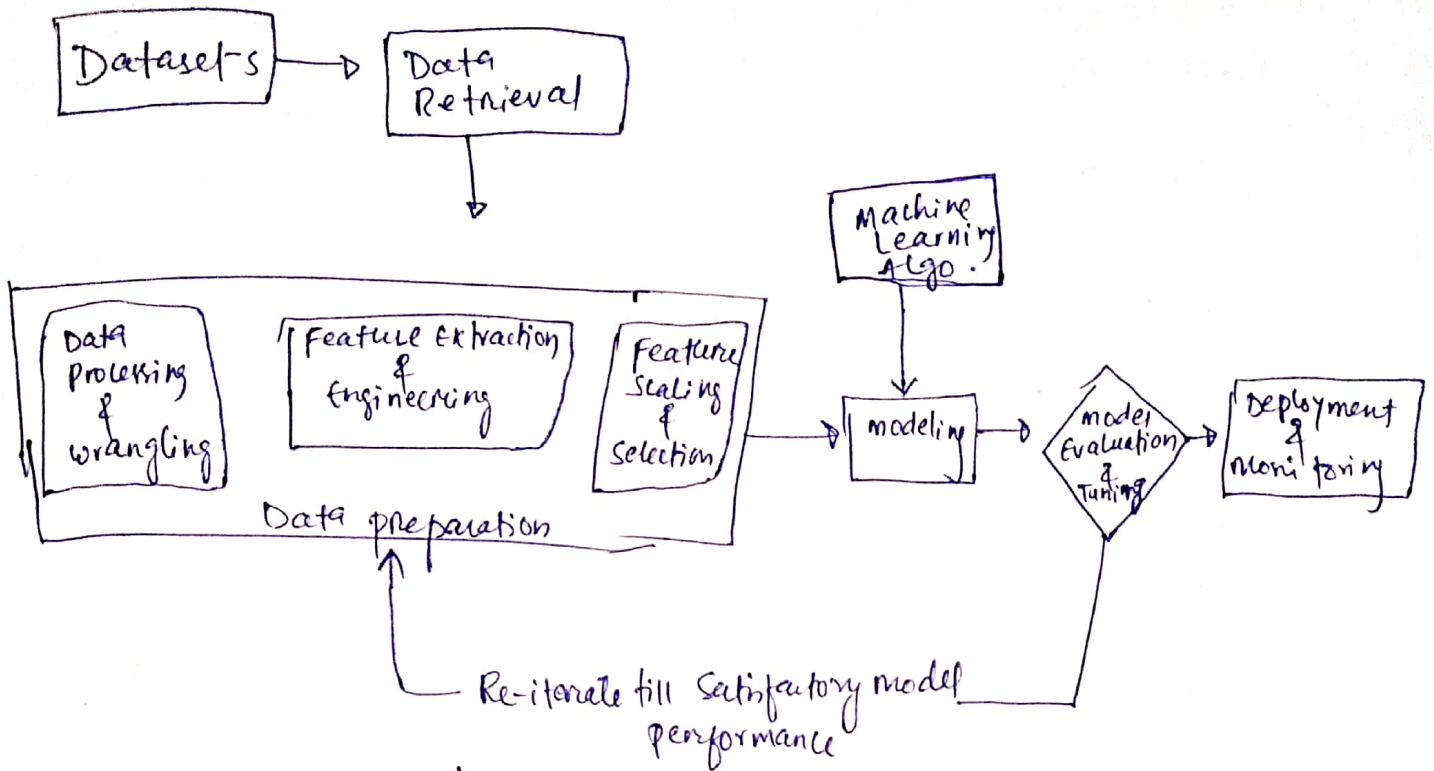


4./ Feature Engineering:- (It is an art sab apne apne tarike se karke hai).

It 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 does not accept missing values while training the model. ∴ Before Training we have to fill those missing values or remove those rows.

Av age = 26

data NAN mei 26 ban denge

• kabhi kabhi bahot kam values missing hoti hai then hum row remove kar dete hai ∴ farak nahi padega tab.

kashi kashi bahot jyada Data missing hota hai Page 18.

toh we need to fill $\rightarrow$ 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	1	0
4	Lion	4	0	0	0	0	1

$\Rightarrow$ then 10000 represents a dog... and soon.

Categorical values

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

Sometimes we convert Numerical values to categorical:-

Like:- Age (Numerical)

0-12

13-19

20-35

Child

Teenage

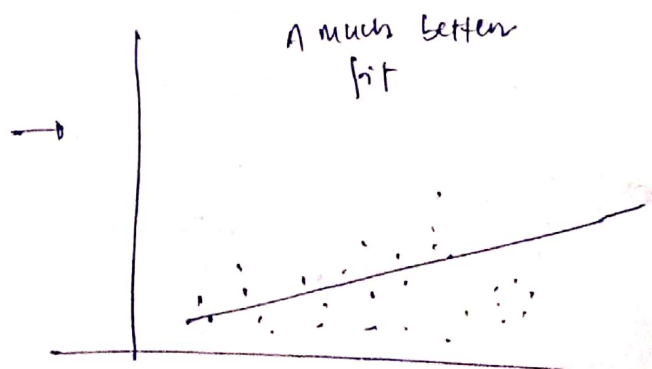
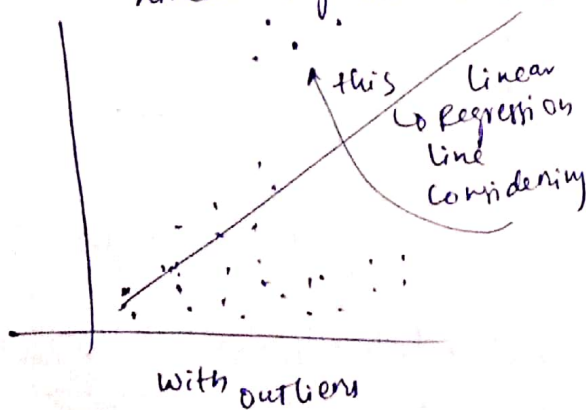
Adult

and so on.

This is called Binning.

Outlier Detection.

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



ML Algos that are highly affected by outliers
There are some outliers & hence it becomes imp to remove outlier but before that we need to detect outliers

— Feature Scaling → It is the last thing we do in the feature eng. 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 to ex given range mei leate hai -1 to 1 (min max scaling, standardization, Normalization, min absolute scaling) are some techniques

● Feature construction

completely Ek naye feature (input col) create karna (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 ~~the~~ to categorical feature

family
 0 → Alone
 0-1 → small family
 4+ → Large family

Some techniques are:
 Feature splitting
 Feature Grouping.

● Feature Selection.

Bohot saare input cols hai & ap important input cols ko select krte ho.

MNIST data set → Deep Learning (Convolutional Neural Network)
 50000 images of handwritten digits.

& Each image is 28x28 resolution = 784 pixels

most pixels are white & the black/gray pixels in digit dikhta hai
 This is converted to tabular form.

each row is an image.

	label	Pixel0	Pixel1	...	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 cols. etc.

Feature Extraction → mostly when we are dealing with high dimensional data. This is not feature extraction ∴ yaha tk completely new feature create hua.

ex: house pricing

# of rooms	# of Bathrooms	price	sq. feet area	Price
:	:	:	:	:

suppose kisi ek input ko Rohan, hi hai & Ek hi col rakhte 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 eng. pipeline. ML model ko data dena & just pehle we do feature scaling.

Standardization:-

Why do we need feature scaling → Already discussed with

age	salary
50	83000
27	47000

suppose we use KNN Algo (it uses distance)
 $(50, 83000)$ $(27, 47000)$
 $(y_2 - y_1)^2 = 1225000000$

then salary factor is dominating the KNN algo & age doesnot seem to have much effect on the Algo.

Agar humare features ka scale accha nahi hoga then hamare kuch ML algo ka logic hi aisa hai designed ki wo accha perform nahi karenge.

Types of feature scaling

standardization

Normalization

min max scaler
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	

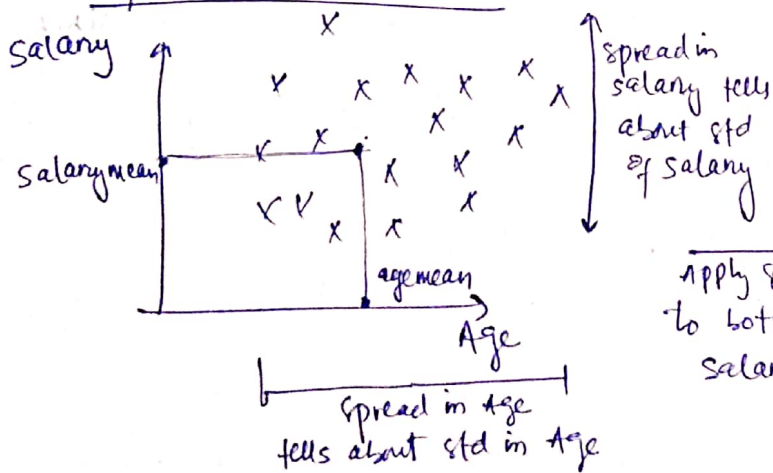
say we want to standardize Age.
then $(x) \text{ Age} \rightarrow \text{Age}'$
 $X_i \rightarrow X_i'$

$$X_i' = \frac{X_i - \bar{x}}{\sigma}$$

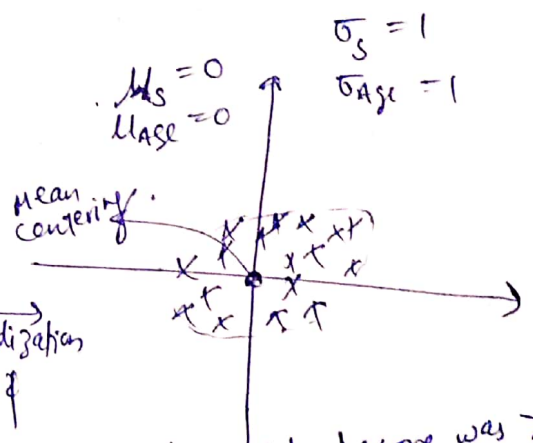
$\bar{x} \rightarrow$ mean of age col
 $\sigma \rightarrow$ std of age col

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

Geometric Intuition:



apply standardization
to both Age & salary



if std before was > 1
then the data was squeezed & if std before was < 1 then data is spread to make std = 1.

See Machine Learning on desktop
(Jupyter Notebook)

Feature Engineering $\rightarrow$ 1. Feature Scaling

$\rightarrow$ 1. standardization.

When to use Standardization?

Page 20

<u>Algorithm(s)</u>	<u>Reason of applying feature Scaling:-</u>
1. / k-means	uses the Euclidean distance Measure
2. / k-Nearest Neighbours	Measure the distances b/w pairs of samples & these distances are influenced by the measurement units
3. / Principal Component Analysis (PCA)	Tries to get the feature with the Maximum variance
4. / Artificial Neural Network	Apply Gradient Descent
5. / Gradient Descent	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 do not do standardization:-

Decision Tree, Random forest, Gradient Boost, XG Boost.
↓
inpe standardization ka koi fark 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 data $\rightarrow$ it has Numerical values + units
 & eliminate the units & keep the magnitude - that too on a
 common scale. Normalization Technique

- MinMax 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
 :
 :

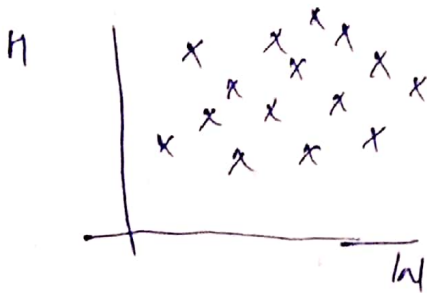
Normalization
 Min-max Scaling

$$X_i' = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \rightarrow \text{all pos}$$

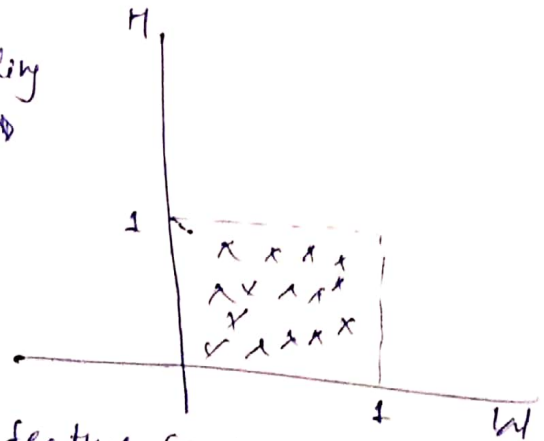
$\rightarrow$ Smallest value 0
 $\rightarrow$ Range :- [0,1]

Geometric Intuition

Weight v/s height



min max scaling



★ develop - Machine-learning - feature engineering - feature scaling - Normalization.

— Mean Normalization

wt (X_i 's)
 73
 86
 54
 31
 :
 :

Mean Normalization

$$X_i' = \frac{X_i - X_{\text{mean}}}{X_{\max} - X_{\min}} \rightarrow \text{Mean Centering.}$$

$\rightarrow$ Range -1 to 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 algos where we need centered data $\rightarrow$ But we then use standardization.

Max Absolute Scaling

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 :- $X'_i = \frac{X_i - X_{median}}{IQR}$

class in Sci-kit 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% value - 25% value

Standardization v/s Normalization

1st Ques. Is feature scaling reqd? (check kaise algo me reqd. hai & kaise me 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/data :-

Encoding Categorical Data

ML algo expect numbers & categorical data are in string format :- It's our 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)

vvv imp

Types of Catego. data

Nominal

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

Ex: ① States we cannot say UP is better than MP
② Gender we cannot say Male is better than Female

Ordinal

Here we have order

Ex: ① Distinctions (Ranks in class)
② Reviews (Excellent, Good, Bad)

Ordinal v/s Label Encoding

Suppose we have dataset



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 mentioned nominal or ordinal)
& Y is categorical → Label Encoding on Y
Ex: Email-spam classifier & X walope
Placement hegg ya nahi etc. Encoding

Ordinal Encoding

Suppose we have an input col

Categorical Data	Education	Education	Education
↓ It is ordinal	High School	0	HS → 0
	undergraduate	1	UG → 1
	under Gr --	1	Ph → 2
	post Grad --	2	
	post Grad --	2	
	High School --	0	
	post Grad --	2	
	High School	0	
	under Grad --	1	

PG > UG > HS

∴ Ph ko sabse jyada value do
UG ko use kam
& HS ko sabse kam

Jupyter Note book.

One Hot Encoding :- (for Nominal)

Consider a Nominal Categorical Data :- 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	0
Red	0	0	0	0	0
Yellow	1	1	0	0	1
Blue	0	0	1	0	0

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
1 0 0
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 mei
convert kar diya.

Q:- what if we have 50 diff categories?

A:- Then we make 50 cols.

Dummy Variable Trap (jo cols banke hai in one hot encoding and called dummy variables)

what we do is after one-hot encoding we drop the
1st col (Generally, kisi bhi col ko kar sakte hai) say 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 do not drop then $\sum each row = 1 \rightarrow$ a mathematical
repⁿ of the col
we do not want this $\therefore$ they create problems for linear
ML models (Regression models)

$\therefore$ we drop col color-Y $\therefore$ Now: Y $\rightarrow [0, 0]$ Before
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

so say there are 40 Brands $\therefore$ in one hot encoding there
will be 40 cols. (this is the way but this will make
the 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 jyada &
kuch kam saarahi hai.

Column Transformer:

Suppose we have input cols & in one of it it is Ordinal, other is Nominal & other has missing values, then we cannot take the three cols as separate parts & then combine them back. Perform the operation of then combine them back.

Ex: 100 Data points/rows

we'll have to use imputer to fill the missing values.

Age (20 values missing)	city	gender	review
----------------------------	------	--------	--------

Same col's pe aag aag transf. lagane hai hence agar aag karke kiya then 3 diff numpy arrays banenge & un teeno ko joinna padega → This is not efficient

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

Jupyter Notebook

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 preprocessing to train & test.

Ex: we have a ML model & data → if we ~~handle~~ handle missing values then handle categorical data then say we train the model



Project

yaha jo series of steps hum training ke time likhte hai wohi series of steps humme production time pe likhne hote hai (which is not good)

without Pipeline
with Pipeline.

Jupyter Notebook

Mathematical Transformations :

Feature Eng $\rightarrow$ Feature Transf. $\rightarrow$ mathematical Transformations

Simple bhasa mei:- hum apne col ke upar mathematical fn laga kar transf. kar ruke hai into new col.

- + log Transf
- + Reciprocal Transf
- + power Transf (power & sqrt. Transform)
- + Box Cox Transf (very famous)
- + Yeo-Johnson Transf.

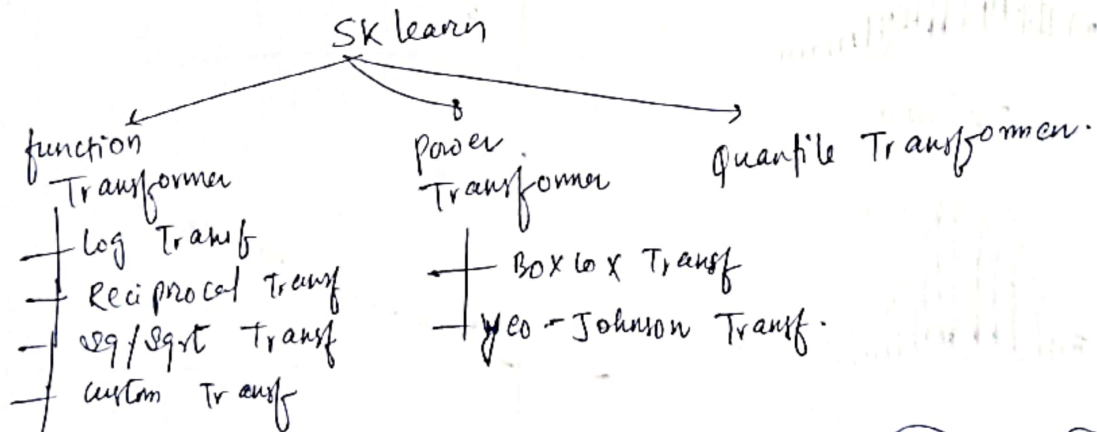
$\rightarrow$ we'll apply these transf. on columns & see ki model ka perf improve hota hai ya nahi

we can also define a custom transf like $(x^2 + 2x)$ but after that we should get Normal dist. of that col. After these transformation the pdf (distribution) becomes

Normal dist. $\rightarrow$ vimp in statistics (since we can predict a lot about the data then) (Calculations kar sakte hai)

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

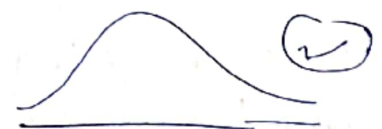
Algos like Decision Tree, Random Forest ko farak nahi padta ki data ka dist Normal hai ya nahi.



How to find if data is Normal?

Ans use $\rightarrow$ Seaborn $\rightarrow$ sns.distplot()

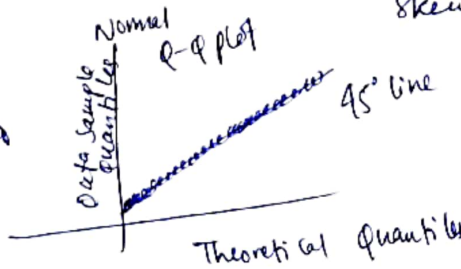
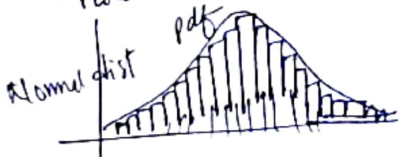
$\rightarrow$ pandas $\rightarrow$ pd.skew() $\rightarrow$ 0 $\checkmark$

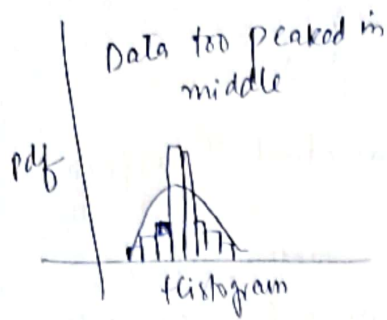


0 $\checkmark$, +ve ya +ve aage then skewed hai dist.

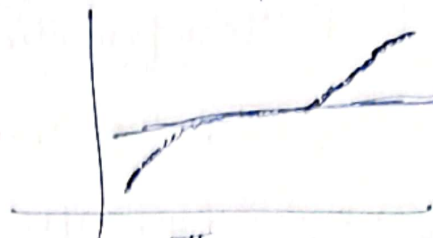
~~Ans~~ $\rightarrow$ QQ plot

How to read



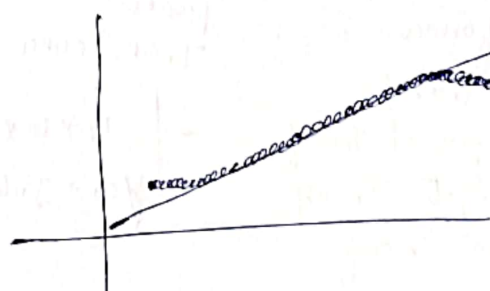
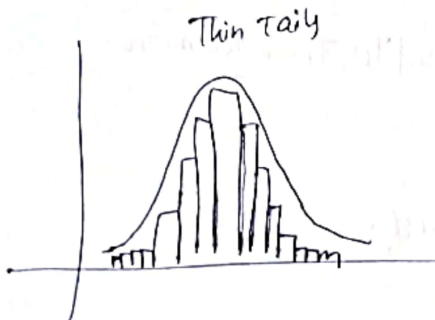
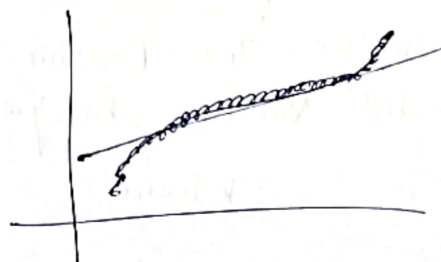
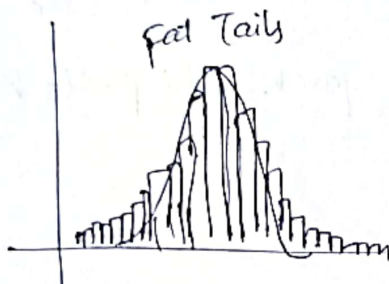
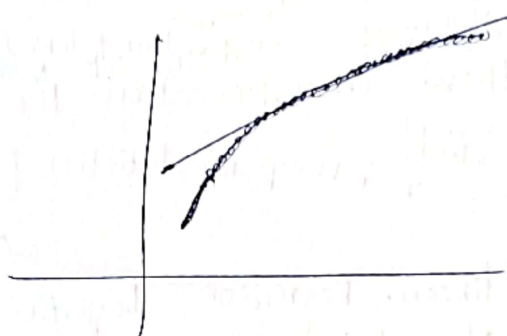
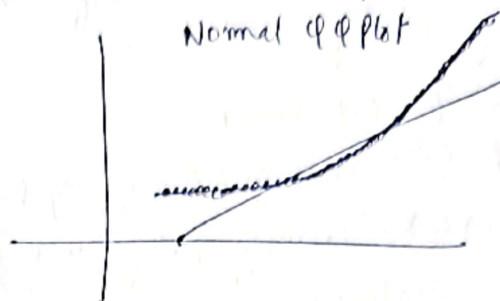


Normal Q-Q Plot



Theoretical Quantiles

Normal Q-Q Plot



Hence, in qq dist jitna aap line ke upar ho utna aap ka data jyada Normally dist hai & bhag rahi hai utna jitne data aap line ke nee ke qq plot dekh 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)

↳ isse apna wo column jo current state hai use jyada

Normally dist. ho jata hai

when to use:-

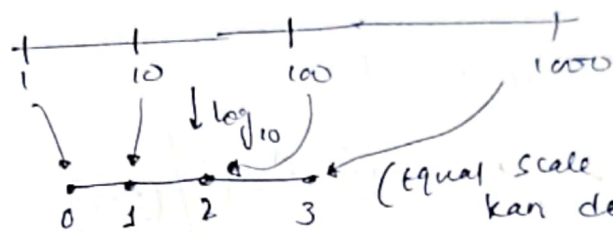
- ve values pe nahi lagega

- Right Skewed data

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

(base 10 say)

what log does



equidistant

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

Small trans big values
Big Transf small values

Square Transform (x^2)

used for left skewed data then wo jo data hai wo pehle 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 λ (lambda) that varies over the range -5 to 5 and in the process of searching, we examine all values of λ . Finally we choose the optimal value (resulting in the best approx to the normal dist) for our variable (input col/feature).

infact:- $\log$, $\sqrt{x}$, x^2 are special cases of Box-Cox Transform.

$\swarrow$ $\lambda=0$ $\downarrow$ $\lambda=1/2$ $\searrow$ $\lambda=2$

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)

This problem is solved by Yeo-Johnson Transf

Yeo-Johnson Transform:-

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

This Transformation is somewhat of an adjustment to the Box-Cox Transf, by which we can apply it to negative numbers. (is 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, etc.) 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)

$\swarrow$
 without Box-Cox or Yeo-Johnson with Box-Cox or Yeo-Johnson.

ENCODING Numerical features:-

Page 25

Ex: google playstore dataset

apps # of downloads	Bins
23	100+
101023499	1000+
9927	10000+
11	100000+
101	
39	
2759862	
13405123	

iss data
ko we can
convert to
categorical
data

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

Discretization
(Binning)

Binartization

Discretization (Binning)

Discretization is a process of transforming continuous variables into discrete variables by creating a set of continuous intervals that span the range of the variable's values. Discretization is also called binning, where 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, ...



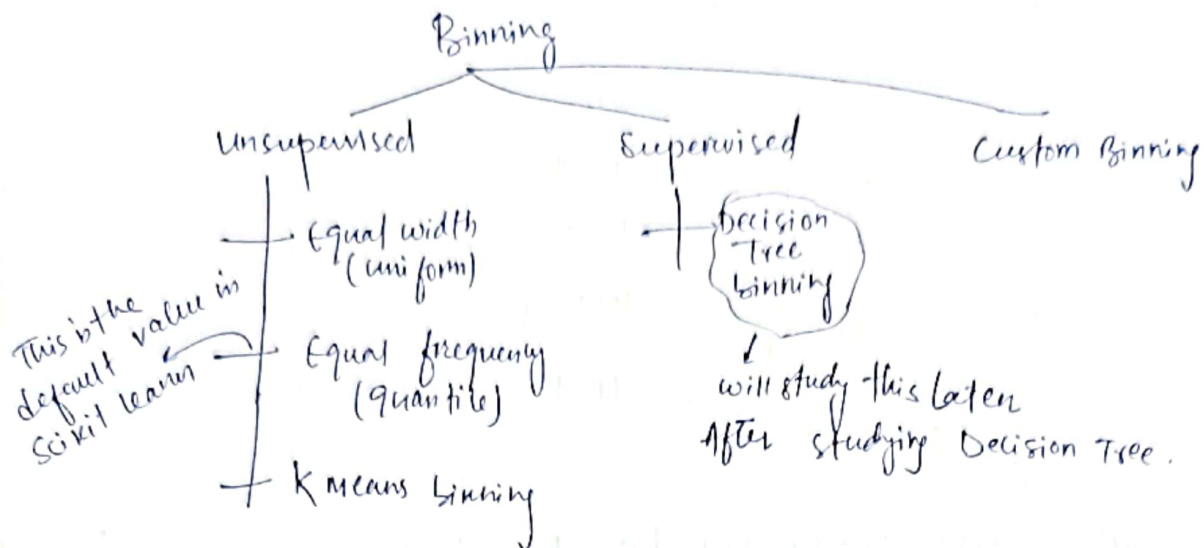
0-10, 10-20, 20-30 (Just like histograms)

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

and now put your values into these bins.

- outliers handle aajale hai.
 ↓ and then we can plot like histogram
 ho jate hai is wo last ya first bin mei
- 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 perc. - 20 perc. of data 90 perc. - 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 shi 10 log hai yaha shi 10 log hai & so on . . .

but each bins hai same # of obs./data.

Quantile binning performs better than uniform binning ^{Page 26} because:

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

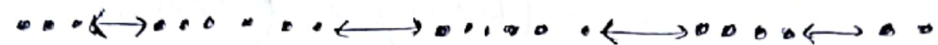
K Means Binning.

Here we use a clustering Algorithm called kMeans.

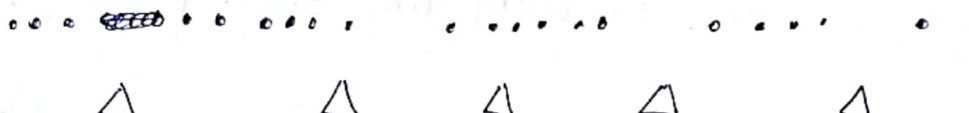
 This clusters are formed by kMeans in any dimensions.

This is used jab hamara data clusters mei ho.
 kum logo ke age phir thore dur ke baad kuch aur logo ka age ... and so on.

Here :- intervals ko Centroid bolte hai

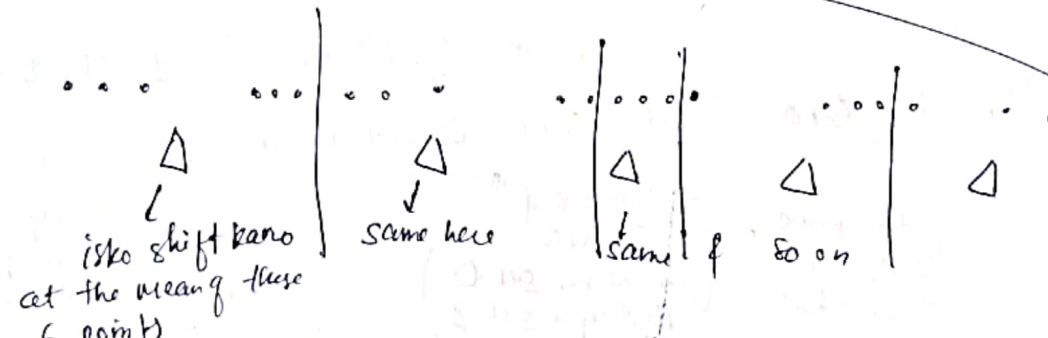
Age: 

say # of bins = 5 then 5 Arrows kahi thi randomly place kardo



then if wale point left arrow mei right wale point right arrow mei.

Now there are two techniques
 → Hando centroids/arrows ka Bisection karo
 → ^{Ek} point ka har arrow ki distance nikalo wo point jis arrow ke sabse karib ho wo point usme jayega.


 isko shift karo at the mean of these 6 points
 same here
 same & so on

after this repeat the steps from step 2 is
 And Aisa tab tak karenge jab tak improvement in shift hona band na hojai.

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

sklearn

class → K Bins Discretizer ()

bins = ?

strategy
+ uniform
+ quantile
+ K Means

encoding

+ ordinal

+ ~~nominal~~

One hot Encoding (if Nominal)

(matlab bin no. of data; konga data kis bin mei gya uska rep. kaise hogi.)

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 
Conts Discrete

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

Ex: Annual Income Binarization
income < 6 Lakhs 0 → no need to pay taxable zone
income > 6 Lakhs 1 (has to pay)

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

iska value by default '0' hota hai
- threshold parameter (iske niche sab 0 And upar sab 1)

copy
say salary col pe binarize kr rhe hai:-
if copy = True then ek nayi col create karega & if copy = False then ~~uski~~ salary col^m mei hi changes karega.

Handling Mixed variable & DateTime columns:

Handling Mixed variable:

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

Ex: In Titanic Data set:-

Cabin.

B5

C23

D41

⋮

→ These are not categories. Since they
- there will be many categories

B5 → we split this cat num

cat	num
B	5
C	23
D	41

3 5
↓ ↓
Categorical Numerical.

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

Again we split into two:-

cat	num
NAN	7
NAN	3
NAN	1
A	NAN
C	NAN
NAN	4
D	NAN

7
3
1
A
C
4
D

Jupyter Notebook.

Handling Date Time based cols:

date:-

23 Aug 2012

Date (23) month year days of week quarter of year which semester

ye date weekend hai ya nahi etc.
ya ye date weekdays hai

time:-

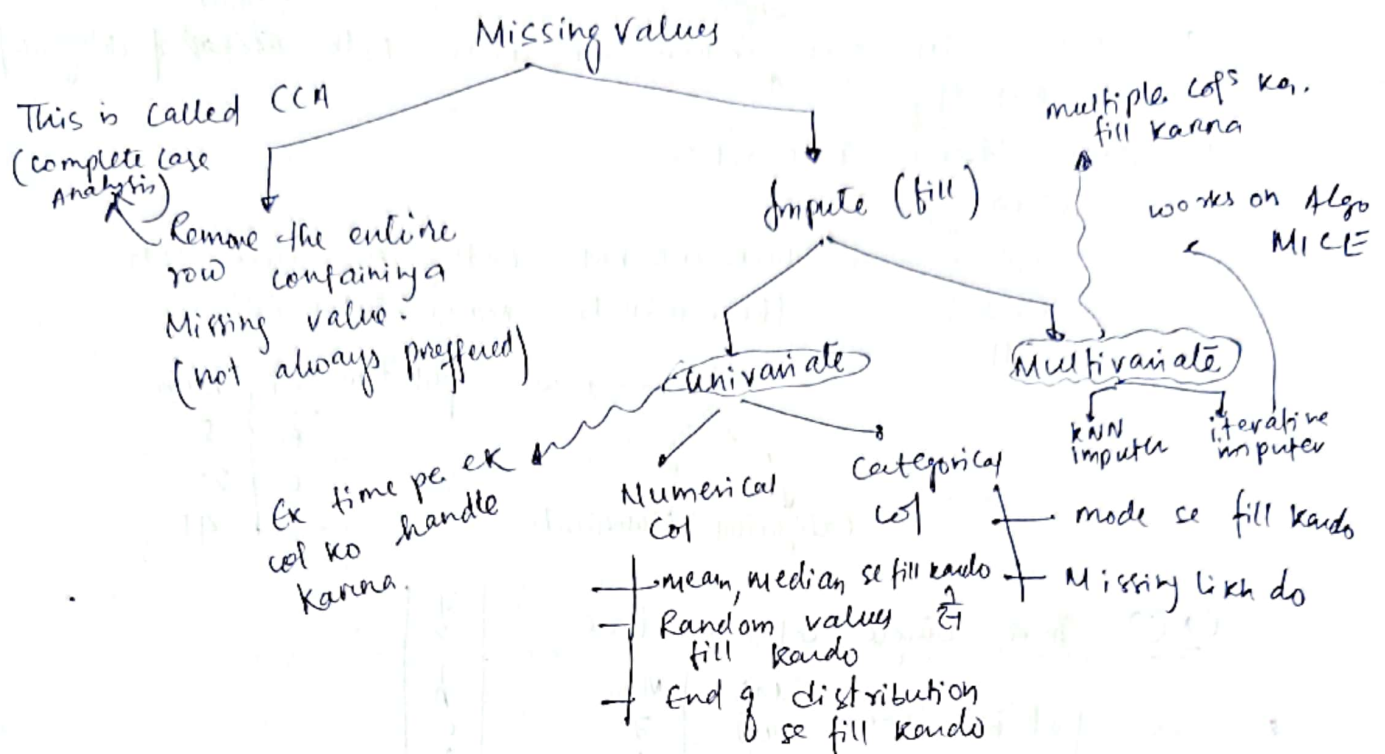
08 : 45 : 03

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 suru/ke so values missing hai. (tab hum CCA Apply nahi kar sakte)

Adv/Disadv. of CCA

Page 28

Adv.

- 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 NotebookImputing (filling) missing values.Handling Missing Numerical DataImputing Numerical Dataunivariate
imputation

say col2 mei kuch values missing hai & unko fill karne ke liye agar hum bas col2 ke baaki data 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 rahi hai then it is called multivariate imputation

UNIVARIATE IMPUTATION

missing values ke jagah

- + mean/median se 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?

Ans -> Jupyter Notebook

Mean/Median se 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

When to use:-

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

Disadv.

- changes the dist shape
- outliers aajate hai.
- Jo col mei impute krta uska relationship with other cols mei changes aajate hai.
This is called changes in covariance/correlation.

All these should not happen.

Jupyter Notebook.

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

$\rightarrow$ mostly this technique is used for categorical variables

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

$\rightarrow$ 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 model)
idea is :- we are creating a difference b/w the existing data and the missing values.

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 doesn't work good on Decision Tree based Algos.

- However the covariance of the col with other cols gets disturbed.

- This technique is 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 koi ek random value select karke 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 $\rightarrow$ True & False
 $\downarrow$ for missing value $\downarrow$ for value not missing.

Age	Fare
21	23
32	10
71	9
NaN	78
38	100

Age-NA

F
F
F
T
F

$\rightarrow$ isse ML model missing & nonmissing data mei diff create karna pata chahiye and hence uska perf improve ho jata hai.

How can we automatically select the ^{best} parameter values Page 30.
for any imputation Technique?

Here we use a technique called Grid Search CV.
↳ yaha scikit-learn kaare combinations try krte hai
and best result de deta hai.

Jupyter Note Book.

Handling Missing Categorical Data

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

Most frequent technique

* MCAR (Missing Completely at Random) → Replace with
and missing data $< 5\%$ mode of the
and one more imp thing col data.
is the mode should appear
in the data much more than
the remaining categories.
jo sabse jyada bar
appear hua hai.

- adv:- Easy to use

- disadv:- Data dist is changed.

Missing Category Imputation

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

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

Jupyter
Notebook.

↓
in numerical data the same technique was
called Arbitrary imputation technique.

MULTIVARIATE IMPUTATION TECHNIQUE

KNN imputer Iterative Imputer

KNN Imputer

Lo q works on algo called KNN (apne neighbours ki tarah)
 Lagana kisi row mei koi value missing hai toh wahs us row ki value fill hoga jo sabse jyada similar hai row to be filled se.

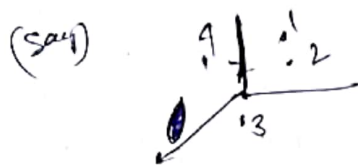
Similarity is calculated using ^{NAN} Euclidean distance se.

Ex.	S-NO	Feature1	Feature2	Feature3	Feature4
	1	33	-	67	21
	2	-	45	68	12
	3	23	51	71	18
	4	40	-	81	-
	5	35	60	79	-

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

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

This can be represented in a 2-D coord system



this we do if we take

$k=1$

say row 2 mei fill karna chahate hai then we find dist (using this) of 2 with all other pts jo shi row/point. 2 ke sabse jyada pass/close hoga uske feature 1 ke value se replace kar diya jaiga.

slly. agar q fill karne then $\text{dist}(4, y) \rightarrow$ jo sabse close uske feature 2 ki qth row mei fill karenge.

In KNN, $k \rightarrow$ # of neighbours we want to start with
 say $k=1$ then sabse closest neighbour ka feature 1 ka value ki replace karenge row 2 mei

if $k=2$ then two nearest neighbours from point/row 2 ke feature 1 ka mean ki fill karenge in row 2 and so on.

Calculating NaN distance.

we want to fill this

F1	F2	F3	F4	Sl.No
33	-	67	21	1
45	68	12	-	2
51	71	18	-	3

then leaving feature 1 we take F2, F3, F4 for distance

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

total pairs = 3
pairs possible = 2

row 2

row 3

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

of total pairs = 3
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 4th pt./row is closest to point 2.

feature 1 or row 4 i.e. 40 will be filled at row 2

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

if $k=2$.

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

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 missing values are calculated based on training data. (Hence memory usage is more)

Jupyter Notebook

There is a parameter called weights = uniform or Distance in kNN imputer class (Default)

uniform is the normal what we calculate above to fill.

if weight = distance

1	9	9	9	1
5	4	4	4	1
5	4	4	4	1

to fill this

$k=2$ say nearest neighbours are 1 & 4

then value to be filled at row 3 =

$$\frac{\frac{1}{10} \times 70 + \frac{1}{50} \times 50}{2} = 9$$

isse jo jyada close hai uska weightage jyada hote hai.

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 kar sakte hai

Adv: • It is quite accurate

Disadv: • It is slow bcoz we are using an ML Algo to predict a missing value

- Training dataset ko server pe rakna hota hai (memory heavy)

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

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

Target Col.
Profit
Numerical
∴ Regression Problem.

2. Removing the Target Col.

RDS	Ad	MS
8	15	30
4	5	20
15	10	41
12	15	26
2	16	3

3/ Introducing some fake nan values.

RDS	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.

(Learn)

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 relⁿ b/w the col so other cols cannot be used to fill the missing data in a col.

Predicting the Missing values using MICE Algo

Page 32.

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

R&D	Ad	MS
8	15	30
9.25	5	20
15	10	41
12	11.25	26
2	15	29.25

Mean of R&D: 9.25
Mean of Ad: 11.25
Mean of MS: 29.25

Generation 0

Step 2. Remove all col1 missing values.

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

Y-train: [8, 15, 12, 2]
X-train: [15, 10, 11.25, 15]
X-test: [5, 15]

Step 3. predict the Missing values of col1 using other cols

How -> using Any ML Algo (Regression, Decision tree)

Here: over input training data is. with R&D as target col & test data

we provide the input 5 20 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 col2 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
8	30
23	20
15	41
02	29.5

& then provide input

R&D	MS
12	26

& predict the NaN output of Ad.

Step 5. predict the missing values of col2 using other cols say we got 11.06

Step 6. continue for col3.

After that say we have:

<u>R2D</u>	<u>Ad</u>	<u>MS</u>	→ generation 1
8	15	30	
23.14	5	20	
15	10	41	
12	11.06	26	
2	15	31.56	

Diff b/w iteration 1 & iteration 0	<u>R2D</u>	<u>Ad</u>	<u>MS</u>	→ the diff should be 0 or almost 0.
Generation 1 - Generation 0	0	0	0	
	13.89	0	0	
	0	0	0	
	0	-0.19	0	
	0	0	2.31	

So again we repeat the process:-

By taking ~~iteration 0~~ iteration 1 as Base Table:

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

& get iteration 2 table & calculate

Say we got iteration 2

<u>R2D</u>	<u>Ad</u>	<u>MS</u>	12-11 difference
8	15	30	
23.78	5	20	
15	10	41	
12	11.22	26	
2	15	31.56	

<u>R2D</u>	<u>Ad</u>	<u>MS</u>
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 (Sharmaji ka beta)

↳ Ek aisa data point jo baaki data se alag behane karata hai aur ML Algos ko distract kar sakte hai.

Suppose ek class hai:- we are finding the ~~mean~~ ^{salary} of that

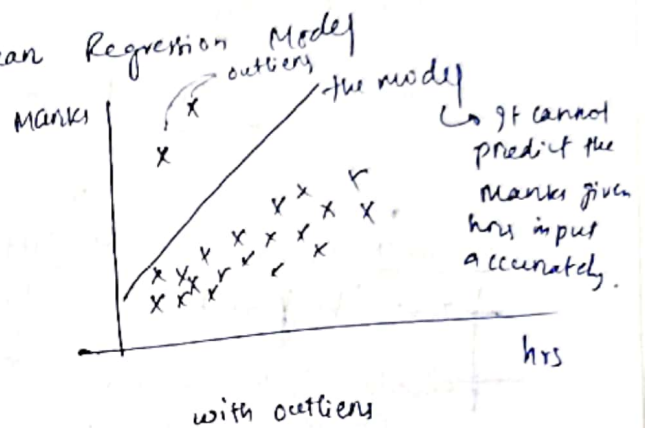
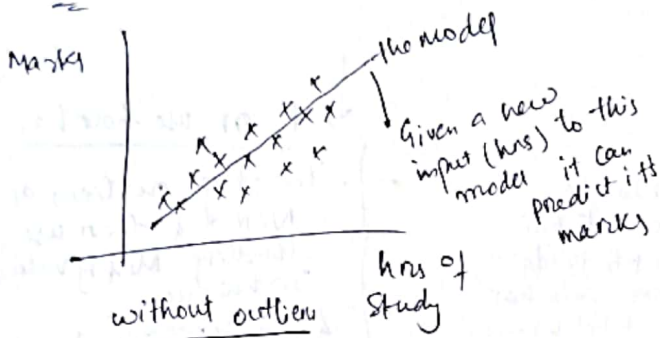
class → say ₹75000 per month

Ab ussi class mei ek outlier aagya (Bill Gates) then

Now the Mean salary will be 50 crores per month

Now this new mean is 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 se 300 de diya → this is an outlier → this has to be removed.

But sometimes outliers are imp (for Anomaly detection Mechanism) Hence outliers must kept in the Data.

Thus: outliers rakna 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 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 algos will decrease

- Linear Reg
- Logistic Reg
- Adaboost

in these algos we calculate weights

outliers hampers weight-based ML Algos

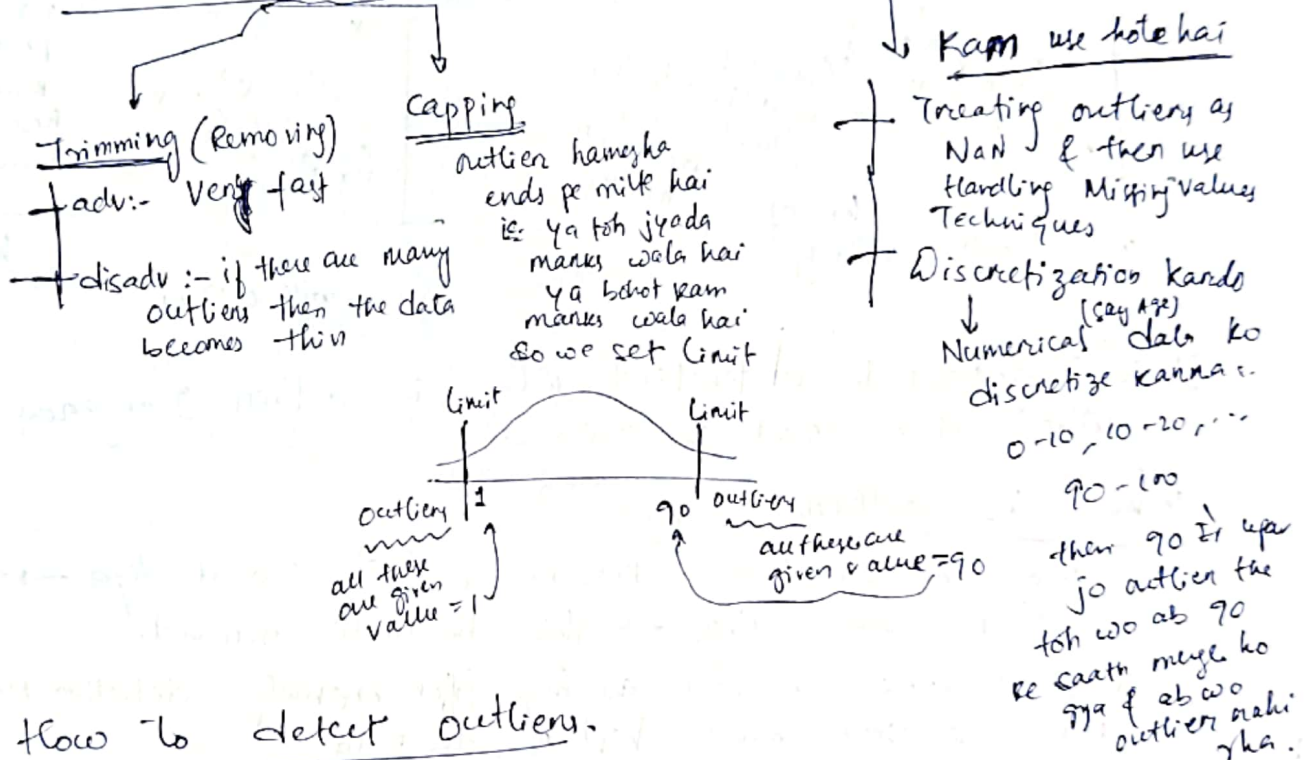
Even in Deep Learning Algos.

Tree Based Algos

Decision Tree
Random Forest
Gradient Boosting
XG Boosting

inpe outliers ka utna farak nahi padta.

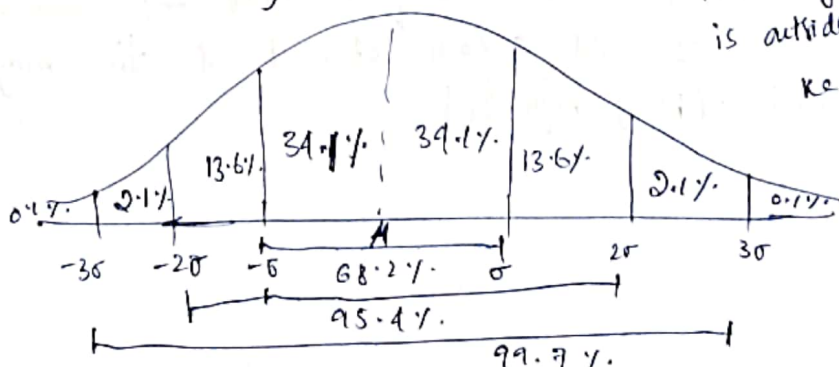
How to treat outliers?



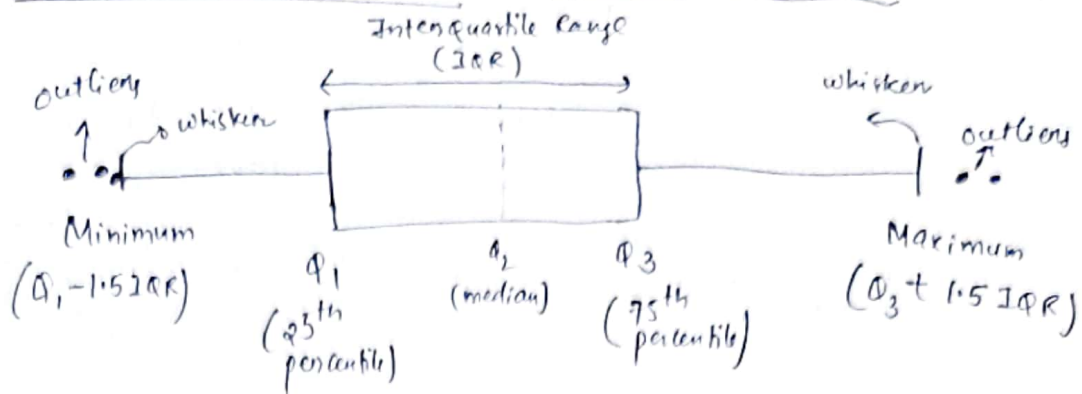
How to detect outliers.

Three imp methods.

- Normal dist. (if the col with outlier is Normally dist)
- (Z-score treatment)
- ya close to → Hence again apna observation is outside ($\pm 3\sigma$, $\pm 3\sigma$) ke bahar hai then we can treat it as outlier.

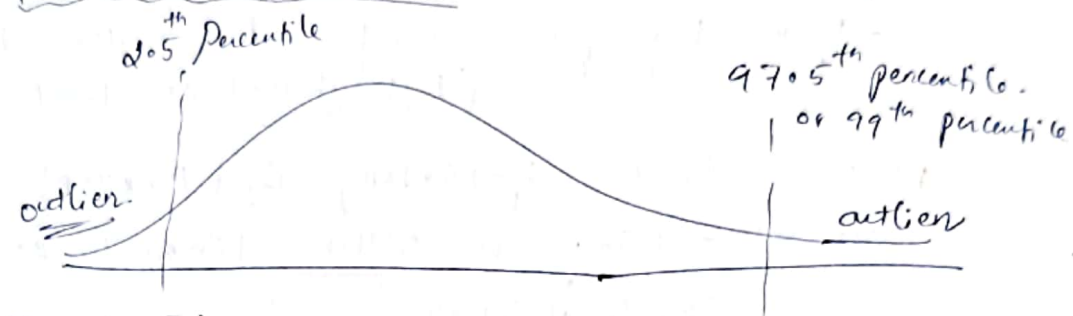


2/ Agar Data is skewed (Left ya Right skewed) Page 34
 Then we use Box Plot (IQR Based filtering)



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. (or almost Normally dist.)
 ↳ outside $(\mu - 3\sigma, \mu + 3\sigma)$ the value is an outlier

Age

27	Z-score	$x_i' = \frac{x_i - \mu}{\sigma}$	Say $x_i' = 3.5$
32			

$\Rightarrow x_i = 3.5 \sigma + \mu$

$\therefore$ if $x_i' \rightarrow (-3, 3)$

the data (x_i) is ok

$\therefore x_i$ is an outlier.

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

corresponding.

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 $x_i = -3 \text{ to } -4$

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

then $x_i = 3 \text{ to } 4$

Hence they are no more outliers,

Jupyter Notebook.

(IQR Proximity Rule)

* Outlier Detection and Removal using IQR

↳ This technique is used when the data dist 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

take upon and note the values so we declare outlier. after that we do

Trimming

Capping

(In this technique Capping is called Winsorization).

Jupyter notebook.

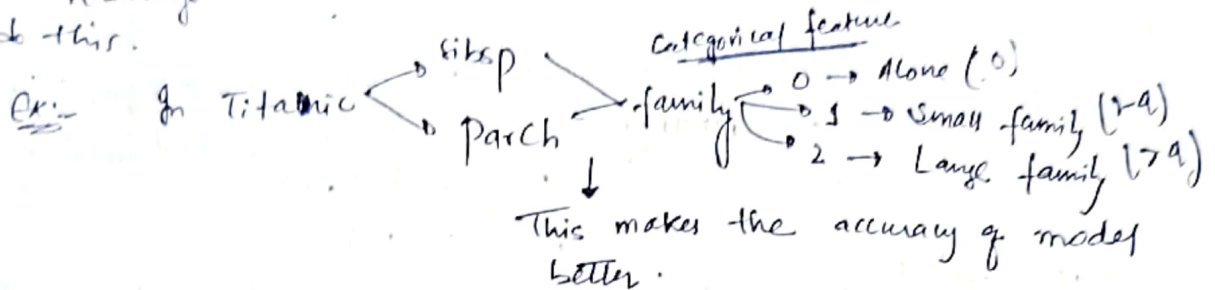
Feature

↳ Given features are Manually new

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

features create karke hai.

+ feature splitting.



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

Ex: in Titanic data set: Name Salutation Name
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 hatane pe kya fayda hola 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
 - 2) Text data
- } In this cases selecting optimum features becomes very useful.

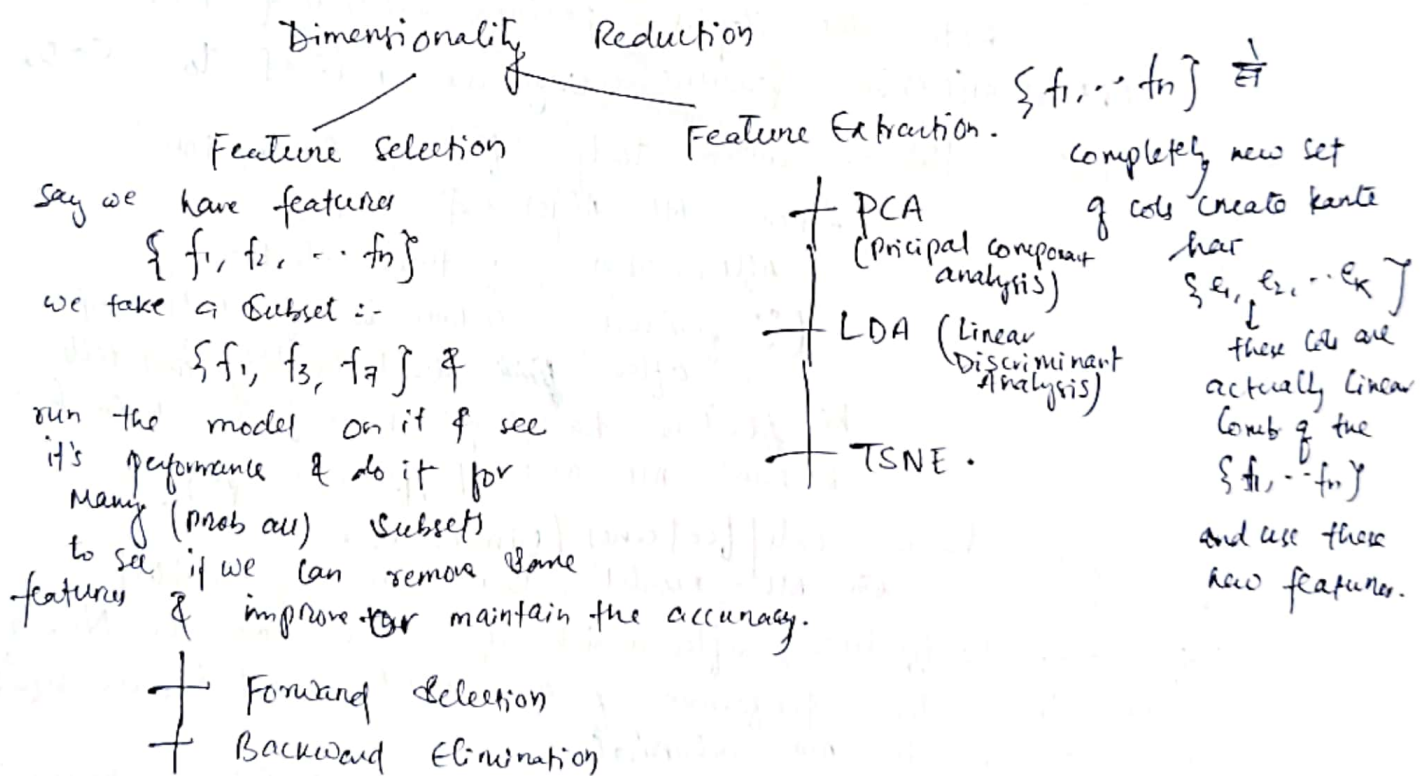
The problem with higher dimensions is: sparsity
 n cols $\rightarrow$ n -dimensional ($n \rightarrow$ very large, 1000, 2000, 3000 like)
Each row is a coordinate in the n -dim
& if we plot them the points are very sparse (far) from each other $\rightarrow$ Analogy:- wallet lost on road (1D - 1D)
playground (2D)
Building (3D)

then finding wallet on 1D is easier. (since wallet to keno case mei same hi hai).

* Many ML Algos are based on Statistical calculations like distances, 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:-
(i) Perf ↓
(ii) Computation Increases.



PCA (Principal Component Analysis)

- It is an unsupervised ML problem (we have input & no output)
- It is a useful, old, reliable technique & its underlying maths is quite complex.
- 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 lekar 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/dimen. we cannot visualise it in 10D, PCA reduces it to 3D or 2D & we are now able to visualise it.)

Geometric Intuition Behind PCA

Ex: House Price prediction

No. of rooms	No. of grocery shops around house	Price (in lakhs)
3	2	60
4	0	130
5	6	170
2	10	90

* 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 rakha hai then we will keep the one jisse output jyada depend kar raha hai is No. of rooms and remove the other one.

This we did :- he had some domain knowledge But what if we don't have domain knowledge then we use mathematics to find input col on which the output depends the most :-

Plot:-

of grocery shops



Rooms. Vahi vary but grocery shops bahot vary kar rahi hai nahi kar rahi.

→ 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

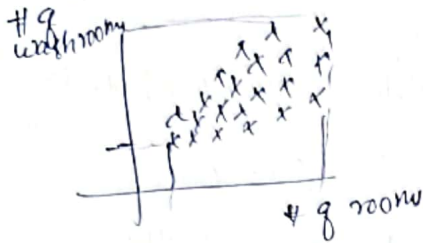
Now Suppose we have changes in Data

# of room	# of washrooms	Price
:	:	:

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

Now both room & washroom pe depend karega price hence yaha feature selection will not work (variance wash trick).

in this case if we Scatter Plot



→ 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

# of room	# of washrooms	Price
:	:	:

→ Even if Real estate Broker ko bole ki konga col imp hai, kisi ek ko hi select karo → he will not be able to select it → other way round is:-

Collect your data in a different way
more the # of rooms + # of washrooms more is area
& Now we have only one col.

size (area)	Price
:	:

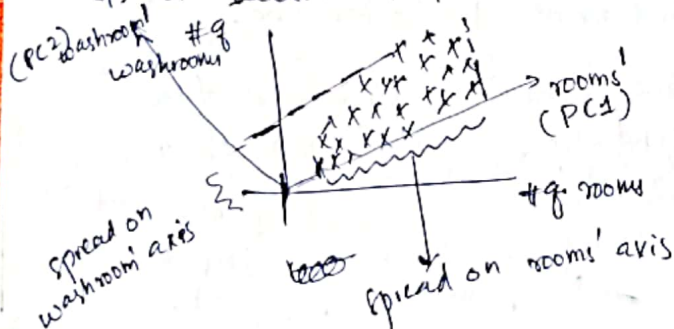
since

↓ 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 bat diya o he had Domain Knowledge:-

As a Data Scientist how do we do it:-



→ yaha feature selection technique fail kar gya tha fine variance dono axis pe same aaveha 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 rakhega & washrooms' (PC2) ko hata denge.
→ ∵ variance of data on PC1 is more.

Now we transform the Data according to the PC1

Note:-

No. of Principal components = # of original cols/features/dimensions

* Why is Variance Important?
for PCA

What Variance Represents

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

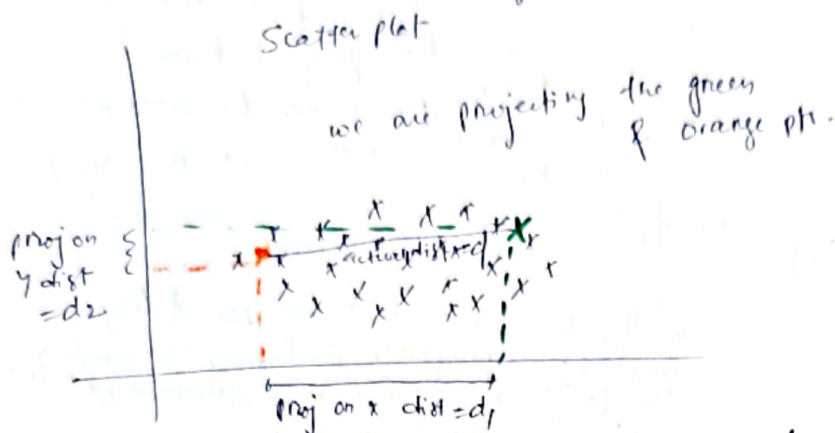
If we have Dataset:

then $var = \sigma^2 =$

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

Note: spread exactly variance. nahi hota. But spread of Variance.

* Mean abs. deviation (MAD) is not used to represent spread in PCA since MAD is not differentiable -- so use of optimization model then is not possible.



Now X axis data has more spread hence for PCA we choose the X axis data why?

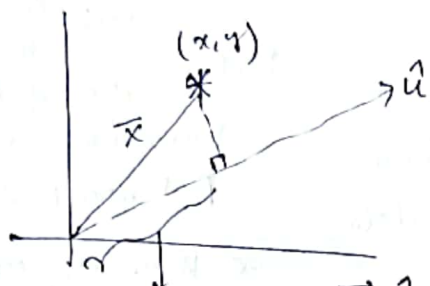
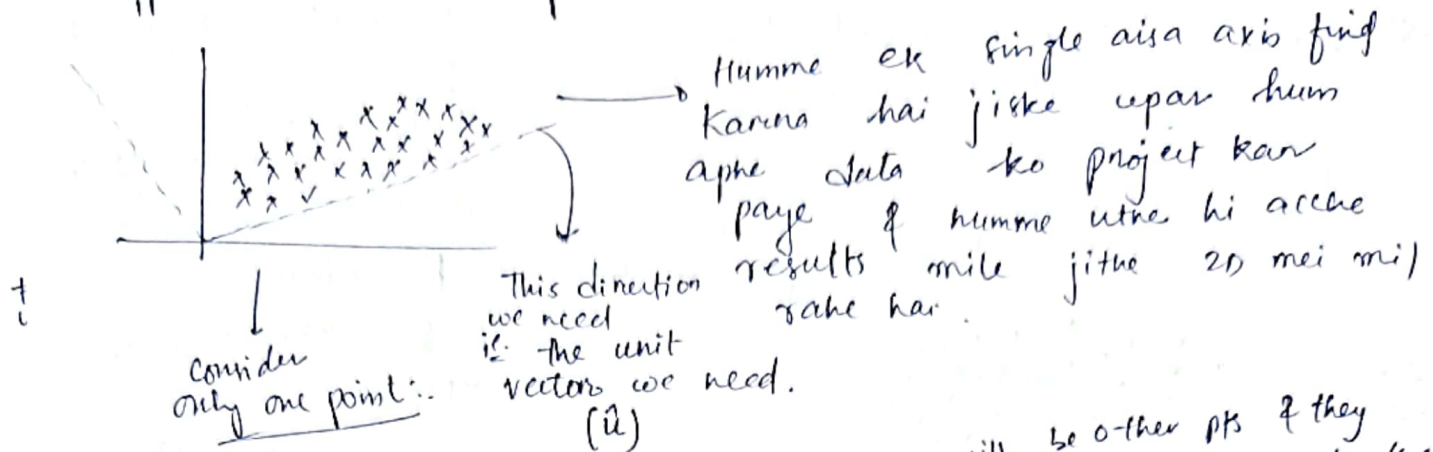
∴ when we choose X-axis data then the ~~actual~~ distance b/w orange & green is d_1 which is comp to d (actual dist) hence ML algo will not be affected (based on distance) but suppose we take the less spread data (Y-axis data) as input & removed the X-axis data then the dist b/w orange & green pt is d_2 which is nowhere near the actual distance d 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/feature with more spread/variance as input.

Hence jab hum higher dim & lower dim mei leke aate hai toh data ke apas ke distance/behaviour disturb na ho, isiliye hum variance ko maximize karte hai.

Problem Formulation in PCA

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



→ sly there will be other pts & they will be projected (which is the length of the projection of $\hat{u}$). Now we need to maximize the variance.

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

Hence: we need to choose that unit vector $\hat{u}$ such that the variance is maximised.

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

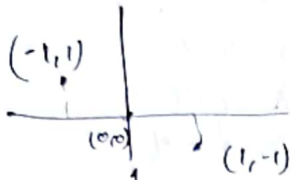
Now PCA finds that $\hat{u}$ corresponding to which these variance is maximised

Now this becomes an optimization problem.

This is called mathematical objective fun.

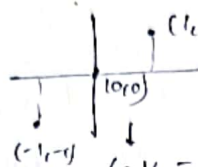
To understand the solⁿ of the Opt. problem, Page 38
 we need to understand the Covariance & Covariance Matrix:-
Covariance & Covariance Matrix

- Variance is a metric of a single axis, a single co/feature.
- Covariance is a metric of two axes, two co/features.
 ↳ it gives the relationship b/w x & y axes.



$$\text{Cov} = \frac{(-1 \times 1) + (0 \times 0) + (1 \times -1)}{3} = -2/3$$

↳ -ve means as x ↑ y ↓.



$$\text{Cov} = \frac{(-1 \times -1) + (0 \times 0) + (1 \times 1)}{3}$$

= 2/3 → the means as x ↑ y ↑

↳ cov is similar to corr, except corr is restricted to [-1, 1] but cov has no restrictions.

Covariance Matrix:-

say we have:-

$$\begin{array}{c|c} x_1 & x_2 \\ \hline \vdots & \vdots \end{array}$$

then Cov Matrix → will be 2x2 matrix

$$\begin{array}{c} x_1 \\ x_2 \end{array} \begin{bmatrix} \text{Cov}(x_1, x_1) & \text{Cov}(x_1, x_2) \\ \text{Cov}(x_2, x_1) & \text{Cov}(x_2, x_2) \end{bmatrix} = \begin{bmatrix} \text{Var}(x_1) & \text{Cov}(x_1, x_2) \\ \text{Cov}(x_2, x_1) & \text{Var}(x_2) \end{bmatrix}$$

slly. if $\begin{array}{c|c|c} x_1 & x_2 & x_3 \\ \hline \vdots & \vdots & \vdots \end{array}$

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

$$\begin{aligned} \because \text{Cov}(a, a) &= \text{Var}(a) \\ \& \text{Cov}(a, b) &= \text{Cov}(b, a) \end{aligned}$$

→ This Matrix is important since it has all the variances $\text{Var}(x_1), \text{Var}(x_2), \text{Var}(x_3)$ that tell the spread of data on each axis. all the pairs of axes by

And also tells the relation b/w cov. Cov tell the orientation of the Data.

Eigen Decomposition of Covariance Matrix:-

Reviser: Eigen vectors & Eigen values. (matrices as linear transformation)

→ for this Read inside 100 days of ML

Lecture 47 PCA part 2

Link in Descrip. vision.dummy.com.

The covariance matrix defines both the spread (variance) & the orientation (Covariance) of our data. $\Rightarrow$ 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. $\rightarrow$ means jiska largest spread Eigen value hoga.

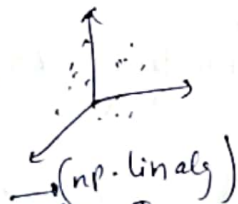
★ In other words, the largest Eigen vector of the covariance matrix always points into the direction of the largest variance of the data, & the magnitude of this vector equals to the corresponding Eigen value.

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 sor:-

input Dataset:-

$f_1 \mid f_2 \mid f_3$
:
:
:



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

& jis Eigen vector ka Eigen Value sabse jyada hai uski PC1 kanta 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 PC1 $\rightarrow$ 1D.

Step 1:- Mean centering of Data

(not a mandatory step)

but PCA improves if done properly.

Step 2:- find cov Matrix. $\rightarrow$ (np.cov using numpy)

$$\begin{matrix} f_1 \\ f_2 \\ f_3 \end{matrix} \begin{bmatrix} V_{f_1} & C_{f_1 f_2} & C_{f_1 f_3} \\ C_{f_1 f_2} & V_{f_2} & C_{f_2 f_3} \\ C_{f_1 f_3} & C_{f_2 f_3} & V_{f_3} \end{bmatrix}$$

Now after getting the Principal Components then Page 39
 how do we transform the points. $\begin{matrix} f_1 & f_2 & f_3 \\ \vdots & \vdots & \vdots \end{matrix} \rightarrow \begin{matrix} PC_1 & PC_2 \end{matrix}$
 i.e. originally in 3D $\rightarrow$ 2D if we choose PC_1, PC_2
 " $\rightarrow$ 1D if we choose PC_1 .

3D $\rightarrow$ 1D. Take the projections of the points on PC_1 i.e.
 dot product of pts with PC_1 .
 the shape of pts is $(1000, 3)$ matrix
 & the PC_1 is $(1, 3)$ matrix
 $\therefore$ dot product gives $(1000, 1)$ matrix
 which will be our PC_1
 $\vdots$

3D $\rightarrow$ 2D. PC_1 & PC_2 (plane)
 projection on the PC_1, PC_2 plane.
 again shape of datapoints is $(1000, 3)$ & shape of PC_1, PC_2 plane
 is $(2, 3) \rightarrow$ take dot product i.e. $(1000, 3) \cdot (2, 3)^T$
 $= (1000, 2) \rightarrow \begin{matrix} PC_1 & PC_2 \end{matrix}$ $\rightarrow$ this is U^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 reqd.)
 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 $\therefore$ The dataset has
 42000 rows and 784 cols. and 1 label col.
 the label col indicates which digit is there in the image

finding the optimum no. of principal components:-

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

then the eigen values of the covariance matrix

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

this denotes
the spread/variance of
data on PC_1 .

This explains the spread of data on PC_2 .
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 is explained
by λ_1

and so on & hence total
90% variance explain
karuxena hola hai

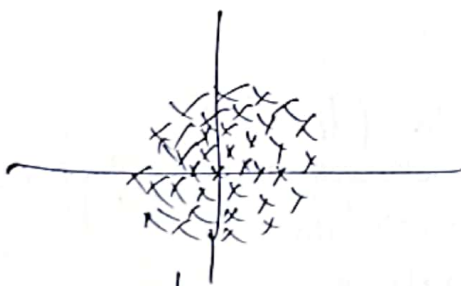
hence humme utne values (λ_i 's) lena hai jitne meri
joste joste hum 90% pohuch jaye.

Say $(\lambda_1 \% + \dots + \lambda_{15} \%) = 90\%$. Then we take the

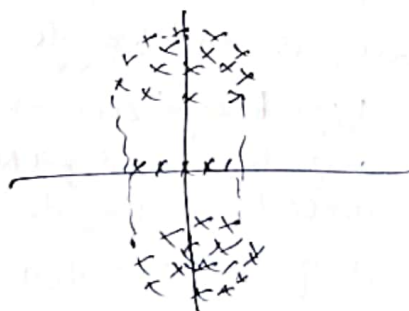
$PC_1 \dots PC_{15} \rightarrow$ ie 15 principal components are the
Optimum components for good/best accuracy.

guyfex Notebook (kaggle).

When PCA Doesnot 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
ke toh wo ek dusre
pe overlap karjainge &
PCA do points ko diff.
hi nahi kar payega.



↓ data has some pattern
iska projection $\hat{A}$
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)

LR

Simple Linear Regression

only one input col & one output col.

Ex:- Consider the data :-

input	cgpa	package	output
	6.66	3001	
	:	:	

Multiple Linear Regression

more than 1 input col

Ex:-

input	cgpa	Gender	state	IQ	package
	:	:	:	:	:

polynomial Linear Regression

↳ This helps when data is not linear

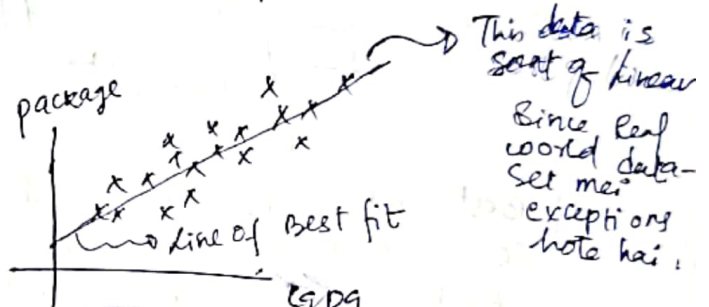
Regularisation

• Simple Linear Regression

input	cgpa	package	output
	7.1	305	
	4.7	102	
	8.9	402	
	8.1	309	

plot this data

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



Line eqn. $y = mx + b$ (m = slope, b = y-intercept)

package = $m \times \text{cgpa} + b$

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

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

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

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

this denotes
the spread/variance of
data on PC1.

This explains the spread of data on PC2.
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 & hence total
90% variance explain
karwana hola hai

hence humne uthe values (λ_i 's) leke hai jithe mei
jote jote hum 90% pohuch jaye.

Say $(\lambda_1 \% + \dots + \lambda_{15} \%) = 90\%$. Then we take the

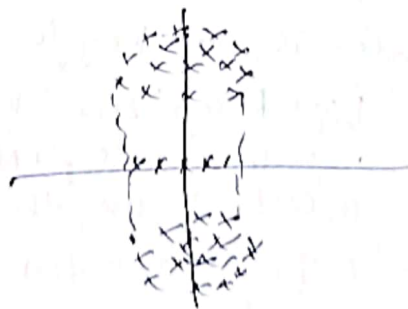
PC1 $\dots$ PC15 $\rightarrow$ i.e. 15 principal components are the
Optimum components for good/best accuracy.

Jupyter Notebook (kaggle).

When PCA Doesnot 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 remain the
same in the new coord. system
hence PCA will not work.



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



Data has some pattern
iska projection to
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)

LR

Simple Linear Regression

only one input col & one output col.

Ex: Consider the data :-

input	cgpa	package	output
	6.66	3.01	
	⋮	⋮	

Multiple Linear Regression

more than 1 input col

Ex:-	cgpa	Gender	cfato	IQ	package
	⋮	⋮	⋮	⋮	⋮

Polynomial Linear Regression

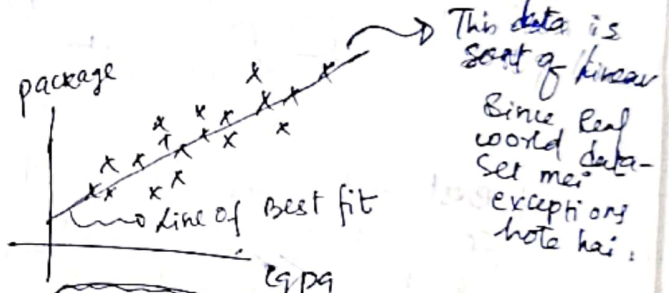
↳ This helps when data is not linear

Regularisation

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



Line eqn. $y = mx + b$ (m = slope, b = y-intercept)

Package = $m \times \text{cgpa} + b$

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

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

Jupyter Notebook

Mathematical model of Linear Regression

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

Closed form solⁿ

in this case we have formula to calculate values/solⁿs

This technique is called OLS (Ordinary Least Squares)

↳ scikit learn uses this internally when we use the class `LinearRegression()`

Here the formula is:-

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

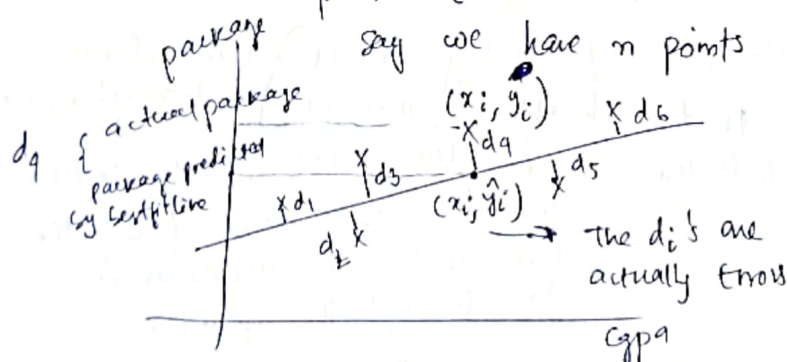
$\bar{y}$ = package ($\bar{y}$ → mean)

$\bar{x}$ = cgpa ($\bar{x}$ → mean)

m = slope

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

Best fit line wo hote hai jo saare points ke pass se hokar guzarne ka try karega. ye line each point ki distance minimize karta hai.



Total error

$$\therefore E = d_1 + d_2 + \dots + d_n$$

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

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

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

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)

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

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

Page 41.

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

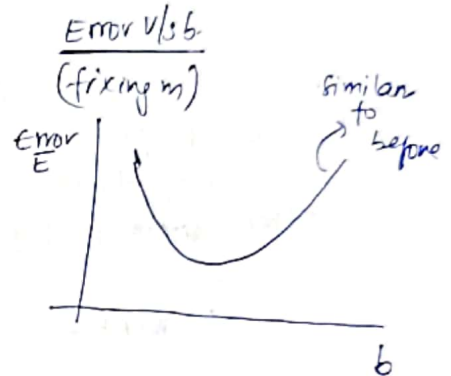
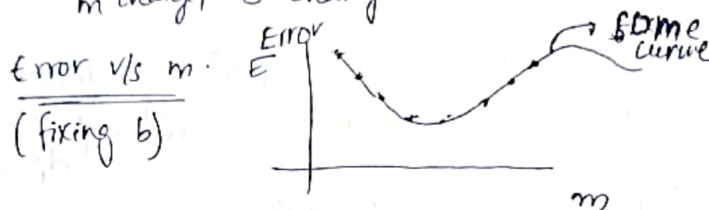
note: $\hat{y}_i \rightarrow$ predicted value
 $\therefore \hat{y}_i = mx_i + b$

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

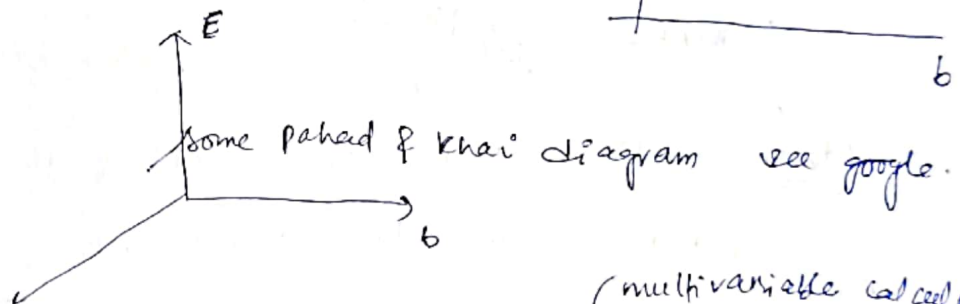
$$\Rightarrow E(m, b) = \sum_{i=1}^n (y_i - mx_i - b)^2 \quad ; \quad x_i, y_i \text{ are the input \& output data resp. \& they are fixed.}$$

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

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



$\therefore$ in 3-D.



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

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

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

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

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

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

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

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

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

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

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

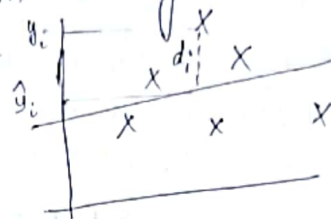
Jupyter Notebook.

Regression Metrics

- (1) MAE $\rightarrow$ mean Absolute Error,
- (2) MSE $\rightarrow$ Mean Square Error,
- (3) RMSE $\rightarrow$ Root Mean Square Error,
- (4) R^2 -score $\rightarrow$ Coefficient of determination or Goodness of fit.
- (5) Adjusted R^2 -score.

① MAE:

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



We know what d_i is $\rightarrow d_i \rightarrow$ its point regression line ke respect mei kitni galti kar raha hai. (we saw:-)

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

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

$\rightarrow$ It is basically the loss function.

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

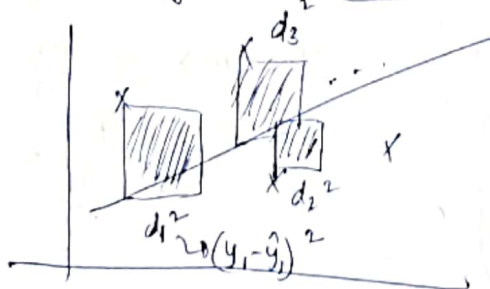
(ii) Robust to outliers (handles outliers relatively better)

Disadv:- (i) MAE is not diffble 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:-



Hence, we take the mean/average of this areas = MSE & try to minimize it.

Adv:- It is diffble everywhere

Disadv:- (i) The unit of MSE is square of the unit of y-axis, hence it's 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 $\because$ for outlier very big areas are considered in MSE

Hence: agar Data mei jyada outliers hai then go with MSE & kam hai -then go with RMSE.

Page 42

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

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

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

↳ Not Robust To outliers, just like MSE.

→ As a beginner → techno metrics niral lena.

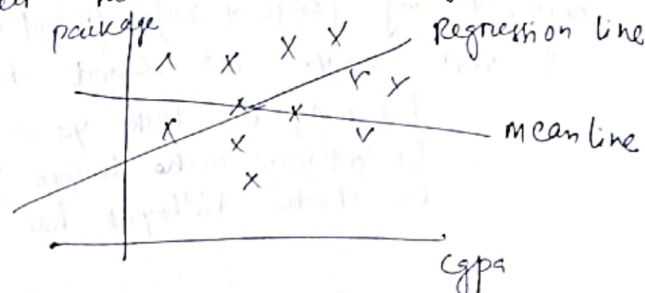
↳ Coeff of Determination or Goodness of fit.

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

(worst case scenario)

It works with any type of data set with Regression problem.

Suppose if we had only the package data & someone asked how much package he will get. The we have only one measure (the mean); we'll calculate the mean of all the packages & 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 kitna better hai.

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

SSR = sum of squared d's wst Regression line
SSM = " " " " wst Mean line.

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

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 perfect line

Hence: jitha Reg. line perfection ke taraf move karke hai pe hota hai utna R²-Score 1 & jitha Reg line kharab hota hai utna R²-Score 0 ke taraf move karke hai.

* what if R^2 -score is $-ve \Rightarrow SS_R > SS_M \Rightarrow$ Regression line, flatter than mean value line (worst case) $\Rightarrow$ shi jyada galthi kar rahi hai.
This should not happen.

★ Another Interpretation of R^2 -score:

Say we have $\begin{array}{c|c} \text{cgpa} & \text{package} \\ \hline \vdots & \vdots \end{array}$
& R^2 -score = 0.80
This means cgpa explains the 80% of variance in the package (lpa) (y-axis) (output)
ie 80% jo variance hai in package (is ki ba 3.1 lpa hai & ki ba 4.2 lpa etc),
-that is ^{due to} cgpa
& rest 20% variance in package $\rightarrow$ we don't know.

fr. $\begin{array}{c|c|c} \text{cgpa} & \text{IQ} & \text{package} \\ \hline \vdots & \vdots & \vdots \end{array}$

$\rightarrow R^2$ -score = 0.80
Then again 80% variance of package is explained by (cgpa + IQ) (input cols)
& rest 20% we don't know.

$\hookrightarrow$ may be luck ya
 $\hookrightarrow$ interview accha hogya ya
 $\hookrightarrow$ chacha Vidhayer hai etc.

⑤ Adjusted R^2 -score :-

Say we are using R^2 -score with dataset $\begin{array}{c|c} \text{cgpa} & \text{package} \\ \hline \vdots & \vdots \end{array}$ & the R^2 -score = 0.80 & now if we add further input col say IQ $\rightarrow \begin{array}{c|c|c} \text{cgpa} & \text{IQ} & \text{package} \\ \hline \vdots & \vdots & \vdots \end{array}$ then now the R^2 -score will increase say 0.90. $\rightarrow$ which is OK
package may depend upon ~~from~~ IQ.
But say if we add an irrelevant col ie temperature (this should not have any impact upon package) but even then sometimes R^2 -score increased indicating temp is affecting package.
Hence we use adjusted R^2 -score:-

formula :-
$$\text{Adj. } R^2\text{-score} = 1 - \left[\frac{[1 - R^2\text{-score}](n-1)}{n-1-K} \right]$$

 $n \rightarrow$ ^{in x-test} total no. of rows/datapoints
 $K \rightarrow$ total no. of indep. cols
(only cgpa $\rightarrow K=1$
cgpa + IQ $\rightarrow K=2$
cgpa + IQ + temp $\rightarrow K=3$)

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

- Since fallu ke col add karne pe K $\uparrow$ hai deno hai
(like temp) (less)

$$\therefore \frac{(1-R^2)(n-1)}{n-1-K} \uparrow \text{res}$$

• Adding Relevant col say $R_2 \text{ adj}$ $\downarrow$ ses

$K \uparrow$ deno $\downarrow$; again in 1-R2 in num $\downarrow$ ses rapidly $\therefore$
R2 $\uparrow$ ses hence Num $\downarrow$ ses. & in such case the $\downarrow$ in Num
is greater than the $\downarrow$ in Deno, Hence R2-adj $\uparrow$ ses hogz.
Hence ρ_{adj} is better.

Hence R^2_{adj} is better if we are dealing with Multiple Linear Regression.

Tracy's Notebook.

Real world data are of multiple Linear Regression. Ex:

MULTIPLE LINEAR REGRESSION

↳ more than 1 input col ex:

cgpa	Gender	Id	package
:	:	:	:

input / independent.

x_1 | x_2 | x_3 | y

↑
 output / dependent

→ output /
if depends

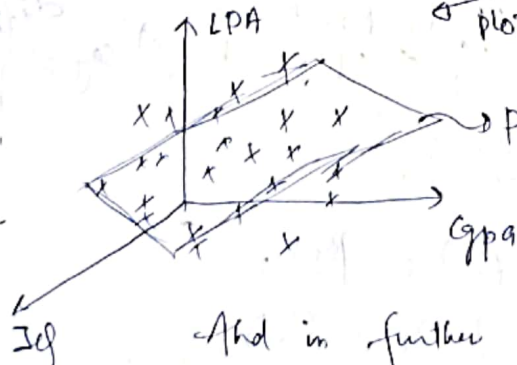
- Simple LR is a special case of Multiple LR
- The concepts in Simple LR are as it is applicable to Multiple LR.

- The concepts in Simple LR are as it is applicable to Multiple LR.

Consider the data set :-

Cyfra	Id	Package
:	:	:

← plot it in 3D



→ plane of best fit (similar behaviour to line of best fit)

✓
3d And in further higher Dimensions we draw hyperplane of best fit.

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

$$y = mx_1 + nx_2 + b$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 = \beta_0 + \beta_1 \text{cgrpa} + \beta_2 \text{Id}$$

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

Silly in n -Dim: we would be searching the eqn 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 than IQ and so on.

$\beta_0 \rightarrow$ intercept / offset value.

Mathematical formulation of Multiple LR

$\rightarrow$ finding out $Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ (ie $\beta_0, \beta_1, \beta_2$)

for dataset:-

CGPA	IQ	LPA
$\vdots$	$\vdots$	$\vdots$

and if dataset is:-

CGPA	Gender	IQ	LPA
x_1	x_2	x_3	$\vdots$

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 will be difficult.

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 of all 100 students.

$\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_{100} \end{bmatrix}$

$= \begin{bmatrix} \beta_0 & \beta_1 x_{1,1} & \beta_2 x_{1,2} & \beta_3 x_{1,3} \\ \beta_0 & \beta_1 x_{2,1} & \beta_2 x_{2,2} & \beta_3 x_{2,3} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_0 & \beta_1 x_{100,1} & \beta_2 x_{100,2} & \beta_3 x_{100,3} \end{bmatrix}$

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

Page 44

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \dots & \beta_m x_{1m} \\ \beta_0 & \beta_1 x_{21} & \dots & \beta_m x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_0 & \beta_1 x_{n1} & \dots & \beta_m x_{nm} \end{bmatrix}$$

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

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

Now: The output Data:-

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

And Here:-

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

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

$$\begin{aligned} \Rightarrow E = e^T e &= (y - \hat{y})^T (y - \hat{y}) = (y^T - \hat{y}^T) (y - \hat{y}) \\ &= (y^T - (X\beta)^T) (y - X\beta) \because \hat{y} = X\beta \\ &= (y^T - \beta^T X^T) (y - X\beta) \rightarrow \text{All are matrices so dist. prop holds.} \\ &= y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X \beta \\ &= y^T y - y^T X\beta - (X\beta)^T y + (X\beta)^T X\beta \end{aligned}$$

Now: $y^T X\beta = y^T \hat{y} = [y_1 \dots y_n] \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = y_1 \hat{y}_1 + \dots + y_n \hat{y}_n$
& $(X\beta)^T y = \hat{y}^T y = [\hat{y}_1 \dots \hat{y}_n] \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \hat{y}_1 y_1 + \dots + \hat{y}_n y_n$
so $y^T X\beta = (X\beta)^T y$ → or basically we need to prove

$$\Rightarrow E = y^T y - 2y^T X\beta + \beta^T X^T X \beta$$

↳ Error or loss function

→ $(y^T X\beta)^T = y^T X\beta$ → $y^T X\beta$ is symmetric
↳ y^T is 1×1 matrix hence it is symmetric

$E = Y^T Y - 2Y^T X \beta + \beta^T X^T X \beta \rightarrow$ we need to minimize this
 choose β so that this gets minimized.
 same method (diff wrt β) :-

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

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

$$-2Y^T X + 2X^T X \beta^T = 0$$

matrix differentiation

$$Y^T X = X^T X \beta^T$$

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

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

$$\beta = \underbrace{(X^T Y)}_{\text{shape } (m+1, 1)} \underbrace{(X^T X)^{-1}}_{\text{shape } (m+1, m+1)}$$

Learn

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

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

$X = X_{\text{train}}$
 $Y = Y_{\text{train}}$

Why Gradient Descent?

Scikit-Learn uses 2 methods to solve Linear Regression problems

- OLS (Closed form) $\rightarrow$ Here we find formula the one we found above.
- (nonclosed form) $\rightarrow$ Gradient Descent

$\rightarrow$ The reason is the inverse operation used above to obtain β ; the $(X^T X)^{-1} \rightarrow$ 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 computation just to find the inverse. (This is taking 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

Multiple Linear Regression then we use Gradient Descent.

Scikit Learn

- OLS Linear Regression class
- Gradient Descent SGD Regressor class.

approximation Technique.
 Jupyter for Notebook.

Gradient Descent

Page: 45

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.

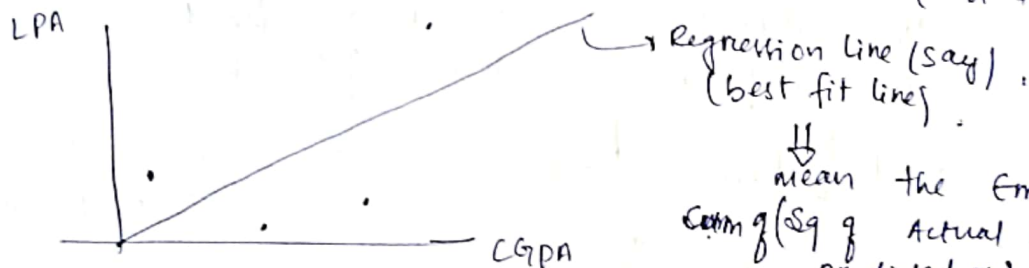
* Gradient Descent is a General Algorithm

- ↳ Linear Regression
- ↳ Logistic Regression
- ↳ Tsne
- ↳ 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 (ie. CGPA | LPA of 4 students)



mean the Error = loss for
 $\sum (y_i - \hat{y}_i)^2$ (Actual y - predicted y) should be minimum.

To keep the discussion simple suppose we know what m is (using closed form) loss for.

Say $m = 78.35$
 then loss for $\Rightarrow E(b) = \sum_{i=1}^n (y_i - 78.35x_i - b)^2$
 $\hookrightarrow$ depends only upon b .

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

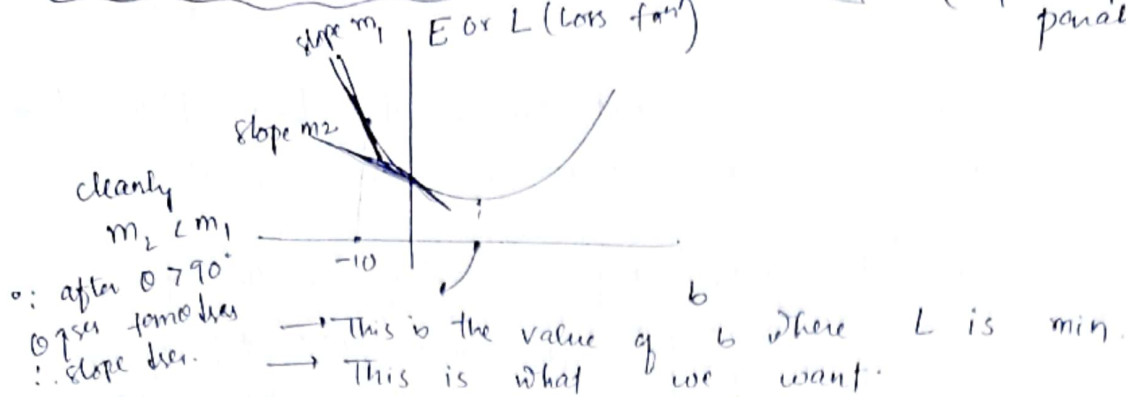
$$E = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

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

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

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

The Per b/w b and b is quadratic $\therefore$ (in fact upward parabola)



Now we start with Gradient Descent

step 1: select a Random ~~b~~ 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 $\downarrow$ ses. There are two directions to move; how would the algo know which direction to move so that the fn value $\downarrow$ ses → This is slope

The algo should move in that 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 $\downarrow$ se b
 if slope is $-ve$ then $\uparrow$ se b .

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

Note as we go closer & closer to actual b the slope $\downarrow$ ses & hence the jump in b $\downarrow$ ses.

This eqn will bring drastic changes in values of b $\therefore$ we use iterations
 $\therefore$ Hence we use a transformed eqn:

$b_{new} = b_{old} - \eta \text{ slope}$; 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 sol vvv slowly.

The step is iterative, so when do we stop.

Page 46

We stop when $b_{new} - b_{old} = 0.0001$ or even less than that
that means we are converging to the actual b value

Actually: $b_{new} = b_{old} - \eta \text{slope}$

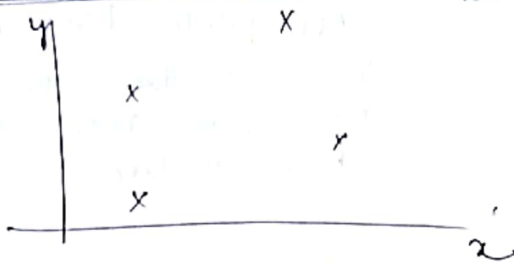
$\Rightarrow b_{new} - b_{old} \rightarrow -\eta \text{slope}$ which should be very small

means the slope is becoming 10^4
to x -axis hence we are reaching our point of min.

Or: we can fix the no. of iterations (called epochs), k_i hum
the k_i iterations range of 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 $m = 78.35$
(for simplicity)
we need to find $b \rightarrow y$ -intercept

Step 1: Start with a Random Value of b

for i in epochs $\rightarrow$ # of iterations:-

$$b_{new} = b_{old} - \eta \times (\text{slope at } b_{old}) ; \eta = 0.01 \rightarrow \text{Learning Rate}$$

$$b_{old} = b_{new}$$

How to find?

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

note: $\eta \text{slope at } b_{old} = \text{step size}$

This is Gradient Descent

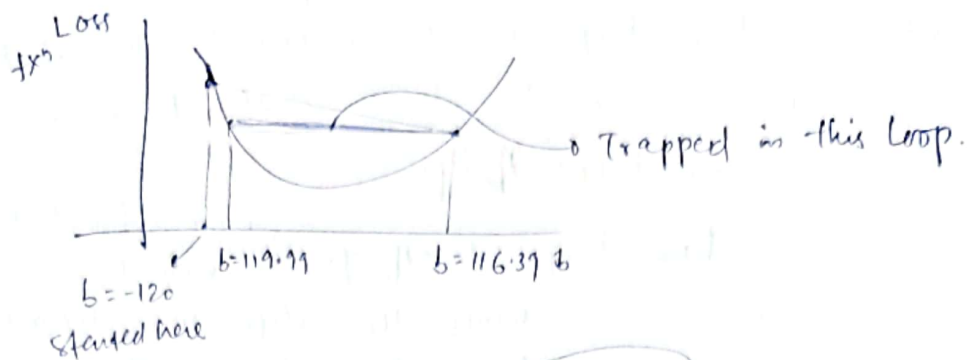
$$\frac{dL}{db} = \sum_{i=1}^n -2(y_i - mx_i - b)$$

Jupyter Notebook for

upto this part

$$\text{slope at } b_{old} = \left. \frac{dL}{db} \right|_{b=b_{old}}$$

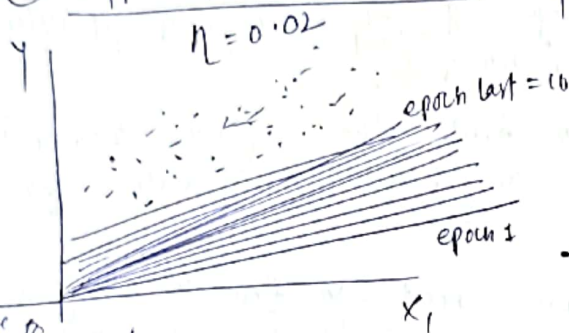
Gradient Descent $\rightarrow$ file 2 $\rightarrow$ Drawing 1:



Few Discussions.

(1) Effect of Learning Rate

Say epoch = 10
 $\eta = 0.1$

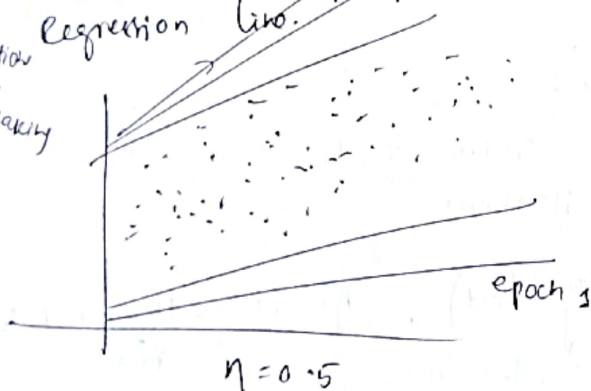


in this case to reach the actual reg. line we need more epochs

Learning Rate too low

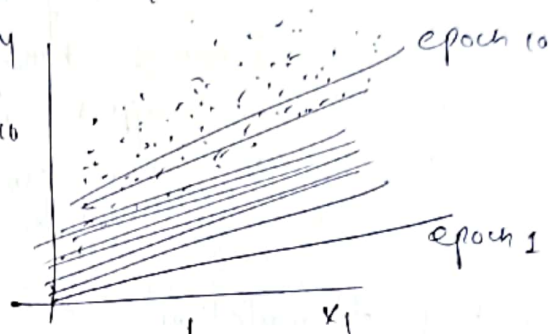
Cannot reach the actual Regression line.

means more calculation & more time taking



$\eta = 0.5$

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.



Appropriate Learning Rate:- reached then in the epoch range, we reach the actual Regression line.

Learning Rate Data pe depend karta hai, chala kar hi dekhna hoga.

It is a hyperparameter. Sahi tuning of Learning Rate in Gradient Descent is very important.

② The universality of Gradient Descent

→ So far we studied Gradient Descent w/ 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 condⁿ 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 & $lr = 0.01$

then

for in epochs:

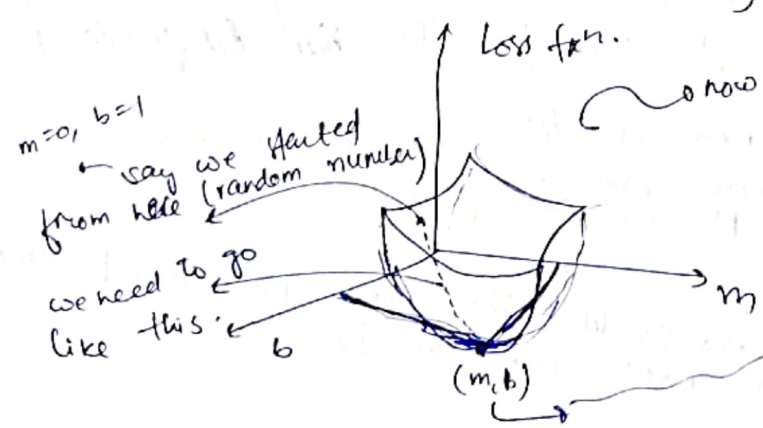
update b : $b = b - \eta \text{slope}$ → This slope we know, we saw previously

update m : $m = m - \eta \text{slope}$ → let's see what this slope is.

Loss fn:-
$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum (y_i - mx_i - b)^2$$

$$L(m, b) = \sum (y_i - mx_i - b)^2$$

→ paraboloid in 3-D



→ now we need to find m & b such that loss fn is minimum

So now we need the gradient descent with resp to both m & b .

Now: on this paraboloid at any point the slope will be a tangent plane $\rightarrow$ it will have two components; one component in direction of m & other in direction of b .

diff of Multivariable func is gradient.

$\therefore$ In the formulas:

$$b = b - \eta \text{ slope} \quad \text{this is slope wrt } b \quad \text{ie. } \frac{\partial L}{\partial b} \Big|_{b=b_{old}}$$

$$m = m - \eta \text{ slope} \quad \text{this is slope wrt } m \quad \text{ie. } \frac{\partial L}{\partial m} \Big|_{m=m_{old}}$$

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

$$\therefore \frac{\partial L}{\partial b} = -2 \sum_{i=1}^n (y_i - mx_i - b) \quad \text{slope of } L \text{ wrt } b$$

$$\& \frac{\partial L}{\partial m} = -2 \sum_{i=1}^n (y_i - mx_i - b) x_i \quad \text{slope of } L \text{ wrt } m$$

(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.

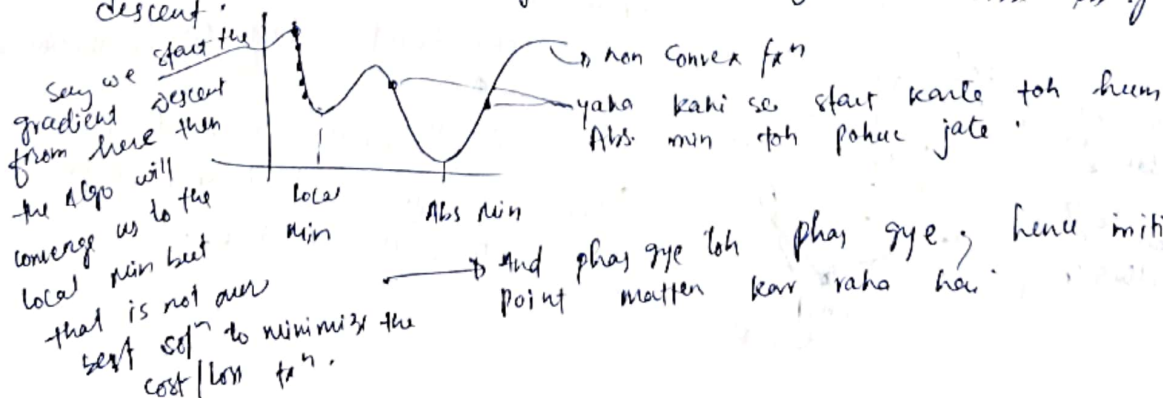
Supra Note Book.

Such fns are good for gradient descent

Effect of loss fn on Gradient Descent:-

So far we had loss function:- $L = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ is a convex loss fn.

But if we have nonconvex functions then there can be more than 1 minima (local minima) but to minimize the loss fn we need to find the global minima. & local minimas are big problems for the ~~loss~~ gradient descent.



In higher Dimensions say we have 10 cols, Page 18
 we cannot check for ~~convex~~ hai ya nahi & agar initialisation
 agar aisa hua ki humara Algo local minimum pe converg
 kar jya then we have to accept it.

↳ isse bachne ke kuch tarike hote hai like
 tricks for initialization, depending on cost function.
 (this we do in deep learning ∴ wohi pe jyada
 non convex for milte hai.

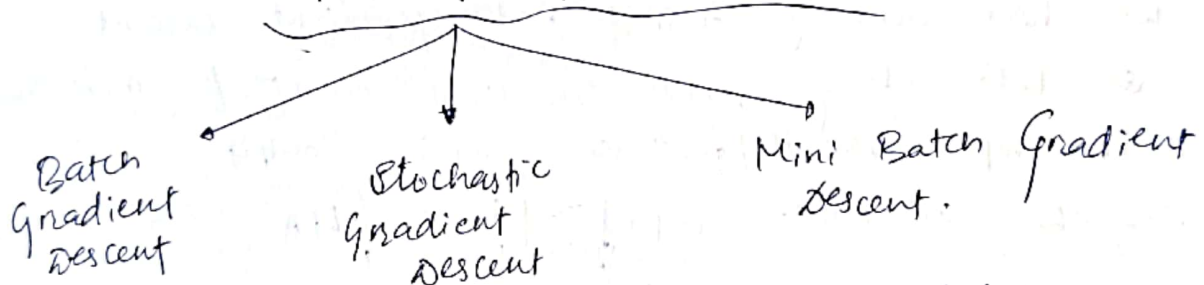
There is yet another problem if the loss/cost function has
 Saddle points (where the slope change karta hai & hence
 kind of flat plateau
 actual sor pe jane ke liye both jyada time/epochs lagenge.

Effect of Data on Gradient Descent:

Agar features in the data are equally Scaled then the
 contour plot of the cost fn/loss fn ~~is~~ is circular and we
 descent faster to the point of minima.

Hence for Linear Regression → Scale the input cols
 to the same scale.

Types of Gradient Descent



What is the basis of this Types/distinction:

we had two eqn:

$$m_{\text{new}} = m_{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial m} \Big|_{m=m_{\text{old}}} \right)$$

$$\& b_{\text{new}} = b_{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial b} \Big|_{b=b_{\text{old}}} \right)$$

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

Now in Batch Gradient Descent :- To calculate the
 slope we use the entire data. & we 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 & make the loss fun/Error & make changes in m & b $\rightarrow$ faster with large data set but

$\therefore$ pure dataset pe ek baar Error phone. we make 300 updates considering there were 300 total rows/day.
Again there is Mini Batch Gradient Descent (also Batch & stochastic Grad. descent).

In this we decide a Batch size say 30, & there were 300 data originally; then the update in m & b will be based on this 30 rows hence ek baar dataset pe ghumenge (300 rows) matlab 10 bar update karenge m & b ko.

Batch gradient descent is quite Rare to use:-

$\rightarrow$ It is used generally when we have convex loss fun like in Linear Regression. But still there is one condⁿ 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 $\rightarrow$ input output

Say the dataset is:-

CGPA	IQ	Gender	LPA
.	.	.	.
.	.	.	.
.	.	.	.

Now the Regression hyperplane will be:-

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \rightarrow 4 \text{ dimensional.}$$

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$
LPA CGPA IQ Gender

So if we have n input cols:-

$$\text{then } Y = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

Hence we need to find $\beta_0, \beta_1, \dots, \beta_n$.

$\rightarrow$ 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
col = (2+1)

CPG	QA	LPA
8.1	9.3	3.2
7.5	9.5	3.5

we will extend this.

→ Eqn of Reg plane: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ → To find $\beta_0, \beta_1, \beta_2$

$\downarrow$ $\downarrow$ $\downarrow$
 LPA CPG QA

Step 1: start with random values of $\beta_0, \beta_1, \beta_2$

Generally: $\beta_0 = 0$ & slopes, $\beta_1, \beta_2 = 1$
 ↳ intercept

Step 2: choose epochs = 100 & learning rate = 0.01 (say).

Then updating the values for each epochs:

$$\beta_0^{\text{new}} = \beta_0^{\text{old}} - \eta \left(\text{slope} = \frac{\partial L}{\partial \beta_0} \right)$$

$$\beta_1 = \beta_1 - \eta \left(\text{slope} = \frac{\partial L}{\partial \beta_1} \right)$$

$$\beta_2 = \beta_2 - \eta \left(\text{slope} = \frac{\partial L}{\partial \beta_2} \right)$$

Hence if we have n input cols then we will need n+1 such ephs with (n+1) partial derivatives.

Calculating:

$$\frac{\partial L}{\partial \beta_0}, \frac{\partial L}{\partial \beta_1}, \frac{\partial L}{\partial \beta_2}$$

Mean squared error (MSE)

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

for our above data

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

we have: $\hat{y}_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

$$L = \frac{1}{2} \sum_{i=1}^2 (y_i - \beta_0 - \beta_1 x_1 - \beta_2 x_2)^2$$

$$= \frac{1}{2} (3.2 - \beta_0 - \beta_1 \times 8.1 - \beta_2 \times 9.3)^2 + \frac{1}{2} (3.5 - \beta_0 - \beta_1 \times 7.5 - \beta_2 \times 9.5)^2$$

So 1/n Lagane $\sum$ we are calculating Mean squared error (MSE)

↓ jo $\beta_0, \beta_1, \beta_2$ MSE

ko minimise karogi wo with $\frac{1}{n}$ wale total error ko hi minimise karogi.

$$\text{So } \hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$$

$$\& \hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22}$$

$$\text{So } L = \frac{1}{2} \left[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 \right]$$

$$L = \frac{1}{2} \left[(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2 \right]$$

Now:

$$\frac{\partial L}{\partial \beta_0} = -\frac{1}{2} (y_1 - \hat{y}_1) - \frac{1}{2} (y_2 - \hat{y}_2)$$

$$= -\frac{1}{2} [(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2)]$$

for n ~~rows~~ rows

$$\frac{\partial L}{\partial \beta_0} = -\frac{1}{n} [(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + \dots + (y_n - \hat{y}_n)]$$

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

Again if we have the same dataset: $L = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

$$\frac{\partial L}{\partial \beta_1} = -\frac{1}{2} [(y_1 - \hat{y}_1) \cdot x_{11} + (y_2 - \hat{y}_2) \cdot x_{21}]$$

for n rows.

$$\frac{\partial L}{\partial \beta_1} = -\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{i1} = -\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{i1}$$

↓ slope

$$\frac{\partial L}{\partial \beta_2} = -\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{i2}$$

↳ x_{i2} is the 2nd col of the

So Now if we have m input cols: $(n \text{ rows})$ ie. $2d$ col (x_2)

then the hyperplane $y = \beta_0 + \beta_1 x_1 + \dots + \beta_m x_m$

$$\text{slope w.r.t } \beta_k = \frac{\partial L}{\partial \beta_k} = -\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{ik}$$

General formula

Jupyter Notebook:

Note we are increasing the rows not the cols ie the no. of input dataset is still 2.

Stochastic Gradient Descent :- (most imp)

Page 50.

The problem in Batch Gradient Descent is - that
 have coeff (of hyperplane of best fit) we respect our
 loss fun (L) ka derivative nikalna hoto hai to
 make one update in the coeffs.

again the loss fun is a summation over the rows.
 Now Suppose there are 1000 rows (which is very easy to
 get on real life problem)
 & cols = 5 (input cols)
 And also say there is epochs = 50.

$$\therefore L = \sum_{i=1}^{1000} (y_i - \hat{y}_i)^2$$

$\therefore$ 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 L & diff)

$$\text{slly for } \frac{\partial L}{\partial \beta_1}, \frac{\partial L}{\partial \beta_2}, \dots, \frac{\partial L}{\partial \beta_5}$$

So there are 6000 derivatives in each iteration

$\therefore$ for 50 epochs/iteration: there are

$$6000 \times 50 = 300000 = 3 \text{ lakhs derivatives}$$

Now if we have:- rows = 10^5 ; & cols = 10^2

& epochs = 10^3

then Total derivatives to be calculated

$$\text{till the loop complete} = 10^{10}$$

hence our Algo will become huge 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)
 for this X -train ko memory mei rakhna hoga tabhi dot
 product ho payega but sometimes X -train may be huge

then the calculation of $\hat{y}$ will not be possible.

So there is a hardware problem (i.e. if the system RAM is small)

In Batch Grad Descent :- After loading the entire data (X_{train}) 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 & so on till last row for a epoch.

So in a single iteration / epoch $\rightarrow$ there are $\rightarrow$ randomly selected (not sequentially) $\rightarrow$ mean Stochastic

& because of these frequent updates we converge to the correct answers in less epochs.

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 $\rightarrow$ 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 $\rightarrow$ say k^{th} row

Loss for k^{th} 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
Then Batch Gr. D with k faster than Grad. Desc.
Since in BGD there are only 100 updates to be made
But in SGD there are $100 \times \#$ 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 is faster on big data sets.

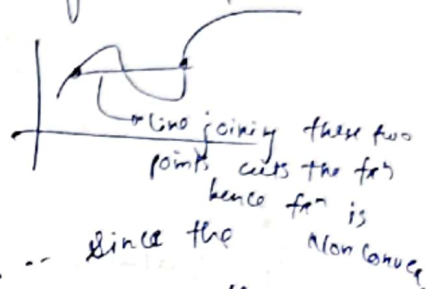
When to use stochastic Grad Descent

but for this learning rate to monitor kanna para. $\uparrow$

+ Big data

+ When the loss function is Non Convex -- Since the jumps in the SGD are random & we saw in the contour plot in stochastic GD.

& due to this 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 use:

$$t_0, t_1 = 5, 50$$

we can define our own defn accordingly.

due to this as epoch goes $\uparrow$ t goes $\uparrow$

def learning-rate(t):
return $\frac{t_0}{t + t_1}$

Hence now Lr is a fn of epoch i

Lr decreases; for i in range(epochs):

for j in range($X.shape[0]$):

of rows.

$$lr = \text{learning-rate}(i * X.shape[0] + j)$$

multiply

& hence fluctuation stabilizes towards end.

The `SGDRegressor()` class provided by `Sklearn` has a parameter called `learning_rate` = 'constant', 'optimal', 'invscaling', 'adaptive' → to handle

It is a comb of Batch & stochastic. Learning schedule.

Mini Batch Gradient Descent

in Batch Grad Descent → 1 update in 1 epoch
in stochastic GD = # of rows update in 1 epoch
in Mini Batch GD = # of chosen updates in 1 epoch

→ Here we decide # of rows say = 100 → batches (rows are random) 100 rows out of 1000 rows,
total rows = 1000 ∴ total batches = 10

∴ in each epoch # of batches updates is 10 updates in 1 epoch.

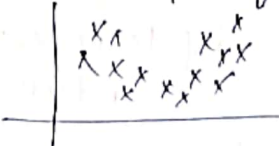
if Batch size = # of rows → it becomes Batch GD
if Batch size = 1 → it becomes Stochastic GD.

Mini Batch improves (uses) the Randomness in the Stochastic GD.

Jupyter Notebook.

POLYNOMIAL REGRESSION

Linear Reg. is all about linear data i.e. the data visualisation is kind of linear.
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.

what if the scatter plot of the data is not linear
Ex. 

→ 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. reln b/w them Page 52.

then we find $\begin{array}{c|c|c|c} x^0 & x^1 & x^2 & y \\ \hline 1 & 35 & 35^2 & 12 \end{array} \rightarrow$ I send this as x-train of 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 degree = 3.
we will have:

x^0	x^1	x^2	x^3	y
1	35	35 ²	35 ³	12
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

& 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 regression curve will underfit & ML engineer predicts this. attributes/properties fetch na kar paye. Data ki saani
if this value is high then the regression curve will overfit (it tries to touch every point).

What if Dataset is:-

x_1	x_2	y
$\vdots$	$\vdots$	$\vdots$

$\rightarrow$ Then

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2$$

So the Dataset becomes:-

Now consider this:-

x_1^0	x_1^1	x_1^2	x_2^1	x_2^2	y
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\downarrow$ copy just change the col names

$\bar{x}_0$	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_3$	$\bar{x}_4$	y
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\rightarrow$ New data set

Jupyter Notebook.

$$y = \beta_0 + \beta_1 \bar{x}_1 + \beta_2 \bar{x}_2 + \beta_3 \bar{x}_3 + \beta_4 \bar{x}_4$$

Now this is equivalent to Multiple Linear Regression to solve

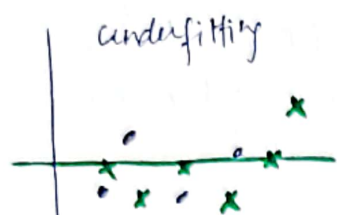
gole na vana

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.

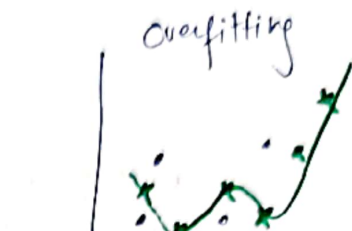
x - training set
 $\bullet$ - test set.



M1 High Bias Low Variance.



M2 Low Bias Low Variance



M3 High Variance Low Bias

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 $\rightarrow$ Low Bias

M2 $\rightarrow$ There is curve in the training data & the model is able to capture that curve.

M1 $\rightarrow$ High bias since it is not capturing the essence of the training data itself (the curved nature).

Variance :- It is the difference of fits 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 $\rightarrow$ 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 $\rightarrow$ $\therefore$ High variance.

Overfitting: 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 aacha kam nahi kar pa rha $\rightarrow$ high bias.

Thus we need a model with low Bias & low Variance.

* Kachhi kachhi ke keep a bit bias so that variance less. we need to find the sweet spot b/w under & over fitting and There are three ways to do it

- ├ Regularization
- ├ Bagging
- └ Boosting

we use this to reduce overfitting

Regularization

$\rightarrow$ It is a technique using which you induce some added info in a ML model so that you can reduce overfitting.

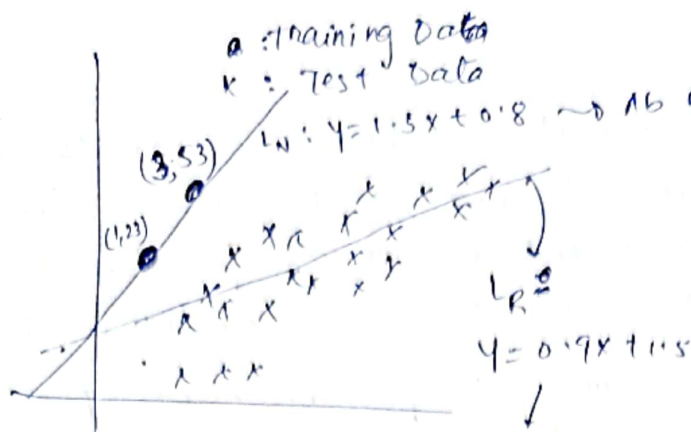
- There are three ^{most used} kind of Regularization Techniques
- ├ Ridge Reg. or L2 Regu.
 - ├ Lasso Reg or L1 Regu.
 - └ Elastic Net (Combination of above two)

Let's talk overfitting w.r.t Linear Regression :-

for one input & one output :- we are finding $y = mx + b$; $m = \text{slope} = \text{weightage of } x \text{ to find } y$.

In Linear regression if we have overfitting 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 ^{thoda} kam karna hai.

considers.



Ab bus do points par calculate krake line of Best fit nikalna hai - then undonon se pass krane wala line hi niklega.

And clearly ye line actual data ke essence ko miss kr dega & it is a case of overfitting.

Consider this - the hypothetical line jo training data pe toh galatya karega but test data pe better perform karega.

Ab humme ab kisi faisle ki model ko convince karna hai ki donot choose L_N ; Chose L_R .

How do we choose the line on training data set:

By Minimizing the Loss function/cost fn on the training data 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 .

this is a hyperparameter, we can tune it. Generally:- $\lambda \in (0, \infty)$

Loss L_N ($\lambda=1$)

Loss L_R ($\lambda=1$)

$$L = 0 + 1.5^2 \times 1 = 2.25$$

Since L_N pass hi undono ke upar hi kr raha hai

$$\begin{aligned} L &= \sum_{i=1}^2 (y_i - \hat{y}_i)^2 + \lambda m^2 \\ &= (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (0.9^2 \times 1) \\ &= (23 - 0.9 \times 1 - 1.5)^2 + (53 - 0.9 \times 3 - 1.5)^2 + 0.9^2 \\ &= 2.03 \end{aligned}$$

$y_1 = 23$ & $y_2 = 53$
 $x_1 = 1$, $x_2 = 3$
 $\hat{y}_i = mx_i - b$

Hence Now the ML model will choose the L_R line as line of best fit Even though it makes more mistakes on the training data Since the loss of L_R line is $<$ Loss of L_N line.

→ Trade off although Bias there badh zyga but Variance significantly ghat zyga. → And this is called Bias-Variance trade off.

Note: if there was two input cols & a output $\hat{y}$ then :-
 $n = \# \text{ of data in training dataset}$
 $L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda (m_1^2 + m_2^2)$ since here square of multiply kar rahi hai hence it is L2 Norm hence the name L2 Regularization.
 sly if there are k input cols then :-
 $L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda (m_1^2 + m_2^2 + \dots + m_k^2)$

Jupyter Notebook

Mathematical formulation of Ridge Regression & Building our own Ridge Regression class :-

Regression line $y = mx + b$

Ridge Regression for 2D Data (one input col) (n rows)

for this case the Loss function is :-

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda m^2 \rightarrow \text{to find } b \text{ \& } m \text{ we want to reduce the slope}$$

Now: $\frac{\partial L}{\partial b} = 0 \rightarrow b = \bar{y} - m\bar{x}$ ($\bar{y} = \text{mean of } y$, $\bar{x} = \text{mean of } x$)

Again: $\frac{\partial L}{\partial m} = \frac{\partial}{\partial m} \sum_{i=1}^n (y_i - mx_i - b)^2 + \lambda m^2$
 $= \frac{\partial}{\partial m} \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 + \lambda m^2$
 $= \sum_{i=1}^n 2(y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) + 2\lambda m = 0$
 $= -2 \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) + 2\lambda m = 0$

- training 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 $\lambda \geq 0$

for $\lambda = 0$ $m = \frac{\sum_{i=1}^n (y_i - \bar{y}) + (x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$

; hence

Since now we are adding $\lambda \geq 0$ to denom $\therefore m$ ↓ yes
 & $\therefore$ bias ↑ but variance ↓ & 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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

 $\sim (n+1)$ cols & m rows

Loss fn in this case:

we learnt earlier that

$$L = \sum_{i=1}^m (y_i - \hat{y}_i)^2 \text{ can be written as}$$

$$L = (XW - Y)^T (XW - Y) \text{ where } Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \text{ \& } X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & \dots & \dots & x_{mn} \end{bmatrix}$$

$$y = w_0 + w_1 x_1 + \dots + w_n x_n$$

So we need to find $(w_0, \dots, w_n)$
 $w_0 \rightarrow$ intercept
 $w_i \rightarrow$ slopes/weights

And the Loss for the Ridge Regression (by adding a Regularization term) :-

Page 55

$$L = (XW - Y)^T (XW - Y) + \lambda \|W\|^2$$

$\hookrightarrow \|W\|^2 = w_0^2 + w_1^2 + \dots + w_n^2$
 $\hookrightarrow L_2$ Norm square.

$$L = (XW - Y)^T (XW - Y) + \lambda W^T W$$

We need to find W matrix: ~ This is the Loss for Ridge Reg for n Dim Data.

$$L = ((XW)^T - Y^T) (XW - Y) + \lambda W^T W$$

$$L = (W^T X^T - Y^T) (XW - Y) + \lambda W^T W$$

$$L = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y + \lambda W^T W$$

$\hookrightarrow$ These two terms are the same (did earlier)

Now:

$$L = W^T X^T X W - 2 Y^T X W + Y^T Y + \lambda W^T W$$

$$\text{or } L = W^T X^T X W - 2 W^T X^T Y + Y^T Y + \lambda W^T W$$

$$\text{Now } \frac{dL}{dW} = 2 X^T X W - \cancel{2 X^T Y} + 2 \lambda W = 0$$

$$W (X^T X + \lambda I) = X^T Y; \quad [I_{n \times n, n \times n}]$$

$$W = \frac{X^T Y}{\lambda I + X^T X}$$

$$\Rightarrow W = (X^T X + \lambda I)^{-1} X^T Y$$

~ earlier it was:

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

code for this part

Jupyter for Notebook.

Some common vector derivatives:	
$X^T B \xrightarrow{\frac{dB}{dB}}$	B
$X^T B \rightarrow$	b
$X^T X \rightarrow$	$2X$
$X^T B X \rightarrow$	$2B X$

Now:- What if the data set is huge & we have multiple input cols $\rightarrow$ Then calculation for the inverse will take a lot of time & hence we'll use

Ridge Regression Technique using Gradient Descent

So for multiple input cols (say n & rows $= m$)

Loss fun with added regularization term

$$L = (XW - Y)^T (XW - Y) + \lambda \|W\|^2 \quad \text{choose starting } W = [0, 1, \dots, 1] \text{ for Gradient Descent.}$$

$$L = (XW - Y)^T (XW - Y) + \lambda W^T W; \quad \text{where the terms have the same meanings as discussed in previous section}$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}; \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}; \quad W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \left\{ \begin{array}{l} \text{intercept} \\ \text{slopes} \end{array} \right.$$

Now we need to update the slopes & intercept values in Gradient Descent using:-

$$w_0 = w_0 - \eta \frac{\partial L}{\partial w_0}, \quad \dots, w_i = w_i - \eta \frac{\partial L}{\partial w_i}, \quad \dots, w_n = w_n - \eta \frac{\partial L}{\partial w_n}$$

$\downarrow$ in a single formula

$$W_{\text{new}} = W_{\text{old}} - \eta \frac{\Delta L}{\Delta W} = W_{\text{old}} - \eta \text{grad } L$$

$$\begin{bmatrix} w_{0_{\text{new}}} & \dots & w_{n_{\text{new}}} \end{bmatrix} = W_{\text{old}} - \eta \nabla L = \begin{bmatrix} w_0 & \dots & w_n \end{bmatrix} - \eta \left[\frac{\partial L}{\partial w_0}, \dots, \frac{\partial L}{\partial w_n} \right]$$

$$\text{we have:- } L = \frac{1}{2} (XW - Y)^T (XW - Y) + \frac{1}{2} \lambda W^T W \quad \left(\frac{1}{2} \text{ multiply kiya for mathematical convenience} \right)$$

$$L = \frac{1}{2} [W^T X^T X W - 2 \cancel{W^T X^T Y} + Y^T Y] + \frac{1}{2} \lambda W^T W \quad \left(\downarrow \text{minimization karna hai} \right)$$

$$\frac{dL}{dW} = \frac{1}{2} [2 X^T X W - 2 X^T Y] + \frac{1}{2} 2 \lambda W$$

$$\frac{dL}{dW} = X^T X W - X^T Y + \lambda W \rightarrow \text{or } \left[\frac{\Delta L}{\Delta W} \right]$$

Choose start $W = [0, 1, 1, \dots, 1]$ each then in epochs we update $W = W - \eta \frac{dL}{dW}$

Jupyter Notebook

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 bias & Bias & variance goes.

1/ How the coefficients get affected in Ridge Regression?

$\lambda \geq 0$ then if $\lambda = 0$ then we are using Simple Linear Regression.

if $\lambda > 0$ and as we $\uparrow \lambda$ 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

coeff $w_1 = 1000$ $w_2 = 10$ $w_3 = 1$

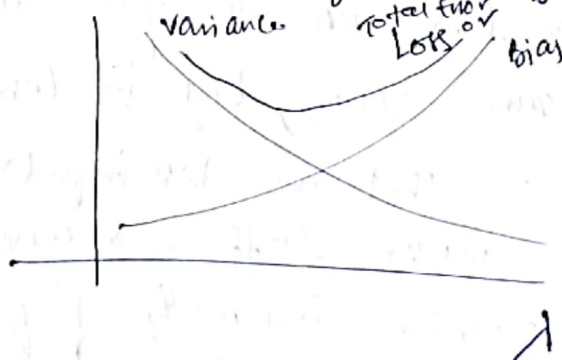
→ this means that y calculates knows like x_1 is most imp. then x_2 & then x_3 is least important.

Now as we $\uparrow \lambda$ then w_1 will be affected the most (uses faster) & w_3 is the least affected (uses slowly)

3/ Effect of Bias Variance Tradeoff:-

As $\lambda \uparrow$ up to some point the Bias $\uparrow$ but the variance $\downarrow$
but if we continue $\uparrow \lambda$ it then Bias will $\uparrow$ but the curve may underfit ~~the curve may underfit~~

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 . → There both bias & variance are least.

4. Effect of Regularization on Loss fun.

The Loss fun:- $L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|w\|^2$

this if we divide by n then it becomes mean squared error (MSE).

$\lambda \geq 0$

Logit is pos, hence the loss fun $\uparrow$ upon regularization

* Mostly Ridge Reg tabs apply kanna chahiye when # of input cols is ≥ 2 .

Lasso Regression or L1 Regularization $\rightarrow$ It also helps in reducing the overfitting.

in Ridge Regression: we added a Reg. term

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|w\|^2, \|w\|^2 = w_0^2 + \dots + w_n^2$$

$\downarrow$ in Lasso also we do the same.

here we just use L1 Norm.

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|w\|, \text{ where } \|w\| = |w_0| + |w_1| + |w_2| + \dots + |w_n|$$

Jupyter Notebook - using Scikit Learn.

as $\lambda \uparrow$ we try to min $\|w\| \rightarrow$ if decreasing too much may cause underfitting.

~~***~~ In Ridge Reg $\rightarrow$ no matter how much we $\uparrow$ α /lambda, the coeff never became zero, but in Lasso, as we $\uparrow$ lambda, the x_i 's that are less important to determine y (output) unka 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 increases then the coefficient (slope) in case of Lasso regression becomes zero $\rightarrow$ this 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.

x/y $\rightarrow$ Linear Regression case: - we have to find the line
 $b = \bar{y} - m\bar{x}$ we know $\bar{y} = \text{mean of } y$
 $\bar{x} = \text{mean of } x$.

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

These were the formula for Simple Linear Regression.

& 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:

$$b = \bar{y} - m\bar{x} \text{ (same)}$$

This 2 is added just for mathematical conv.

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda |m|$$

$$L = \sum_{i=1}^n (y_i - mx_i - b)^2 + \lambda |m| = \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 + 2\lambda |m|$$

$m > 0$.

$$L = \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 + 2\lambda m$$

$m < 0$.

$$L = \sum_{i=1}^n (y_i - mx_i - \bar{y} + m\bar{x})^2 - 2\lambda m$$

Since ye modulus hai iska direct diff nahi kar sakte, we have to break it.

$$m > 0$$

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

$$\frac{\partial L}{\partial m} = \sum_{i=1}^n 2(y_i - m x_i - \bar{y} + m \bar{x})(-x_i + \bar{x}) + 2\lambda = 0$$

$$\Rightarrow \lambda - \sum_{i=1}^n (y_i - m x_i - \bar{y} + m \bar{x})(\bar{x} - x_i) = 0$$

$$\Rightarrow \lambda - \sum_{i=1}^n ((y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2) = 0$$

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

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

(Note: The numerator is circled and labeled "this is fixed")

clearly as λ increases the value of m decreases. if we can pick λ s.t. $m = 0$ & if we further pick λ then m gets -ve. value!

but -ve value not possible

Since in that case we use this formula

if $m = 0$. then we will have Simple Linear Reg & then $m = \frac{\sum (y_i - \bar{y})(x_i - \bar{x})}{\sum (x_i - \bar{x})^2}$

& if $m < 0$.

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

Say: $m > 0$.
 $m = \frac{y x - b}{x^2}$

Say $y x = 100$ & $x^2 = 50$
 $\lambda = 0 \rightarrow$ we get $m = 2$
 $m = \frac{100 - b}{50}$

as λ increases m decreases & at $\lambda = 100$ $m = 0$

& if $\lambda > 100$ then m gets -ve but as m gets negative

& $m < 0$.
 $m = \frac{y x + b}{x^2}$

this formula is used

$m = \frac{100 + 150}{50} = 5$ $\rightarrow$ Now slope increased even more than the simple linear ($\lambda = 0$) Reg case
 not good
 hence the algorithm stops increasing or fixing m after $m = 0$.

Now if $m < 0$.

Page 58.

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

$\therefore \lambda > 0 \therefore$ for $m < 0$

$YX < 0$ say $YX = -100$ & $X^2 = 50$

$$m = \frac{-100 + \lambda}{50}$$

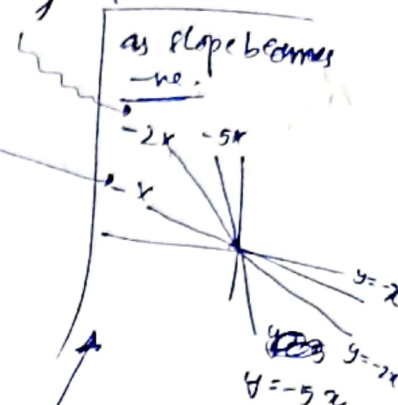
$\rightarrow$ Now as λ increases m increases (simple linear case: $\lambda = 0$ $m = -2$)

$$\lambda = 50 \rightarrow m = -1$$

$$\lambda = 100 \rightarrow m = 0$$

$\lambda = 150 > 100 \approx |YX|$ then m becomes pos but in that case $m > 0$ formula will be used.

$$m = \frac{YX - \lambda}{X^2} = \frac{-100 - 150}{50} = -5 \rightarrow \text{The line became even worse than before}$$



and hence after $m = 0$ we stop the algorithm.

Again: Ridge Reg. mei sparsity (coeff=0) nahi hota hain. waha lambda (λ) deno mei divide ho raha hota hai; upar num mei add ya subtract nahi.

Hence: λ kitna bhi bada kardo, coeff zero nahi hoga, kyun na kahi value ayga hi; Ridge mei coeff 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 f_n in Elastic Net Regression is:

$$L = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha ||w||^2 + \alpha ||w||_1$$

in Scikit Learn we get two hyper parameters:-

$$\lambda = \alpha + b \quad \& \quad \text{a ratio} = \frac{b}{\alpha + b}$$

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.

$\therefore$ The Scikit learn with $(\lambda, l1_ratio) \rightarrow$ is used in
in elastic net Reg; kitha % Ridge Lagaye & kitha %
Lasso Lagaye.

★ Apart from big Data sets; when the input col has multi-
collinearity hota hai (input cols apas mei bahot jyada depend krte hai)
then we use Elastic net Regression.

Jupyter Notebook

Ex: $\frac{\text{weight}}{\text{height}}$

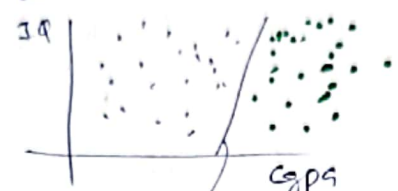
Logistic Regression

There are two perceptions of Logistic Regression

- ⊥ Geometric (Learn yourself).
- ⊥ probabilistic. (we learn this)

On what type of dataset should we apply Logistic Regression?

↳ dataset which are linearly separable or almost



- No placement
- Yes placement

Linearly Separable
(i.e. ex do
data points
Yaha waha
ho nah
to hai)

in 3-D its a plane &
in higher dim its a
hyperplane.

Perceptron Trick

Consider the same above dataset & we have to predict for a given new child ki uske placement hua ya 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.

$$Ax_1 + Bx_2 + C = 0$$

CGPA → 3D

Hence to find A, B & C.

In Perceptron Trick:

we start with a Random line; random value of A, B & C

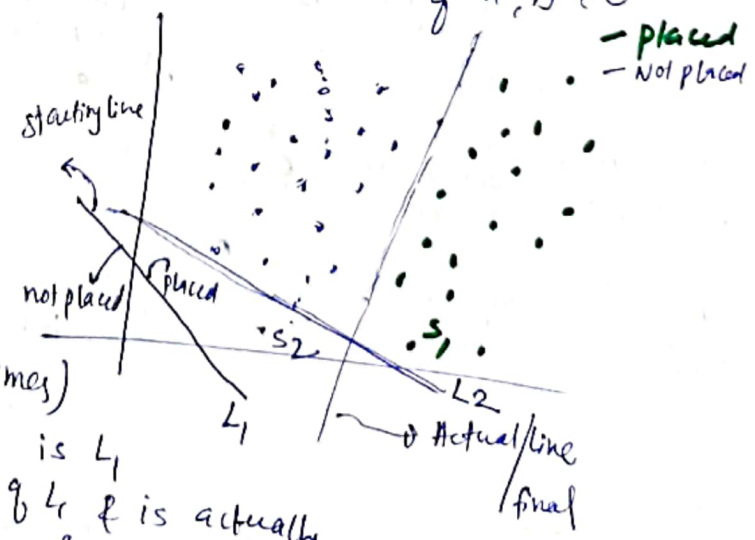
say $A = 1, B = 1, C = 0$

& for this line we consider one side all placed & other side not placed

(see diag) so accordingly to this line everyone is placed.

Now we run a loop (mostly 1000 times) we take a point say S_1 & ask is L_1

good & S_1 is on placed side of L_1 & is actually placed & S_1 say L_1 is ok. so for S_1 we don't change the value of A, B, C; Again in 2nd iteration of loop we pick S_2



ask S_2 : is L_1 OK $\rightarrow$ it says well i am not placed but L_1 say I am placed so L_1 is not OK for me. so we now make transformations in A, B, C & change b_1 in such a way that S_2 becomes not placed. (see diag)
 & 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. (ie No. of misclassified points = 0)

1st Qn: 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:- (cost a point $\rightarrow$ jis pt ke taraf move karana hai)

Line is:- $Ax + By + C = 0$

change in $C \rightarrow$ parallel shifts

change in $A \rightarrow$ rotation about y-intercept

change in $B \rightarrow$ Rotation about x-intercept.

$\uparrow$ or $\downarrow$ C we shift line up or down

Jab line ko pos direction mei move karana hai to C ko subtract karo.

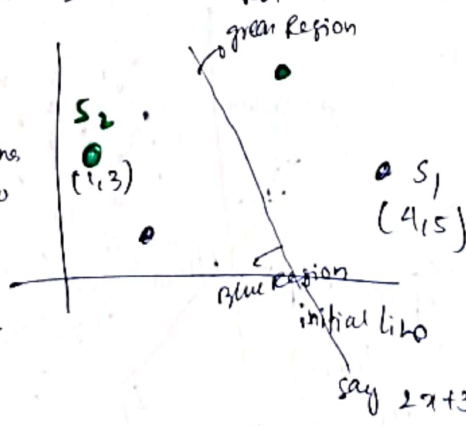
Consider.

Jab line ko -ve direction mei move karana hai to C ko add karo

Again. S_2 says the initial line is faulty:- so transforming:-

$$\begin{array}{r} 2 \ 3 \ 5 \\ + \ 1 \ 3 \ 1 \\ \hline 3x + 6y + 7 = 0 \end{array}$$

$\rightarrow$ for this line S_2 will be on pos/green side.



S_1 says line is not good so we move line towards

$$\begin{array}{r} S_1 \rightarrow (4, 5, 1) \\ 2 \ 3 \ 5 \\ - \ 4 \ 5 \ 1 \\ \hline -2x - 2y + 4 = 0 \end{array}$$

$\rightarrow$ for this line S_1 will be in blue region (-ve region)

Thus Summary of transformation is:-

Agar apka point (x, y) -ve side mei hai then we need to move the line to -ve side to make (x, y) on pos side for that new line:-

$$\begin{matrix} A & B & C \\ + & x, & y, & 1 \end{matrix}$$

if (x_1, y_1) -ve point is on pos side then the line to pos side to make the point -ve $\rightarrow$ Transformation is:-

$$\begin{matrix} A & B & C \\ - & x, & y, & 1 \end{matrix}$$

$$(A-x_1)x + (B-y_1)y + (C-1) = 0$$

But these transformations are huge hence we use $\eta = \text{learning rate}$ generally & transformations then become:-

$$\pm \eta x_1 \quad \eta y_1 \quad \eta$$

$$(A \pm \eta x_1) + (B \pm \eta y_1) + (C \pm \eta) = 0$$

with this the change/transformation is not abrupt $\rightarrow$ but it still does the job.

Algorithm for perceptron trick:-

cgpa	IQ	placed
7.5	81	1
8.9	90.9	1
7.0	81	0

The eqn of line will be:- $Ax + by + c = 0$

for this data:- $w_0 + w_1 x_1 + w_2 x_2 = 0$

$$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0 \quad (\text{assuming the is a col of all ones})$$

$$\Rightarrow \sum_{i=0}^2 w_i x_i = 0$$

General eqn of line.

This is our model.

$$w_0 x_1 + w_1 x 7.5 + w_2 x 81$$

≥ 0 means placed
 < 0 means not placed.

How does this make prediction:-
say for S_1 : (we have w_0, w_1, w_2)

$(w_{\text{eff}} = (w_0, w_1, w_2)) \cdot (1, 7.5, 81) = (1, \text{cgpa}, \text{IQ}) = (1, x_1, x_2)$

$\Downarrow$ dot product

$[w_0, w_1, w_2] \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$

$\rightarrow \geq 0$ placed
 $\rightarrow < 0$ not placed

Algorithm is:

epochs = 1000 generally ; $\eta = 0.01$ (Generally)

for i in range (epochs):

randomly select a student

if $x_i \in \text{Negative Region}$ and $\sum_{i=0}^2 w_i x_i \geq 0$ $\rightarrow y_i = 0$

update coeff.

$$w_{\text{new}} = w_{\text{old}} - \eta x_i$$

$$= [w_0, w_1, w_2] - \eta [x_0, x_1, x_2]$$

This is 1

placement nahi
hues but
model ne bola
ho gya

i.e. student
actually belongs
to +ve Region
but model
(current line) is
predicting
-ve region
we need to move
to the direction
i.e. we need to
subtract the
current coeff with
(coord of point, 1).
for transformation

placement
hues but
model ne bola
nahi hue.

if $x_i \in \text{pos Region}$ and $\sum_{i=0}^2 w_i x_i \leq 0$ $\rightarrow y_i = 1$

$$w_{\text{new}} = w_{\text{old}} + \eta x_i$$

$$= [w_0, w_1, w_2] + \eta [x_0, x_1, x_2]$$

Simplifying the Algo

for i in range (epochs)

select a random student:

$$\text{update: } w_{\text{new}} = w_{\text{old}} + \eta (y_i - \hat{y}_i) x_i$$

why it works.

Actual
output y_i

predicted
output $\hat{y}_i$

$y_i - \hat{y}_i$

1
0
1
0
1

1
0
0
1
1

0
0
1
-1
0

$$0 \rightarrow w_{\text{new}} = w_{\text{old}}$$

$$0 \rightarrow w_{\text{new}} = w_{\text{old}}$$

$$1 \rightarrow w_{\text{new}} = w_{\text{old}} + \eta x_i$$

$$-1 \rightarrow w_{\text{new}} = w_{\text{old}} - \eta x_i$$

actual placement hues
& model ne bhi bola
placement hues.
placement nahi hues &
model ne bhi bola
nahi hues.
placement hues but
model ne bola nahi
hues.
(see above
algo)
placement nahi hues
but model ne bola
ho gya.

Jyoti Note Book.

So far we had perceptron trick :-

using $W_{t+1} = W_t + (y_i - \hat{y}) x_i$, 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.

↓ changing this approach.

Now both correctly classified & misclassified points will transform the line

ye points line ko apne pass bulayenge (as always).

ye line ko apne & durs bhejenge

isse kya hoga ki :- jab same points

classify hojenge & line bias hai then good points line ko durs bhejenge & agar jyada durs bhejdiya toh durs wala class ke points phir is durs bhejenge (∵ 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.

Agar point misclassified hai then the pull of the point on the line is proportional to the distance b/w the pt & line
jyada distance → jyada pull
kam distance → kam pull.

for correctly classified points :- then the push of the point on the line is ∝ proportional to the dist. b/w pt & line
intensely

jyada distance → kam push
kam distance → jyada push.

↓ This is logical also.

This is the change that Skit 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.

hamme jh dono cases
 mei thi change karne
 hai
 i.e. in cases mei
 $W_{new} = W_{old}$ nahi

classified points
 yaha change
 nahi ho rhe the
 in perceptron
 trick
 i.e. $W_{new} = W_{old}$

y_i	$\hat{y}_i$
1	1
0	0
1	0
0	1

yaha
 changes
 ho rhe the
 (mis classified
 points)

chahiye i.e.
 $\eta (y_i - \hat{y}_i) x_i$ ko zero
 nahi hone dena hai

→ $y_i - \hat{y}_i \neq 0$ hona chahiye

pehle ham predicted
 placed & not placed ko
 0, 1 & represent kar rhe
 the ab kisi aur & rep.
 same chahiye.

This is data
 Give (cannot
 change)

$\hat{y}_i$ is the predicted value of the
 model ($w_1 x_1 + w_2 x_2 + w_0 = 0$)

if $w_1 x_1 + w_2 x_2 + w_0 \geq 0$ → then
 placed (1)
 else not placed
 aise hi $\hat{y}_i$
 (0) nikalta hai.

if $\hat{y}_i$ hai 0, 1 value lega toh
 hence target is ki $\hat{y}_i$ 0, 1 ke alawa kuch aur value

$y_i - \hat{y}_i = 0$ hoga hi

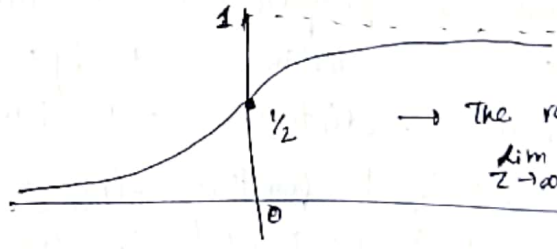
le. $\hat{y}_i$ ka calculation mechanism change karna hoga.

→ in the perceptron 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.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



→ The range is $(0, 1)$
 $\lim_{z \rightarrow \infty} \sigma(z) = 1$
 $\lim_{z \rightarrow -\infty} \sigma(z) = 0$

Domain $(-\infty, \infty)$

Now to calculate $y_{hat} = \hat{y}_i$ if $w_1 x_1 + w_2 x_2 + w_0 > 0 \rightarrow 1$
 Now we replace the step fn with Sigmoid fn
 $< 0 \rightarrow 0$

$\hat{y}_i = \text{sigmoid}(w_1 x_1 + w_2 x_2 + w_0)$
 $T > 0 \rightarrow \hat{y}_i > 0.5$
 $T < 0 \rightarrow \hat{y}_i < 0.5$

So given input :- $\{7.5, 81\} \rightarrow$ placement hogi ya nahi?

Page 62.

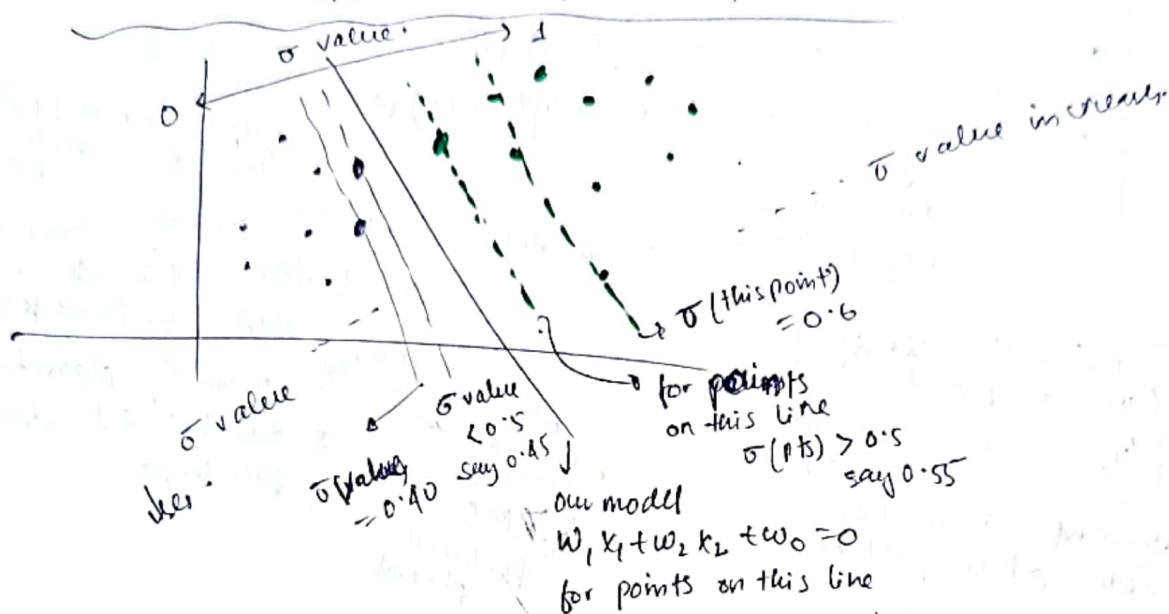
then $T = x_1 w_1 + x_2 w_2 + w_0$

$T = w_1 (7.5) + w_2 (81) + w_0$

then $\sigma(T) > 0.5 \rightarrow$ placed

then $\sigma(T) \leq 0.5 \rightarrow$ not placed.

Let's see how the sigmoid fn works.



So now what the sigmoid fn represent is probability of placed.

on a line say σ value = 0.5 means $P(\text{placed of pts. on that line}) = 0.5$

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 $P(\text{Nahi hogi}) = 1 - P(\text{placed})$

Hence ab ham student ka placement hone ka & nahi hone ka dono ka prob hai using the Sigmoid instead of step fn.

& using this probabilistic interpretation we will make changes in our formula to get the desired result.

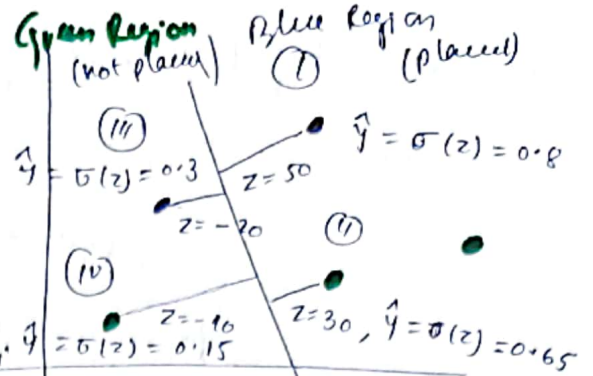
$$W_1 = W_0 + \eta (y_i - \hat{y}_i) x_i$$

$$\hat{y}_i = \sigma(z), \quad z = \sum_{i=0}^2 w_i x_i \quad \& \quad x_0 = 1$$

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 $\rightarrow w_n = w_0 + \eta (-0.65) X_i$
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 fn)



$y_i = 1 \rightarrow$ actual placed
 $\hat{y}_i = 0.8 \rightarrow$ predicted placed

1st case: correctly classified & far toh dhene $\hat{z}_i$ push kiya (add kiya)

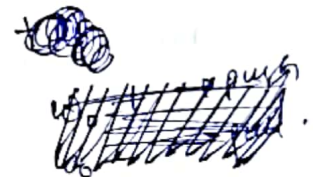
2nd case: incorrectly classified & dur hai toh strongly pull kiya

3rd case: incorrectly classified
 $y_i = 1 \rightarrow$ placed
 $\hat{y}_i = 0.3$ (0.5 $\rightarrow$ predicted not placed)

and same hai toh dhene pull kiya

$y_i = 0$ (Actual not placed)
 $\hat{y}_i = 0.5$ predicted placed.

4th case: correctly classified
 $y_i = 0$ (Actual not placed)
 $\hat{y}_i = 0.15$ (predicted not placed)
 & dur hai toh dhene $\hat{z}_i$ push kiya.



Case I: $w_n = w_0 + \eta \times 0.2 \times X_i =$ add kar rhe hai toh niche bhej rhe hai line $\rightarrow$ hence (I) ne push kiya line

Case II: $w_n = w_0 + \eta \times (-0.65) \times X_i =$ subtract $\rightarrow$ upar bhej rhe hai matlab pull kiya line ko (II) ne

Case III: $w_n = w_0 + \eta \times (0.3) \times X_i \rightarrow$ add $\rightarrow$ niche bhej rhe hai matlab (III) pull kar rhe hai line ko apne farq

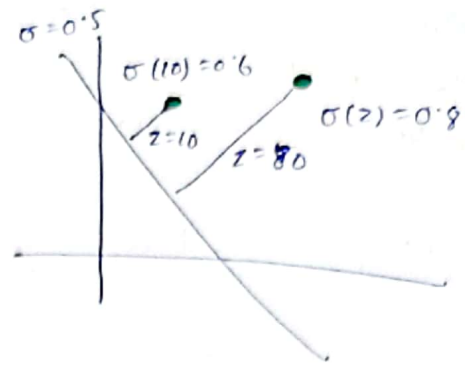
Case IV: $w_n = w_0 - \eta \times 0.15 \times X_i \rightarrow$ subtract $\rightarrow$ line ko upar bhejne matlab line ko push kar rhe hai (IV)

$\therefore$ Misclassified pt $\rightarrow$ pull karrahe hai
 & correctly classified pts $\rightarrow$ push kar rahe hai line ko.

what effect
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	0.4	$w_0 + \eta \times 0.4 \times x_i$
②	1	0.8	0.2	$w_0 + \eta \times 0.2 \times x_i$
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 line ka wo jyada push kar rhe hai

Case II: add kiya toh line niche jaygi & kam add kiya compared to ① toh kam niche jaygi compared to ① & hence lower point jo dur hai 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 rahe hai & har iteration mei ek random point lekar pate kar rahe hai if mis classified line ko apni taraf pull karo nahi to push karo → finally jo answer aya wo hum pure conviction ke saath nahi bol sakte ki sahi hai. Since agar hum dusri baar code run kare toh 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 → This is not how ML works → Every time we run the code for the same data set the line must be same.

So our Algo is only finding a SOL^n but it doesnot guarantee that the SOL^n obtained is the best SOL^n .
 Since next time we run it, it may give ~~the~~ better SOL^n , but again we have no guarantee it is the best one.

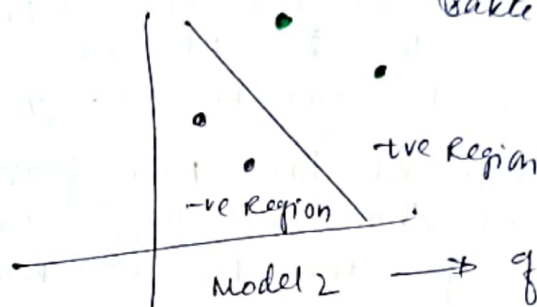
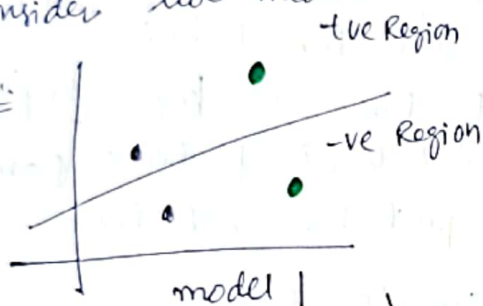
So using the ML way:- finding the Loss fn or Error fn

Finding the Loss fn

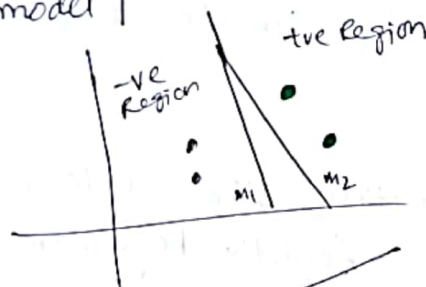
Maximum Likelihood:-

Consider two models:-

1st On:-



Next On:-



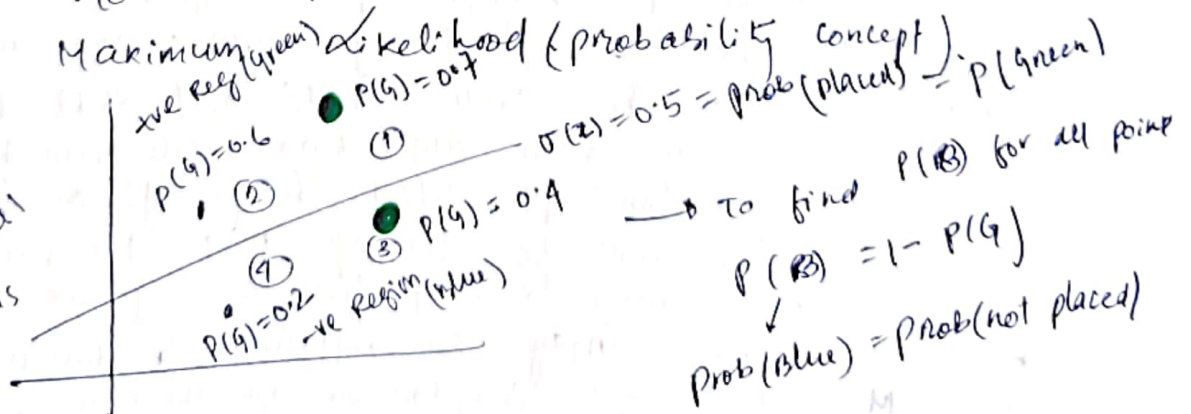
Here Loss fn will help & tell which one is better. & using it we can fetch the best model.

Now it asked which model is better M_1 or M_2 → difficult to say

Consider the models in OH1.

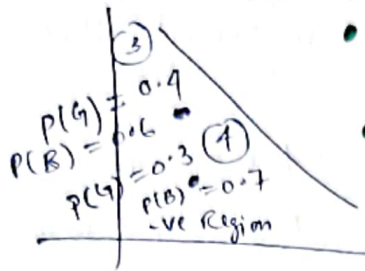
Acc to this model.

Maximum Likelihood (probability concept)



Acc to model 2.

4ve Region (1)



- $P(1) = 0.7$
 $P(2) = 0.3$
- $P(3) = 0.6$
 $P(4) = 0.4$

Page 64.

$$y = \sigma(z) = \text{prob of that point to be placed}$$

$$z = \sum w_i x_i$$

Now: Acc. to maximum likelihood says:
for a model multiply all the prob obtained & the model with the highest prob is better.
But: AAP Singh wo prob logo ek point ka jis color ka wo point hai.

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 = $0.7 \times 0.4 \times 0.4 \times 0.8 = 0.089$
 $= P(1) \times P(2) \times P(3) \times P(4)$

$\therefore$ maximum likelihood of model 2 $>$ max likelihood of model 1
 $\therefore$ Model 2 is better than model 1.

Hence humne ek aisa model chahiye jiska maximum likelihood max aaye.

The only problem is $\rightarrow$ product of small Nos.
 upon 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 vv small & hence the comparison will not be valid.

So we convert the product to sum
 i.e. the model whose $\log$ 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 $\therefore$ the sum will be -ve $\rightarrow$ so to avoid this

for model 1: $-\log(\text{max likelihood}) = -(\log(0.7) + \log(0.4) + \log(0.4) + \log(0.8))$

for model 2: $-\log(\text{max likelihood}) = -(\log(0.7) + \log(0.6) + \log(0.6) + \log(0.3))$

$\downarrow$
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 $\uparrow$ for maximize karna
so $\log(\text{max likelihood})$ ko bhi maximize karna
hoga but $-\log(\text{max likelihood}) = \text{cross entropy}$
ko minimize karna hoga for best model selection.

$\therefore$ formula of cross entropy $\rightarrow$ (note humne wo prob lena chota 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.

$\&$ for ~~blue~~ ^{blue} pts. cross entropy = $-\log(\hat{y}_i)$

$\&$ for ~~not placed~~ ^{not placed} pts. cross entropy = $-\log(1-\hat{y}_i)$

$$P(\text{placed}) = P(\text{Green}) = \hat{y}_i$$

$\therefore$ formula direct

$$-\log(\hat{y}_1) - \log(\hat{y}_2) - \dots - \log(\hat{y}_n)$$

nahi likh sakte since ho sakta hai kahi humne $P(\text{Blue}) = 1 - \hat{y}_i$ liya ho.

$\therefore$ formula should consider both.

$\therefore$ Loss fun := $\sum_{i=1}^n [-y_i \log(\hat{y}_i) - (1-y_i) \log(1-\hat{y}_i)]$

$\&$ to find average error = $-\frac{1}{n} \sum_{i=1}^n y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)$

$\rightarrow$ This is called log loss error or Binary cross entropy function.

And Now we need to minimize this
fn.

page 65

ie we need to find w_0, w_1, w_2 such that:-

$$L = -\frac{1}{n} \sum_{i=1}^n y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i) \text{ is minimized.}$$

↓ This doesnot have any closed form
 solⁿ (ie 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.

Derivative of Sigmoid function:-

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

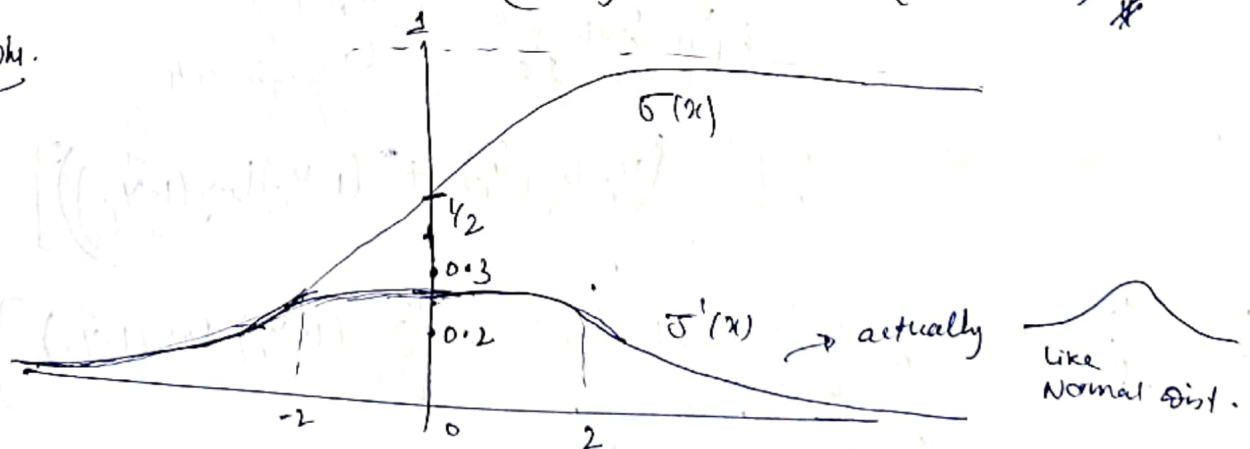
$$\sigma(x) = \frac{1}{1+e^{-x}} \rightarrow \sigma'(x) = \frac{d}{dx} \left(\frac{1}{1+e^{-x}} \right) = \frac{-[0+e^{-x}(-1)]}{(1+e^{-x})^2}$$

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}}{(1+e^{-x})}$$

$$\sigma'(x) = \sigma(x) \left[\frac{e^{-x}}{1+e^{-x}} \right] = \sigma(x) \left(1 - \frac{1}{1+e^{-x}} \right)$$

$$\sigma'(x) = \sigma(x) - (\sigma(x))^2 = \sigma(x) (1 - \sigma(x))$$

Graph.



Optimizing the Loss fn using gradient Descent

Consider a Data set with m rows & n input cols

1	2	3	...	n	output(y)	$\hat{y}$
x_{11}	x_{12}	x_{13}	...	x_{1n}	y_1	$\hat{y}_1 = (w_0 + w_1 x_{11} + \dots + w_n x_{1n})$
x_{21}	x_{22}	x_{23}	...	x_{2n}	y_2	$\hat{y}_2 = (w_0 + w_1 x_{21} + \dots + w_n x_{2n})$
$\vdots$						$\vdots$
x_{m1}	x_{m2}	x_{m3}	...	x_{mn}	y_m	$\hat{y}_m = (w_0 + w_1 x_{m1} + \dots + w_n x_{mn})$

As in Logistic Regression we need to find the coeff- $w_1, w_2, \dots, w_n$ & intercept- w_0

Total $(n+1)$ coeff

$$\hat{y}_i = \text{Sigmoid of } w_0 + w_1 x_{i1} + \dots + w_n x_{in}$$

$$= \sigma(w_0 + w_1 x_{i1} + \dots + w_n x_{in})$$

So: $\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} = \begin{bmatrix} \sigma(w_0 + w_1 x_{11} + \dots + w_n x_{1n}) \\ \sigma(w_0 + w_1 x_{21} + \dots + w_n x_{2n}) \\ \vdots \\ \sigma(w_0 + w_1 x_{m1} + \dots + w_n x_{mn}) \end{bmatrix}$

matrix of predicted values

This Sigmoid operation is on all terms

$$\hat{y} = \sigma \left(\begin{bmatrix} w_0 + w_1 x_{11} + \dots + w_n x_{1n} \\ \vdots \\ w_0 + w_1 x_{m1} + \dots + w_n x_{mn} \end{bmatrix} \right)$$

we take it out

$$\hat{y} = \sigma \left(\begin{bmatrix} 1 & x_{11} & \dots & x_{1n} \\ 1 & x_{21} & \dots & x_{2n} \\ \vdots & \vdots & & \vdots \\ 1 & x_{m1} & \dots & x_{mn} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \right) = \sigma(XW)$$

input data with 1st col of all 1's.

coefficients

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

~~Let's~~ $L = -\frac{1}{m} \left[\sum_{i=1}^m y_i \log(\hat{y}_i) + \sum_{i=1}^m (1-y_i) \log(1-\hat{y}_i) \right]$

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 ~~it is~~ ~~tricky~~ ka problem hai start se handle karo.

$$\text{Similarly, } \sum_{i=1}^m (1-y_i) \log(1-\hat{y}_i) = (I-Y) \log(\sigma(I-XW))$$

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 doesn't have any closed form $\therefore$ we have to use gradient descent to minimize it.

$$\text{where } \hat{Y} = \sigma(XW)$$

we initialize $W = [\dots]$ with random entries

& we decide epochs & then we update the W with formula:-

$$W_{\text{new}} = W_{\text{old}} - \eta \frac{\Delta L}{\Delta W} \quad \text{gradient of } L \text{ w.r.t } W, \text{ i.e. learning rate}$$

Finding $\frac{\Delta L}{\Delta W}$:-

$$L = -\frac{1}{m} \left[Y \log(\hat{Y}) + (I-Y) \log(I-\hat{Y}) \right]$$

$$\frac{dL}{dW} = -\frac{1}{m}$$

$$\left[\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_n} \right]$$

$$\frac{d}{dW} (Y \log \hat{Y}) = Y \frac{d}{dW} \log \hat{Y} = \frac{Y}{\hat{Y}} \frac{d}{dW} \hat{Y} = \frac{Y}{\hat{Y}} \frac{d}{dW} \sigma(XW)$$

$$= \frac{Y}{\hat{Y}} \sigma(XW) [1 - \sigma(XW)] \frac{d}{dW} XW$$

$$= \frac{Y}{\hat{Y}} \hat{Y} (I - \hat{Y}) X \quad \because \sigma(XW) = \hat{Y}$$

$$= Y (I - \hat{Y}) X$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \therefore \frac{d}{dx} \sigma(x) = \sigma(x)(1-\sigma(x))$$

Similarly,

$$\frac{d}{dW} (I-Y) \log(I-\hat{Y}) = (I-Y) \frac{d}{dW} \log(I-\hat{Y}) = \frac{I-Y}{I-\hat{Y}} [-\hat{Y} (I-\hat{Y})] X$$

$$= -(I-Y) \hat{Y} X = -\hat{Y} (I-Y) X$$

$$\begin{aligned}
 \text{Thus } \frac{dL}{dw} &= -\frac{1}{m} \left[y(1-y)x - \hat{y}(1-\hat{y})x \right] \\
 &= -\frac{1}{m} \left[y(1-y) - \hat{y}(1-\hat{y}) \right] x \\
 &= -\frac{1}{m} \left[y - y\hat{y} - \hat{y} + \hat{y}^2 \right] x \\
 &= -\frac{1}{m} \left[y - \hat{y} \right] x
 \end{aligned}$$

Do the derivative of loss without the matrix form to find gradient.

$$\text{Hence } W_{\text{new}} = W_{\text{old}} + \frac{\eta}{m} (y - \hat{y}) x$$

$$\begin{aligned}
 W &= \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix}_{(n+1) \times 1} ; X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}_{(m, n+1)} ; y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}_{m \times 1} \\
 \hat{y} &= \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_m \end{bmatrix}_{m \times 1}
 \end{aligned}$$

Jupyter Notebook.

We learn ~~the~~ Regression metrics in ~~the~~ Regression problem to find which model performs better on a Dataset. Similarly classification problems have classification metrics which guide us 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: cpa | Ia | placement

Actual Label	Log pred.	Dec. Tree pred
1	1	1
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

1000 students data → 800 train data
 → 200 test data
 we use two classification Algorithm:-

Logistic Reg.
 ↓ Train
 ↓ Test

Decision Tree
 ↓ Train
 ↓ Test

→ on is which model is perform best on the Test data.

To find which model is better :- for every Test data we find Log. Reg. ne galthi ki ya nahi & same for Decision Tree & then find kisme kam galthi 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, Versicol
0 1 2

Actual label	Log pred Reg.	Dec. Tree pred
0	0	0
0	0	0
0	0	0
2	2	2
0	0	0
2	2	2
0	0	0
2	2	2
1	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 thik hai)

→ It actually depends upon the data

→ 80% accuracy pe shi model ko deploy kar sakte hai & 99% pe shi nahi kar sakte.

Mean:- food order karne ya nahi wale model ka 80% accuracy hai. Toh shi deploy kar sakte hai bina agar wrong predict kiya toh kuch khas loss nahi hoga

par :- heart disease hai ya nahi wale model ko 99% pe

Isi deploy nahi kar sakte :- galti kiya toh patient marega → galti ka stake Lokot high hai.

The problem with accuracy score:

It gives a number matlab say acc. score = 0.9 matlab 10% galti kar rha hai model par acc. score ye nahi bata sha ki galti ka type kya hai.

say binary classification hai:-
Galti types

Actual \ Predicted	0	1
0		
1		

Actual \ Predicted	0	1
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 batayga ki konsa
Error kitne 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)

Sahi tha
Sahi predict
kiya
Actual
values

Prediction

	1	0
1	True Positive	False Negative
0	False Positive	True Negative

Type 2 Error
Sahi tha Galat
predict kar diya.

This is when
output (target) has
only two classes (Binary)

Type 1 Error
(Galat tha Sahi
predict kar diya)

galat tha
galat predict
kiya.

→ isse hum accuracy
score bhi nikal sakte
hai

Ex: Heart Disease.

Pred.	0	1
Actual. 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 nahi predict kiya
- 6 log ko heart disease tha & model ne nahi 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 $\rightarrow$ is called Type 1 Error
 & False negative $\rightarrow$ is called Type 2 Error.

Confusion Matrix for multiclass classification problem
 output mei more than 2 categories hai

Confusion matrix =

	Predicted		
	0	1	2
Actual 0	1	0	0
Actual 1	0	1	0
Actual 2	0	0	1

$$\text{slly the Accuracy score} = \frac{\# \text{ of correct pred}}{\text{total pred}}$$

are the Right/ Predictions.
 correct

in the MNIST Data set $\rightarrow$ there are 10 classes (0-9)
 & hence the Confusion Matrix will be 10x10 order.

When accuracy is misleading?

of elements/data

$\rightarrow$ Imbalanced dataset $\rightarrow$ i.e. the no. of categories are not equal/almost equal.

in such case accuracy gives bad result.

ex: In an Airport
 passenger hai ya
 Terrorist

This will be highly
 imbalanced data.

Total 100000 passenger

	0	1
Terrorist hai	0	1
Terrorist nahi hai	0	99999

Actual

$\rightarrow$ acc score = 99.999 %
 almost 100 %

1 ko miss kar gya par
 that 1 is dangerous
 & the stakes are high.

hence we study Precision & Recall & F1-Score

Consider the Ex.:

spam: 1 ; not spam: 0

(Email spam classifier)
predicted

Actual	Spam	predicted	
		sent to spam	Not sent to spam
Actual	spam	100	170 false neg.
	not spam	30 false pos	700

Model 1

Actual	Spam	predicted	
		sent to spam	Not sent to spam
Actual	spam	100	170 false neg.
	not spam	10 false pos	700

Model 2

Accuracy Score is Same for both.

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)
 $\therefore$ if more non spam are sent to spam $\rightarrow$ it is not good.
 Hence model with less False pos mistake is good.
 Hence Model 2 is good comparatively.
 & isse ko precision bola jata hai.

$\rightarrow$ It is what Proportion of predicted positives

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 = $\frac{100}{110} = \frac{10}{11} \rightarrow$ 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
No cancer	800	8000

model 1 false pos

	Det. cancer	Not det
Has cancer	1000	500
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. $\therefore$ False Neg is Better. $\downarrow$ This is called Recall.

Defn: Recall :- What proportional of Actual positive is correctly classified (is predicted positive)?

$$\therefore \text{for model 1 :- Recall} = \frac{1000}{1200} = \frac{10}{12}$$

$$\text{for model 2 :- Recall} = \frac{1000}{1500} = \frac{10}{15}$$

$$\text{formula: Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negatives}} = \frac{TP}{TP + FN}$$

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).

Here: Type 1 Error $\rightarrow$ cat ko dog bol diya
 Type 2 Error $\rightarrow$ dog ko cat bol diya $\rightarrow$ Which is dangerous as 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).

$\downarrow$ In this case we use F1-Score metric.

It is a combination of precision & Recall (actually a Harmonic mean of both).

$$F1\text{-score} = \frac{2PR}{P+R} = \frac{2 \text{ Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-Score $\rightarrow$ [0, 1]
 $\swarrow$ worst $\searrow$ best

A higher F1-Score is generally better.

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		
		Dog	cat	Rabbit
Actual	Dog	25	5	10
	cat	0	30	4
	Rabbit	4	10	20
predicted Total		29	45	34
		$\downarrow$ predicted no. of Dogs	$\downarrow$ predicted no. of cats	$\downarrow$ predicted no. of Rabbits.

actual Total

40 $\rightarrow$ Actual no. of dogs

34 $\rightarrow$ Actual no. of cat

34 $\rightarrow$ Actual no. of Rabbit

Harmonic mean
 ex aisa mean hai jo hamesha lower value ko side inclined rehta hai

Ex: $p=0, R=100$

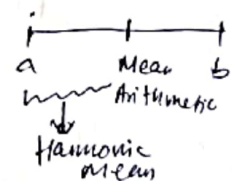
$$AV = \frac{p+R}{2} = 50$$

F1-Score = 0

a, b $a > b$

$$\text{Harmonic mean} = \frac{2}{\frac{1}{a} + \frac{1}{b}}$$

$\frac{1}{b} > \frac{1}{a} \therefore$ Harmonic mean shifts towards $\frac{1}{b}$



Calculation for precision = $\frac{\text{Actual pos}}{\text{total pred pos}}$

Page for

1st col → precision for dog class = $\frac{\# \text{ of Actual dogs}}{\text{total } \# \text{ of predicted dogs}} = \frac{25}{29} = 0.86$

2nd col → " " cat class = $\frac{\# \text{ of Actual cats}}{\text{total } \# \text{ of predicted cats}} = \frac{30}{45} = 0.66$

3rd col → " " rabbit class = $\frac{\# \text{ of Actual Rabbits}}{\text{total } \# \text{ of predicted Rabbits}} = \frac{20}{34} = 0.58$

Combined precision of the models

Macro precision = Avg. of all the precision (all col precision) = $\frac{0.86 + 0.66 + 0.58}{3} = 0.70$

Weighted precision = $\frac{\text{Sum of (class weight} \times \text{class precision)}}{\text{Actual pos / Total}} = (0.86 \times \frac{25}{108}) + (0.66 \times \frac{30}{108}) + (0.58 \times \frac{34}{108})$

Calculation for Recall = $\frac{\text{predicted pos}}{\text{total Actual pos}} = 0.71$

for Row 1 → Recall of dog class = $\frac{25}{40} = 0.62$

for Row 2 → Recall of cat class = $\frac{30}{34} = 0.88$

for Row 3 → Recall of rabbit class = $\frac{20}{34} = 0.58$

Combined Recall of the model

Macro Recall = Avg. of all Recall = $\frac{0.62 + 0.88 + 0.58}{3}$

Weighted Recall = $\frac{\text{Sum of (class weight} \times \text{class recall)}}{\text{predicted pos / total}} = (0.62 \times \frac{25}{108}) + (0.88 \times \frac{30}{108}) + (0.58 \times \frac{34}{108})$

Similarity Calculation of F1-Score

$$F1\text{-score of dog class} = \frac{2 P_p R_p}{P_p + R_p}$$

$$F1\text{-score of cat class} = \frac{2 P_c R_c}{P_c + R_c}$$

$$F1\text{-score of rabbit class} = \frac{2 P_r R_r}{P_r + R_r}$$

Again.

- Macro F_1 score = Av. of all $F1$ -score
- Weighted F_1 -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 Multinomial Logistic Regression

Consider dataset :-

Cgpa	IQ	placement
7.1	7.1	0
8.5	8.5	1
9.5	9.7	2
⋮	⋮	⋮

The classes here - Placed 1
 Not placed 0
 Opt out 2

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 :-

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$$
 ; $k = \# \text{ of classes in the output col.}$

placed.

$$\sigma(\vec{z})_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

↓
 prob of placed class

$$\sigma(\vec{z})_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

↓
 not placed
 ↓
 prob of not placed class

$$\sigma(\vec{z})_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

↓
 prob of opt out class.

Note :- $0 < \sigma < 1$

$$\sum_{i=1}^K \sigma(z)_i = 1$$

 $K = \# \text{ of class/categories in output}$

Softmax for two classes for output is sigmoid fn.

Training intuition of softmax Reg. (Just an example. Aisa exactly sikit learn mei nahi hota hai)

cgpa | 1 | Placement

0
1
2
...

one flat encoding of output col

$\begin{cases} y=0 \\ N \leq 1 \\ 0 \rightarrow 1 \end{cases}$

cgpa	1	0	1	2
...	...	...	...	...
...	...	1	0	0
...	...	0	1	0
...	...	0	0	1

class 1 placed, class 2 placed, class 3 opt out.

Now we have 5 cols.

Now we split it into three (No. of categories) Datasets

As there are three output cols & they are Binary classified.

D₁ class 1

cgpa	1	not placed
...	...	...
...	...	0

D₂

cgpa	1	placed = 1 : class 2
...	...	...
...	...	1

D₃

cgpa	1	opt out : 3 : class 3
...	...	...
...	...	1

Now we train this on three Logistic Reg model.

Model 1
 This will return 3 coeff (2° 2 input data)
 $w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$

Model 2
 3 coeff
 $w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$

Model 3
 3 coeff
 $w_1^{(3)}, w_2^{(3)}, w_0^{(3)}$

Actually sikit learn mei, there diffrent hota hai. (loss fn + Gradient descent use ho sha hai).

Now Training is done :- Let's focus on Predictions :-

M1

$w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$

↓ given a query point

This model predicts placement hua ya nahi

placement hua : pot class

placement nahi : Baaki do cases

0 : opt out ya placed.

M2

$w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$

↓ given a query point it predicts

1 : placement hua (pot class)

0 : Baaki dono class not placement hua ya opt out

M3

$w_1^{(3)}, w_2^{(3)}, w_0^{(3)}$

↓ given a query point it predicts :-

1 : opt out (pot class)

0 : Baaki dono class placed ya not placed.

Given a student :- $cgpa = 7$, $IQ = 70 \rightarrow$ To predict (placed, not placed)
M1 : placed opt out
M2 : placed opt out
M3 : placed opt out

$$w_1^{(0)}, w_2^{(0)}, w_0^{(0)}$$

$$w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$$

$$w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$$

$$Z_1 = 7w_1^{(0)} + 70w_2^{(0)} + w_0^{(0)}$$

$$Z_2 = 7w_1^{(1)} + 70w_2^{(1)} + w_0^{(1)}$$

$$Z_3 = 7w_1^{(2)} + 70w_2^{(2)} + w_0^{(2)}$$

70
 not side means not placed
 $-ve$ side means either placed or not placed

slightly in these two cases also.

Now we apply the softmax

$$\sigma(Z_1) = \frac{e^{Z_1}}{e^{Z_1} + e^{Z_2} + e^{Z_3}}$$

! Prob of Not placed
 $\sigma(\text{not placed})$ || 0.35 (say)

softmax

$$\sigma(Z_2) = \frac{e^{Z_2}}{e^{Z_1} + e^{Z_2} + e^{Z_3}}$$

! $\sigma(\text{placed})$
 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 opt out 0.25 (say)

Hence going by this logic Prob of placed is highest
 $\therefore$ the student will get placed.

$\hookrightarrow$ # of Logistic Reg = # of classes

Hence if there are more no. of classes ; there will be more no. of Logistic Regression to train & hence 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:-

Logistic Regression Loss fn =
$$-\frac{1}{m} \sum_{i=1}^m y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)$$

(with m rows/students)

Softmax

Reg / Multinomial Logistic Regression Loss fn =

$$-\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^K y_j^{(i)} \log(\hat{y}_j^{(i)})$$

for rows $\leftarrow$ $\sum_{j=1}^K$ has row relative # of classes ka summation hai.
 $\rightarrow m = \# \text{ rows}$
 $K = \# \text{ categories in output}$

Ex:

x_1	x_2	y
x_{11}	x_{12}	1
x_{21}	x_{22}	2
x_{31}	x_{32}	3

$\rightarrow$ 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

Model 1 for class 1

Loss fn =
$$-\frac{1}{3} \left[y_1^{(1)} \log(\hat{y}_1^{(1)}) + y_2^{(1)} \log(\hat{y}_2^{(1)}) + y_3^{(1)} \log(\hat{y}_3^{(1)}) \right. \\ \left. + y_1^{(2)} \log(\hat{y}_1^{(2)}) + y_2^{(2)} \log(\hat{y}_2^{(2)}) + y_3^{(2)} \log(\hat{y}_3^{(2)}) \right. \\ \left. + y_1^{(3)} \log(\hat{y}_1^{(3)}) + y_2^{(3)} \log(\hat{y}_2^{(3)}) + y_3^{(3)} \log(\hat{y}_3^{(3)}) \right]$$

 $m=3$
 $\rightarrow$ Model 1 for class 1 $i=1, K=1,2,3$
 $\rightarrow$ Model 2 for class 2 $i=2, K=1,2,3$
 $\rightarrow$ Model 3 for class 3 $i=3, K=1,2,3$

for the above data :-

Loss fn =
$$-\frac{1}{3} \left[y_1^{(1)} \log(\hat{y}_1^{(1)}) + 0 + 0 + 0 + y_2^{(2)} \log(\hat{y}_2^{(2)}) + 0 + 0 \right. \\ \left. + 0 + 0 + y_3^{(3)} \log(\hat{y}_3^{(3)}) \right]$$

$$= -\frac{1}{3} \left[y_1^{(1)} \log(\hat{y}_1^{(1)}) + y_2^{(2)} \log(\hat{y}_2^{(2)}) + y_3^{(3)} \log(\hat{y}_3^{(3)}) \right]$$

Now to find: $\hat{y}_1^{(1)}, \hat{y}_2^{(2)}, \hat{y}_3^{(3)}$:-

$$\hat{y}_1^{(1)} = \sigma(w_1^{(1)} x_{11} + w_2^{(1)} x_{12} + w_0^{(1)}) \rightarrow$$
 where $w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$ are weights/coefficient wrt col $y_{K=1}$
$$\hat{y}_2^{(2)} = \sigma(w_1^{(2)} x_{21} + w_2^{(2)} x_{22} + w_0^{(2)}) \rightarrow$$
 $w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$ are coeff wrt col $y_{K=2}$
$$\hat{y}_3^{(3)} = \sigma(w_1^{(3)} x_{31} + w_2^{(3)} x_{32} + w_0^{(3)}) \rightarrow$$
 $w_1^{(3)}, w_2^{(3)}, w_0^{(3)}$ are coeff wrt col $y_{K=3}$.

So the coeffs are:-
$$\begin{bmatrix} w_1^{(1)} & w_2^{(1)} & w_0^{(1)} \\ w_1^{(2)} & w_2^{(2)} & w_0^{(2)} \\ w_1^{(3)} & w_2^{(3)} & w_0^{(3)} \end{bmatrix} \rightarrow \text{are in the loss fn}$$

Hence to find the gradient of loss fn w.r.t them in gradient descent we need to calculate:

$$\frac{\partial L}{\partial w_1^{(1)}}, \frac{\partial L}{\partial w_2^{(1)}}, \frac{\partial L}{\partial w_0^{(1)}}, \frac{\partial L}{\partial w_1^{(2)}}, \frac{\partial L}{\partial w_2^{(2)}}, \frac{\partial L}{\partial w_0^{(2)}}, \frac{\partial L}{\partial w_1^{(3)}}, \frac{\partial L}{\partial w_2^{(3)}}, \frac{\partial L}{\partial w_0^{(3)}}$$

$\therefore$ In gradient descent $\rightarrow$ we initialize with random q values of the above matrix & then in epochs we update for each of q coefficients.

$$\begin{aligned} w_1^{(1)} &= w_1^{(1)} - \eta \frac{\partial L}{\partial w_1^{(1)}} \\ w_2^{(1)} &= w_2^{(1)} - \eta \frac{\partial L}{\partial w_2^{(1)}} \\ &\vdots \\ \text{and so on.} \end{aligned}$$

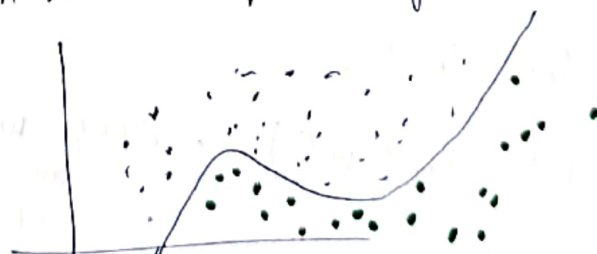
Jupyter Notebook.

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 cols) 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
$\vdots$	$\vdots$	$\vdots$								

& Now we

use Logistic Regression
Jupyter Notebook.

Hyperparameters in Logistic Regression

- Penalty: 'l1', 'l2', 'elasticnet', 'none': $\rightarrow$ Regularization
default 'l2'
- tol (default = $1e^{-9}$) $\rightarrow$ 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) $\rightarrow$ wo 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 = 'lbfgs'.
 $\rightarrow$ we made our own Logistic Reg. class using gradient descent sth 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
 $\rightarrow$ used when Solver = 'sag', 'saga' or 'liblinear' to shuffle data.
- max_iter: int; default = 100
 $\rightarrow$ # of epochs/iterations taken for the solver to converge.
- multi-class: {'auto', 'ovr', 'multinomial'}, default = 'auto'.
when the output has more than 2 categories.
ovr $\rightarrow$ for every class a logistic Reg model is made & coeff are found like we discussed
multinomial $\rightarrow$ Loss fn is minimized & coeff are found.

- warm-start :- default : False :-
 when set to True, jis point pe Training stop kar de ho, uske 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 = 'ovs'
 → means using all processors → multi processing.
- l1-ratio :- ($0 \leq \text{l1-ratio} \leq 1$) → only used if penalty = 'elasticnet'.
 l1-ratio = 0 → means penalty = 'l2'
 l1-ratio = 1 → means penalty = 'l1'.

DECISION TREES

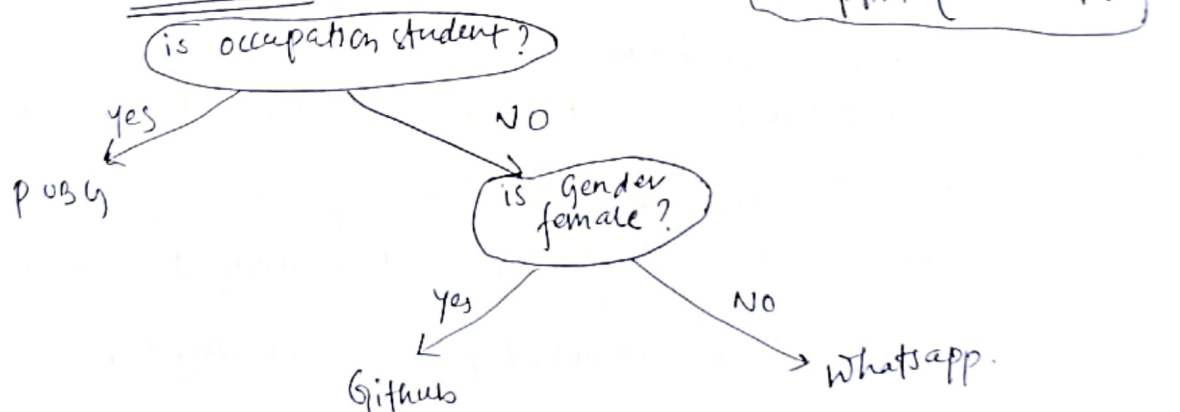
Consider the data:

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 :-



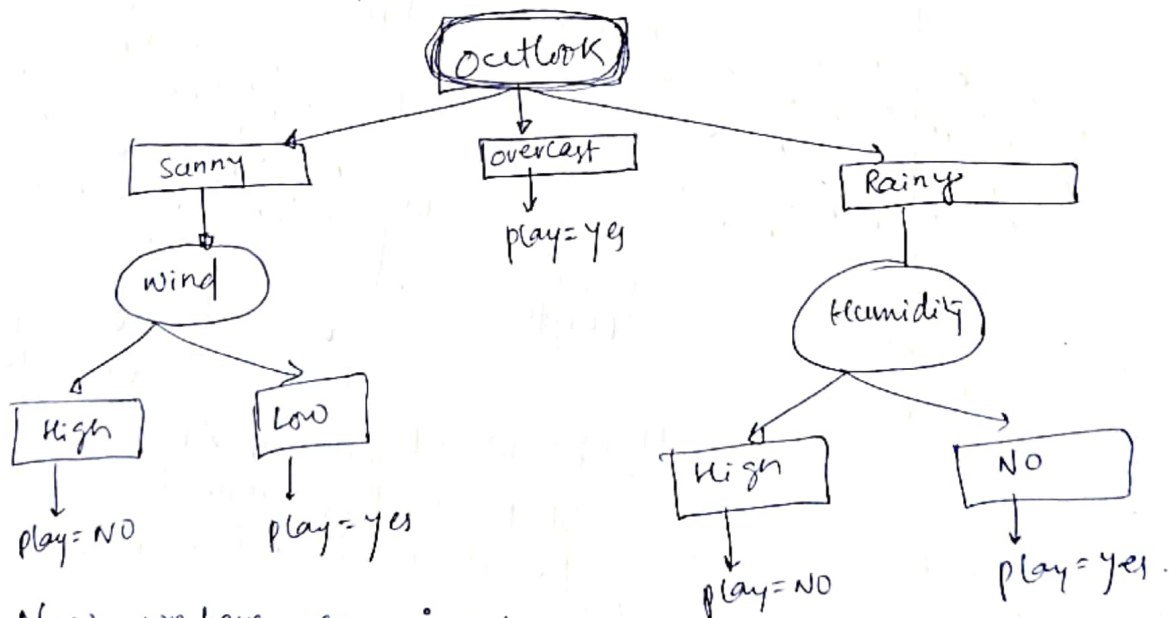
Consider the data:

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]

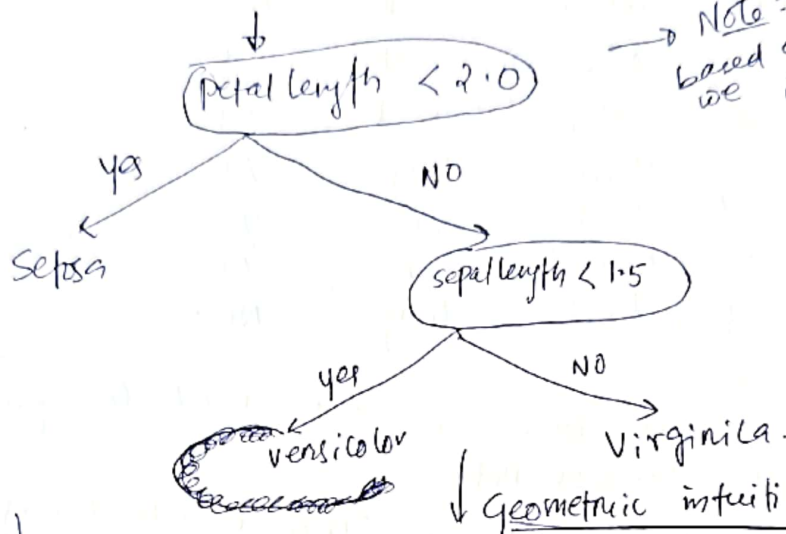
we go to Rainy & then check Humidity instead of Temp & here predict play = NO.

Here, we only checked outlook & Humidity even though there were other cond's. since our decision tree is made that way.

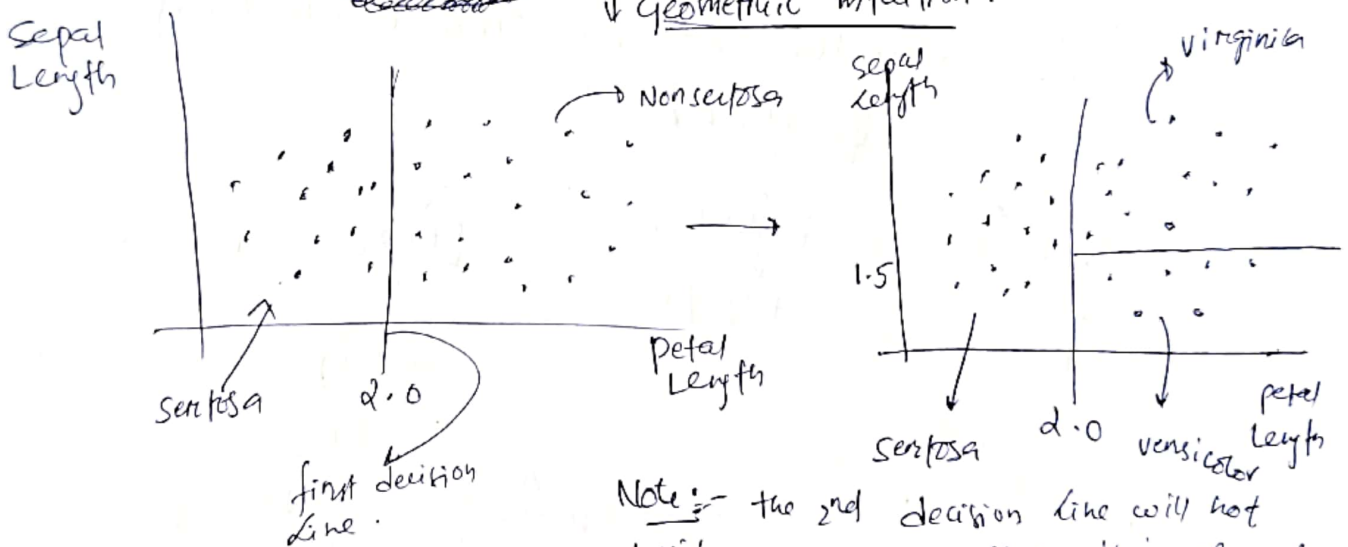
What if we have Numerical Data?

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	0.88	Setosa.

→ Note: first we classify based on one col & then we include other cols.



Geometric intuition:



Note:- the 2nd decision line will not classify the setosa, since it is already classified — it classifies the remaining part.

Notes - The lines we make are parallel to coordinate axes.

↓ In higher Dimensions the Lines will be planes & hyperplanes.

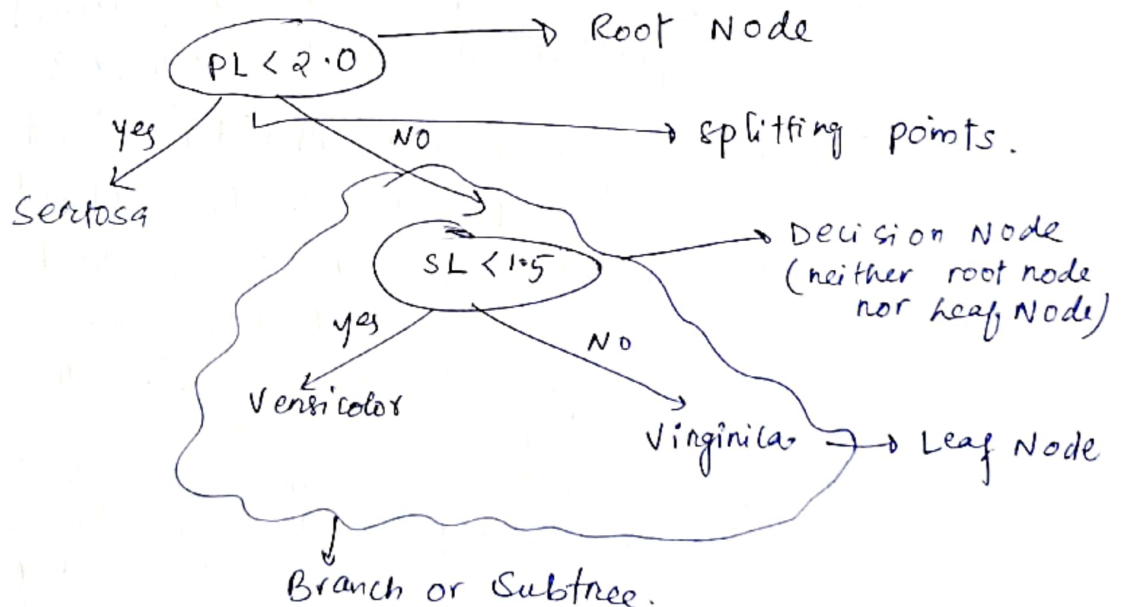
Pseudo code:- for decision Tree

Page 75.

- Begin ^{with} your training dataset, which should have some feature variables and classification or regression output.
- Determine the "best feature/cor" in the dataset to split the data on (we will see how we define best feature)
- split the data into subsets that contains the lowest 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.
- ^{No} we choose the next best feature
- 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/cats)
- 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
 $O(\log(\# \text{ of data points in } X_{\text{train}}))$

* Decision Tree is also called CART → classification and Regression trees.

Since it is mostly used for classification problem & but the logic of decision trees can be applied to regression problem 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 sign 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

more knowledge
more certain
↑
related to knowledge

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.

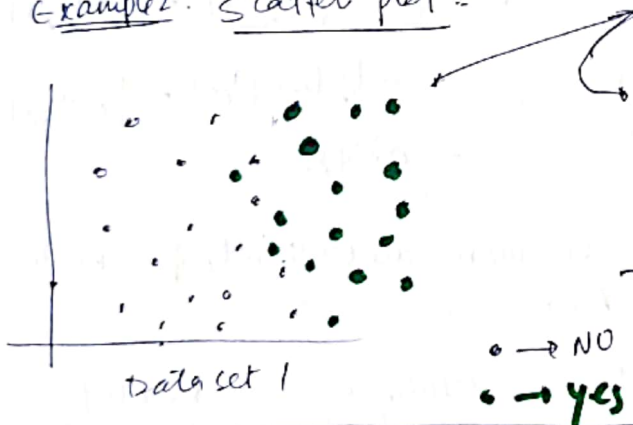
molecules are tightly packed

the molecules are 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 jisme hum more confidently predict kar sakte hai 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 eg: if our data has only two class labels Yes & No,
 $\hookrightarrow$ in the output data

$$E(D) = -P_{yes} \log_2(P_{yes}) - P_{no} \log_2(P_{no})$$

Ex3. Consider two datasets:

D1

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	NO
12000	18	NO

It has two classes:-

Yes & No.

$$P_Y = 2/5, P_N = 3/5$$

$$\therefore E(D1) = -P_Y \log_2(P_Y) - P_N \log_2(P_N)$$

$$E(D1) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$$

$$E(D1) = 0.97$$

$\hookrightarrow$ Logit has higher entropy & more uncertainty & hence less certainty than D2.

D2

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

$$P_Y = 1/5; P_N = 4/5$$

$$E(D2) = -P_Y \log_2(P_Y) - P_N \log_2(P_N)$$

$$= -1/5 \log_2(1/5) - 4/5 \log_2(4/5)$$

$$= 0.72$$

D3 all entries in output col = No (then we will be more certain & hence it should have least entropy)

$$E(D3) = -P_{yes} \log_2(P_{yes}) - P_N \log_2(P_N)$$

$$= -0 \log_2(0) - 1 \log_2(1)$$

$$= 0$$

* Calculating Entropy of a 3 class

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

Problem:

classes: Yes, No, Maybe

$$E(\text{Dataset}) = -P_Y \log_2(P_Y) - P_N \log_2(P_N) - P_M \log_2(P_M)$$

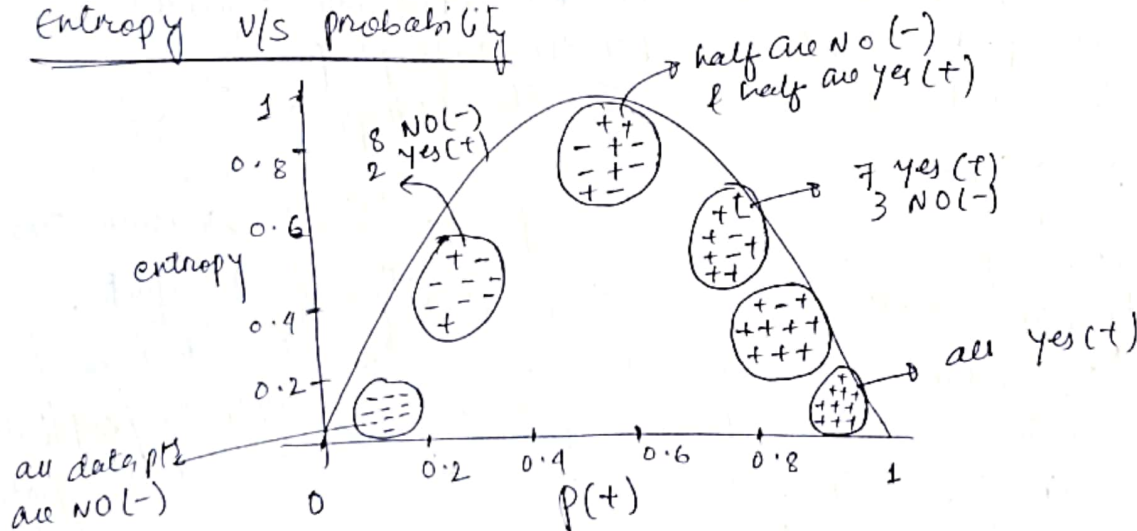
$$P_Y = 2/8, P_N = 3/8, P_M = 3/8$$

$$\therefore E(\text{Dataset}) = 1.56$$

Observations: • more the uncertainty more is the Entropy & 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 $\because$ we only want to know ki ki's dataset mei Entropy/uncertainty jyada hai.

Entropy v/s probability



Hence Entropy is highest if $P(\text{class}) = 0.5$

or, if all the class has equal probabilities then the Entropy is the highest (which is obvious also $\rightarrow \because$ then we are least certain about a given datapoint that it is of a particular class.)

Entropy for Continuous Variables:-

Dataset 1.

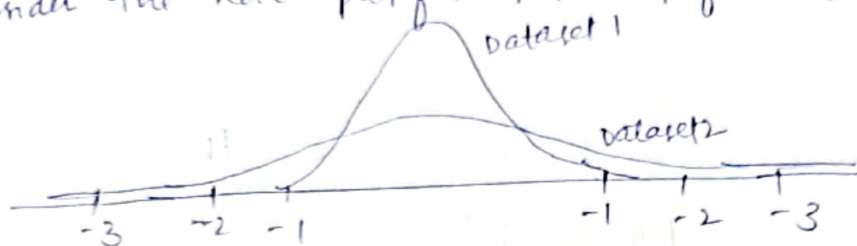
Area	Built in	Price
1200	1999	3.5
1800	2011	5.6
1400	2000	7.3

→ output col is Numerical

Dataset 2.

Area	Built in	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 Dataset.



we know jis bhi Dataset ke baare mei jyada knowledge
hoga uske Entropy kam hoga. → kam area mei jyada point

Dataset 1 ke jyada tar points $[-1, 1]$ $\hat{a}$ hai $\&$ hence we have
Dataset 2 ke jyada tar points $[-3, 3]$ $\hat{a}$ hai $\&$ hence we have
~~more~~ knowledge about Dataset 2 $\because$ Datapoint spread hai

Hence Dataset 1 has lesser Entropy compared to Dataset 2.

Hence for Cont's variable :- plot the KDE (histogram $\rightarrow$ smooth)
Higher the peak of KDE $\rightarrow$ Less the Entropy
Smaller the peak of KDE $\rightarrow$ 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:-

$$\text{Info Gain} = E(\text{parent}) - \{ \text{weighted Average } \} * E(\text{children})$$

$\downarrow$
 complete dataset
 before split.

$\downarrow$
 after split the
 weighted average of Entropy of all
 the children

Consider the Dataset with cols: outlook, Temp, Humidity, wind, play..
Let's calculate the info gain for the outlook col

Step 1: Entropy of parent

$$E(P) = -P_y \log_2(P_y) - P_n \log_2(P_n) = \frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right)$$

Step 2: we'll divide the dataset based on the outlook col:- (Group by outlook col)

Sunny					overcast					Rain				
Day	outlook	Humidity	wind	play	D	o	H	w	P	D	o	H	w	P
1	Sunny	High	weak	No	3	overcast	High	weak	Yes	4	Rain	High	weak	Yes
2	"	High	strong	No	7	"	Normal	strong	Yes	5	"	Normal	"	Yes
8	"	High	weak	No	12	"	High	strong	Yes	6	"	"	Strong	No
9	"	Normal	weak	Yes	13	"	Normal	weak	Yes	10	"	"	weak	Yes
11	"	Normal	strong	Yes						14	"	High	Strong	No

Calculate the entropy of all these children datasets.

$$E(\text{Sunny}) = 0.97$$

$$P_y = 2/5; P_n = 3/5$$

$$E(\text{overcast}) = 0$$

$$P_y = 1; P_n = 0$$

$$E(\text{Rain}) = 0.97$$

$$P_y = 3/5; P_n = 2/5$$

Step 3: Calculated weighted Entropy of children

$$\begin{aligned} \text{weighted entropy} &= \text{Prob}(\text{Sunny}) \times E(\text{Sunny}) + \text{P}(\text{overcast}) \times E(\text{overcast}) + \text{P}(\text{Rain}) \times E(\text{Rain}) \\ &= \frac{5}{14} \times 0.97 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.97 \end{aligned}$$

$$W.E. (\text{Children}) = 0.69$$

Note: Overcast is a leaf node as its Entropy is Zero(0) → this data will have no more splitting.

Step 4: Calculate Information Gain:-

$$\begin{aligned} \text{Info Gain for outlook column} &= E(\text{parent}) - W.E. (\text{children}) = 0.97 - 0.69 \\ &= 0.28 \end{aligned}$$

So the info gain (or decrease in Entropy / impurity) when you split this data on the basis of outlook condition/col is 0.28

Steps: 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.

Step 5: Find info gain Recursively. 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 (Entropy = 0); no more splitting is done on that particular dataset.

Gini Impurity
parameter called

In sklearn Decision Tree Classifier 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:-

$$\text{Entropy} = -P_y \log_2(P_y) - P_n \log_2(P_n)$$

$$GI = 1 - (P_y^2 + P_n^2)$$

Ex:- we have the same Salary | Age | Purchase with two Datasets.

for Dataset 1:-

$$GI = 1 - (P_y^2 + P_n^2) = 1 - (4/25 + 9/25) = 0.48$$

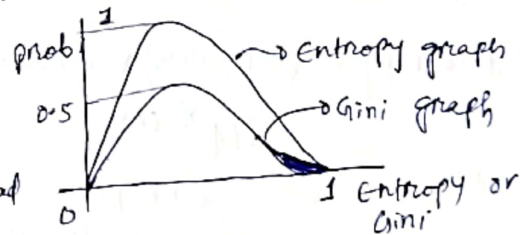
for Dataset 2:- $GI = 1 - (P_y^2 + P_n^2) = 1 - (1/25 + 16/25) = 0.32$

Hence Dataset 2 is more pure than Dataset 1.

Note:- jo kaam hum Info Gain ke liye Entropy ke through karte hai wohi same kaam hum Gini impurity ke through kar sakte hai. The formula will be Info Gain = GI(parent) - Weighted GI(children)

if: $P_y = 1, P_n = 0 \rightarrow E(D) = 0, GI(D) = 0$ & if $P_y = 0.5, P_n = 0.5 \rightarrow E(D) = 1, GI(D) = 0.5$

∴ 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 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 (12) subtrees & hence calculation will be difficult. Solution:-

S.No.	UserRating	Downloaded
1	3.5	Y
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	Y

Step 1:- Sort the data on the basis of the numerical col. (Here user-rating) → 1.6, 1.9, 2.2, 2.5, 2.9, 3.2, 3.3, 3.5, 3.9, 4.1, 4.6, 4.8

Step 2:- Split the entire data on the basis of every value of user-rating

ratings > 1.6 or rating ≤ 1.6 again, ratings > 1.9 or rating ≤ 1.9 & soon for all the rows after sorting.

one data | rest all data
[4 | 1.6 | Y]

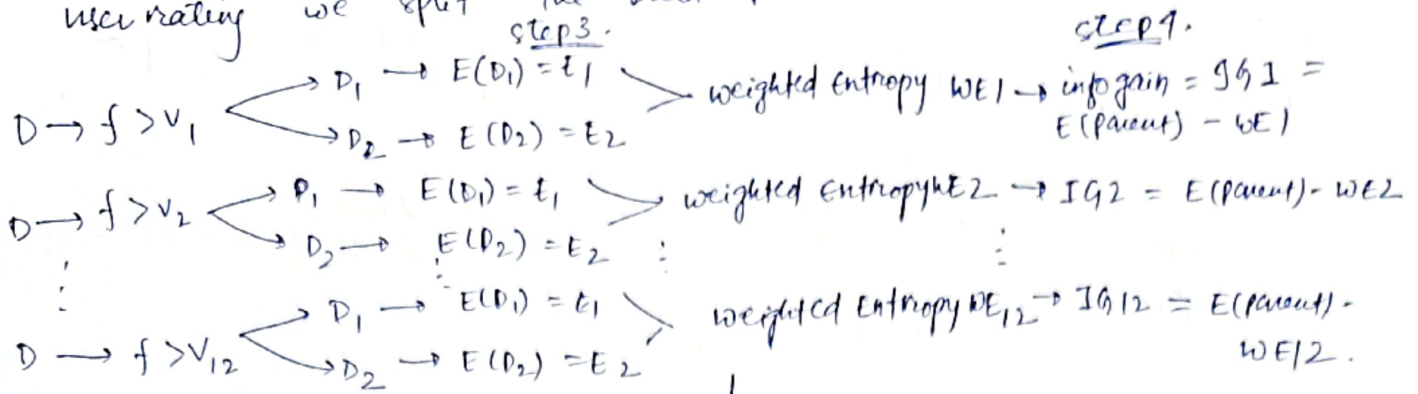
two data | rest all data
[4 | 1.6 | Y
12 | 1.9 | Y]

underfitting :-

page 8/

page 79.

So we have :- for every particular value of user rating we split the data set.



Where :- $D \rightarrow$ dataset
 f is feature/col user-rating
& v_i 's are the values of the various rows of user-rating.

after this :-

step 5. Max $\{IG1, IG2, \dots, IG_{12}\}$

say $IG3$ is the max

then $f > v_2$ will be the splitting criteria for col user-rating.

step 6: Do this recursively until you get all the leaf nodes.



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. $\therefore$ 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.

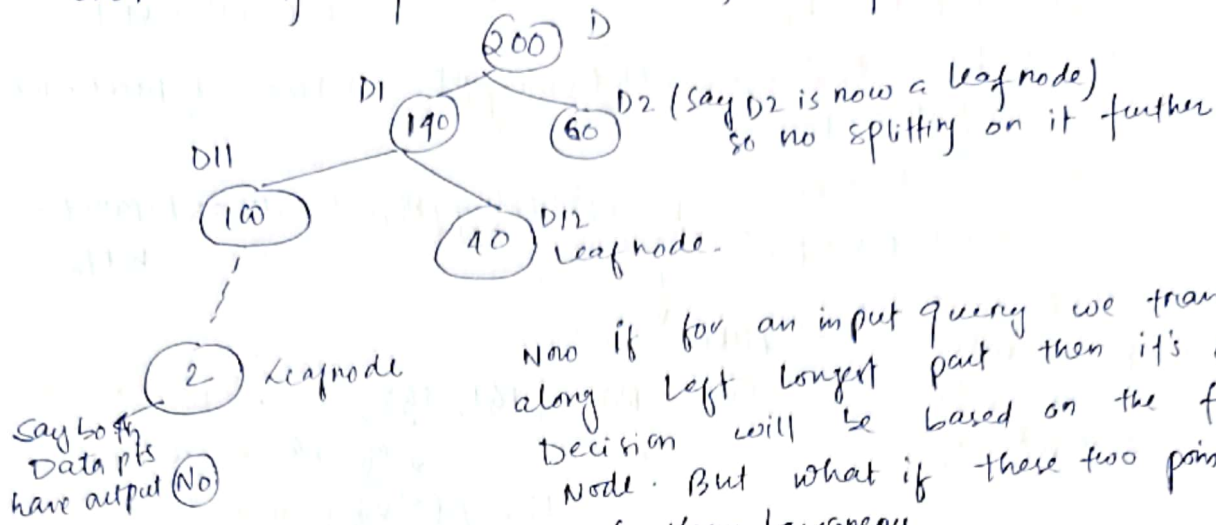
HYPERPARAMETERS (Overfitting & under fitting)

Lo ~~DT~~ has a tendency to overfit (training data pe acche results deta hai par test data pe nahi).

- Depth of Tree $\rightarrow$ is the main thing jiske wajah Δ over fitting ya under fitting hota hai.

Overfitting → performs well on Training data
performs bad on Test data

Let's say we 200 rows in a dataset & on the basis of splitting criteria of a particular column, we split it



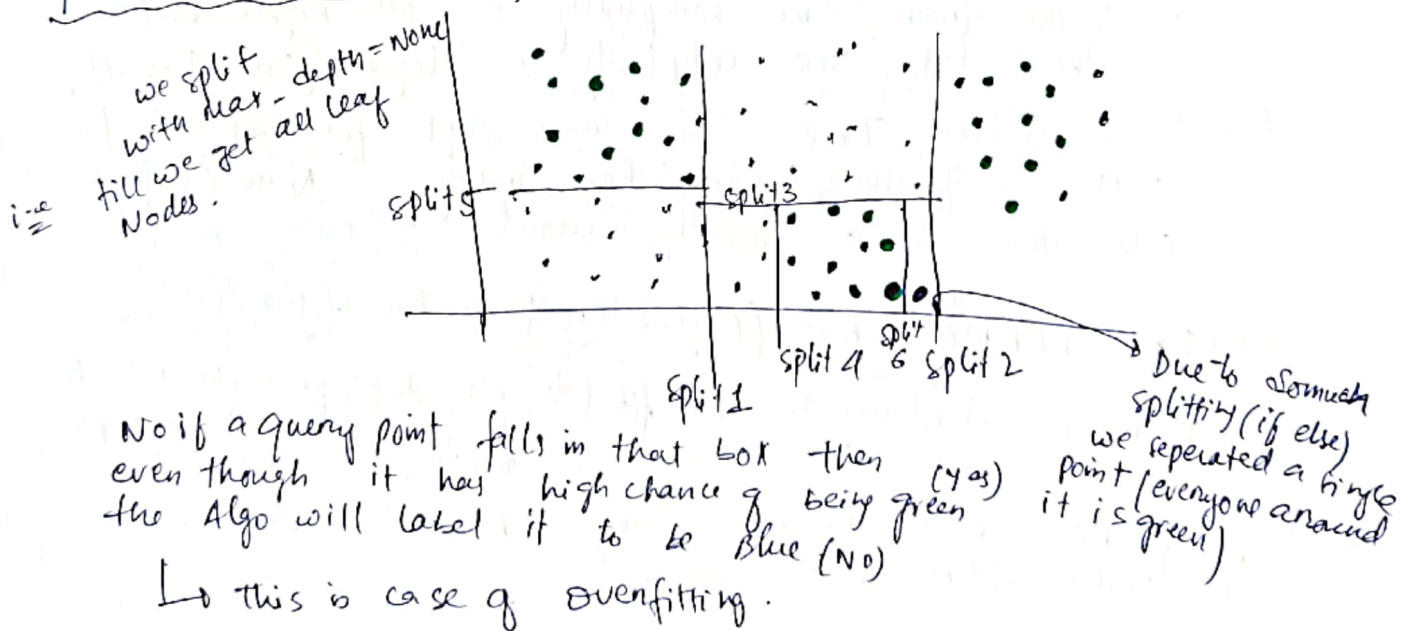
Hence: all those ~~if~~ else to reach outliers → then the system / Algo will fail.

→ 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 Overfitting.

So: if the parameter `max_depth = None` (by default) then the Decision Tree classifier runs till it gets all the Leaf nodes & in such case there is a huge chance of overfitting.

Geometric Intuition of Overfitting:-



→ This is case of overfitting.

underfitting :-

Now max-depth = 1 (only 1 splitting)



Now we have a new query input & it goes into (140) wala then since it has more yes than NO the output of the query will be predicted Yes (even though it may have output any No.) So we don't want both underfitting (low value of max-depth) & overfitting (high or None value for max-depth)

Hyperparameter

for more hyperparameter Tuning:- see again the video Decision Trees - Hyperparameters from 12:00 minutes.

imp hyperparameters are:- Criterion (Gini, entropy)

splitter
by default = None
Max-depth (✓)
min samples split
Min samples Leaf
Max Features
Max leaf nodes.
Min impurity Decrease

see the documentation of decision Tree Classifier & learn about these more.

test (default) random

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 hai for training data pe utna accha fit nahi kar pata algorithm.

min samples split = 10 (say)

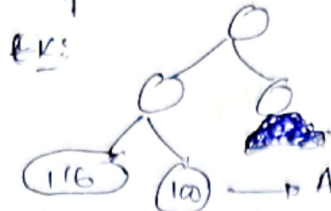
Lo 91 means suppose we are splitting Dataset into D1 & D2. D1 will be further splitted iff min 10 rows are available in D1 & similarly on D2 & if there is further splitting then aage ke child Datasets ko splitting shi tab hi hoga if they have atleast 10 (rows/dataset) hai unme otherwise us child dataset ko splitting ruk jaiga.

Lo This is also other way of trimming our Decision

Tree.
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 min sample leaf 119 nahi hua toh usko split karne hi nahi doge.

Say min sample leaf = 100



Ex: Algo will stop here $\because$ agar iske age split kiya then the leaf nodes will have 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.

- Max features :- we can choose ki hum Decision Tree ko kithe col dena chahte hai to work at every node. Say max features = 3 $\rightarrow$ then it will split based on the 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.

$\rightarrow$ 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
 $\rightarrow$ It means ki agar split karne pe based on a col agar min α impurity ghatega toh hi split hoga warna nahi hoga.

Higher value $\rightarrow$ underfitting

Lower value $\rightarrow$ overfitting.

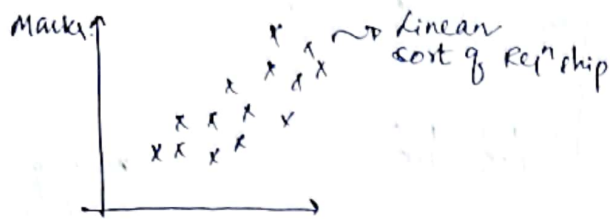
Decision Tree is imp because it is used in

- + Random forest
- + Bagging
- + Gradient Boosting

} also we need to know clearly ki decision Tree ke konse hyperparameters ko kaise tune karne to get desired Result.

Regression Trees (Decision Trees on Regression problem)

Where is it applicable?



hrs of study (in entire semester)
Here we can directly apply Linear Regression

Target col is Numerical

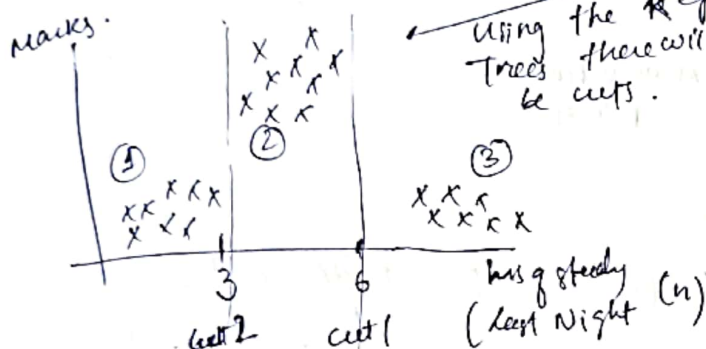
Optimal padhai jyada marks

marks

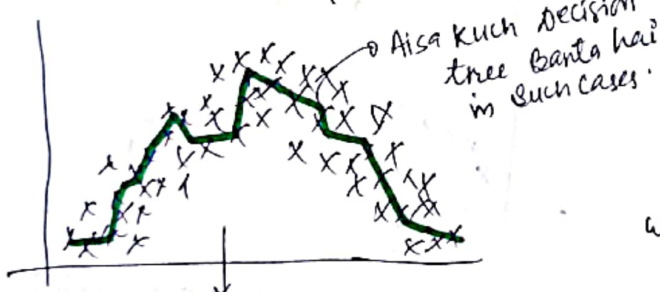


hrs of study (night before exam)

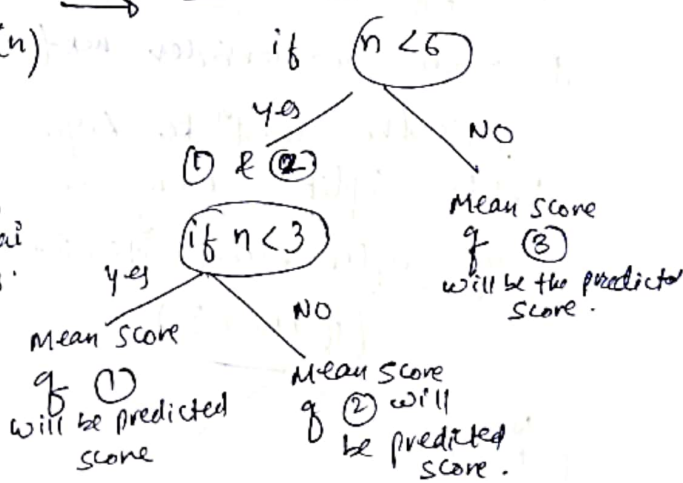
Here Linear Regression will not help. Hence we use Regression Trees when there is a non linear pattern in the data.



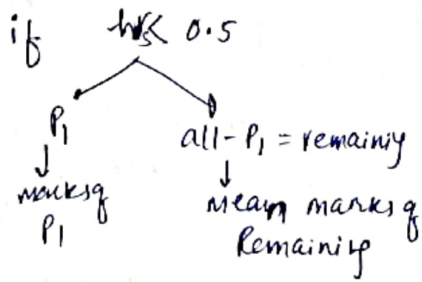
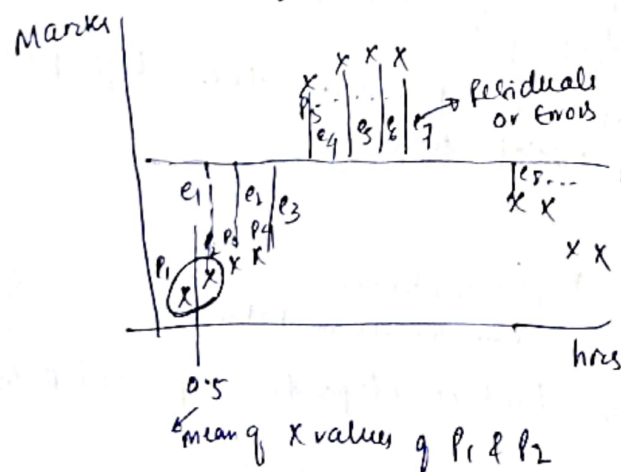
but what if data Aisa hota



code will be:



Then splitting criteria Visible nahi hai → Regression Tree handles this.
So how Regression Tree will work (step 1:- select (at two))



Calculate: $SSE = e_1^2 + \dots + e_n^2$
Sum of squared errors

Now select another two

P_2 & $P_3 \rightarrow$ mean of their

X-values = 1.5 (say)

& proceed similarly.

Here:

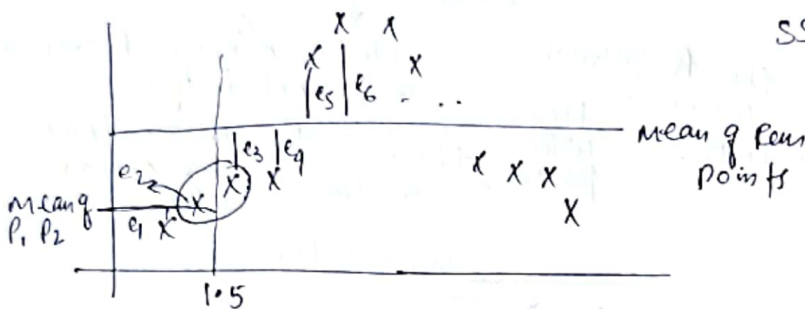
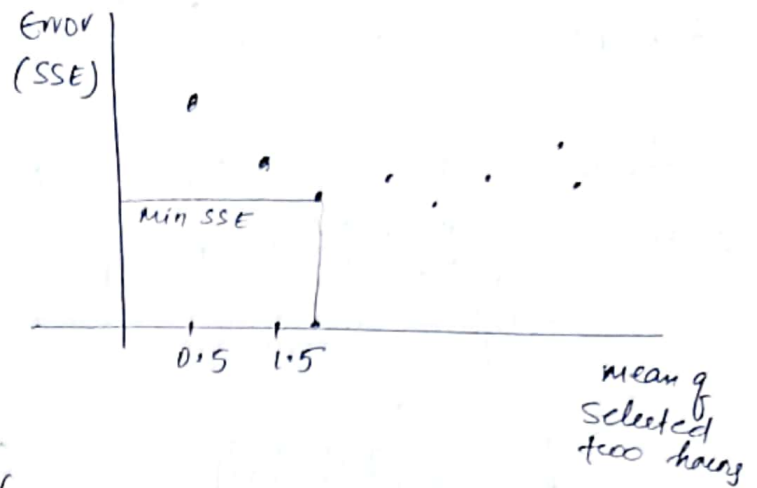
hrs < 1.5

P_1, P_2

mean marks
of P_1, P_2

all - $P_1, P_2 = \text{rem}$

mean marks of
Remaining.



$$SSE_2 = e_1^2 + e_2^2 + e_3^2 + \dots + e_n^2$$

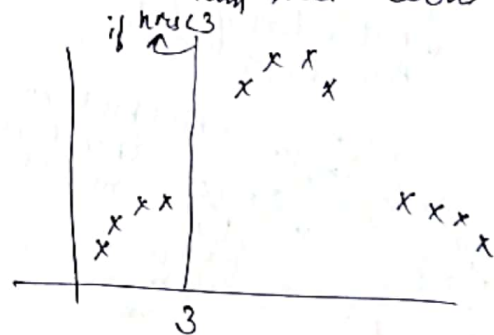
& so on we consider next two points & proceed.

$\therefore$ jis bhi condⁿ ke Liye SSE minimum hai wahi hamara
Pehla split banayega.

so say after first iteration

if hrs < 3

Now in both
parts repeat the
process



Note: if we keep on continuing then we will split each point.
(overfitting) $\rightarrow$ we only need to split in clusters.

for this if a child has $\leq \text{threshold}$ # 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

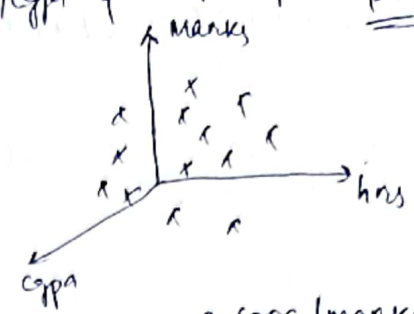
split continue.

using the above method.

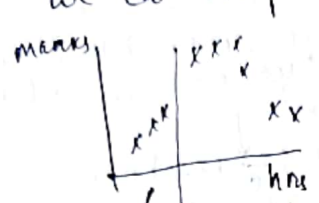
generally the threshold is 20-25 but it depends upon the data

* what if we have input cols hours/cgpa & output col marks?

we use plane to cut/split & in further higher dimension we use hyperplanes.

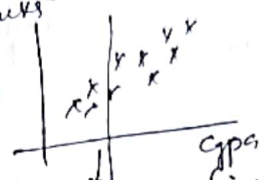


we do independent analysis:- hours/marks & cgpa/marks.



find the first splitting point using above method

for the corresp. we have min sum of squared Error = SSE_{hrs}



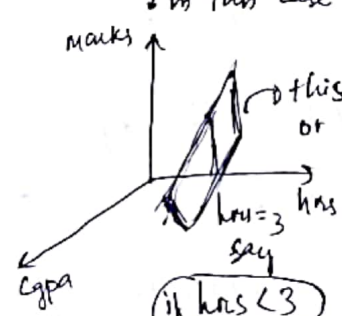
find the first splitting point using the above method.

for the corresp. we have min sum of squared Error = SSE_{cgpa} .

Now there are two cases:-

Case 1: $SSE_{hrs} > SSE_{cgpa}$
in this case

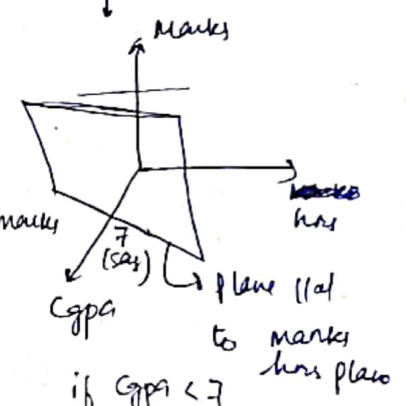
Case 2: $SSE_{hrs} < SSE_{cgpa}$



if $hrs < 3$

if here we have more than threshold data then we split or stop

same here



if $cgpa < 7$

same here

same condⁿ 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

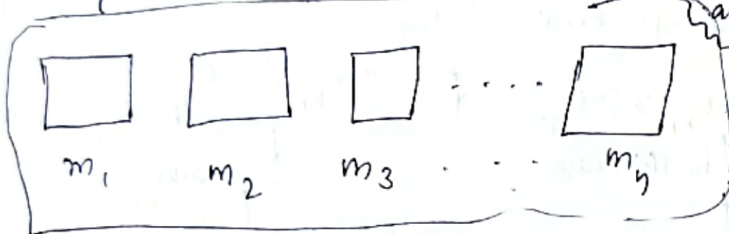
Wisdom of the Crowd:- → in KBC → audience poll, almost all times the audience knew the answer.

Core idea of Ensemble Learning:-

^{stage}
prediction in Ensemble Learning

say we have an Ensemble Learning Algorithm (Collection of smaller ML Algos / base models)

→ Ensemble Learning Algo



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 & $\therefore$ they were trained differently & hence variety is created.

$m_2 \rightarrow$ Linear Reg

$m_3 \rightarrow$ Linear Reg

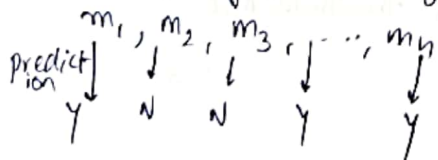
$\therefore$ & so on

or we can use combination of above two.

Now suppose we have a new query input:

we have CGPA & IQ & have to predict placed or not. (classification problem)

this query point is given to every model is



→ Ab Democracy wala logic lagega (majority count) & wohi actual prediction

→ Product review & humme reliability milta hai ki humme product kharidne chahiye ya nahi in online Marketing. Bahot logo 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 → it's 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.

↓
Jesi cheez pe based hai

Ensemble Learning

↓
ie Crowd ko single & jyada pata hota hai

↓
Crowd should be mixed (not all of same type like software Engineer)

Ex: Soft: (multi class classification problem)

page 85

for a regression input query: CGPA 10 & to predict (placement in lacs)

page 83

the input is again given to all the models:-

$m_1, m_2, m_3, \dots, m_n$
 $\downarrow \quad \downarrow \quad \downarrow \quad \dots \quad \downarrow$
 8.1 3.2 4.1 $\dots$ 7.5

Actual prediction will be the Mean of this outputs.

Types of Ensemble Learning

4 types that are used most are

Voting Ensemble

in this the base models (m_i 's) are different Algos

Ex: $m_1 = \text{SUM}$
 $m_2 = \text{Logistic Reg}$
 $m_3 = \text{Decision Tree}$

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 har Algo (model ko) Benabar value diya jaa rha hai

Say each of them will have 500 datapoints then by Bootstrapping we randomly chose 500 datapoints from Data & make it D_1 , again out of 500 random Datapoints $\rightarrow D_2$ & 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 model 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 SUM.

Now in data we have 100 rows $\downarrow$ 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

$m_1 = m_2 = m_3 = \text{Linear Reg.}$

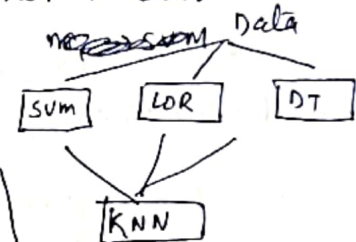
Here: the data is given to m_1 & m_1 keeps track of kaha kaha use galatiya ki &

yahi data with the galatiya m_2 ko Explain karke par kar dete hai & yeh m_2 galatiya par wahi result dene lagta hai par iski khudki galatiya bhi hai & wo m_3 ko batayata jata hai

This is Boosting $\rightarrow$ hence finally the model makes lesser mistake.

Stacking

is similar to voting Ensemble but yaha har model ko kuch weights assign honge by a different ML Algo & hence har model ka value in prediction will not be same



SUM, LOG (log-log) & DT will be trained on two Data & KNN will be trained on

SUM	LOG	DT	Actual/Final Prediction
1	0	1	1
0	1	0	0
-	-	-	-
-	-	-	-

& this KNN algo will give weights to the above models.

isse ye pata chalta hai ki konga model jyada accha hai in prediction.

Ensemble Learning in classification problem pick ^{prediction} data from multiple models & create 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.

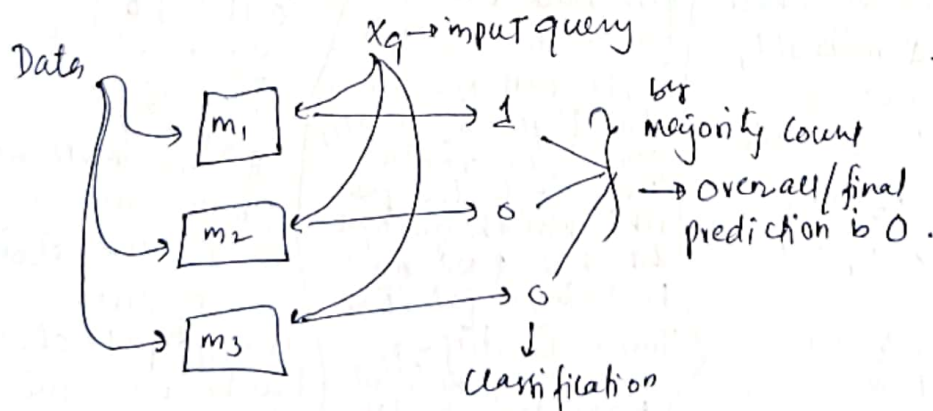
Sly- if model has:- High bias + Low variance

we'll get Low bias + Low variance ^{use Ensemble Learning}

- Robustness — Data mei bohut jyada change shi kardo toh is performance jyada change/fluctuate nahi hota.

VOTING ENSEMBLE

- + Core idea / Intuition
- + Voting classifier (for classification problems)
- + Voting Regressor (for Regression problems)



Agar Regression problem hota toh hamko ka Mean return karta as final prediction.

Disadvantage is:-

we are training multiple models & hence the computational complexity increases.

* When to use Ensemble Learning?

Ans:- Always.

Why Voting Works?

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

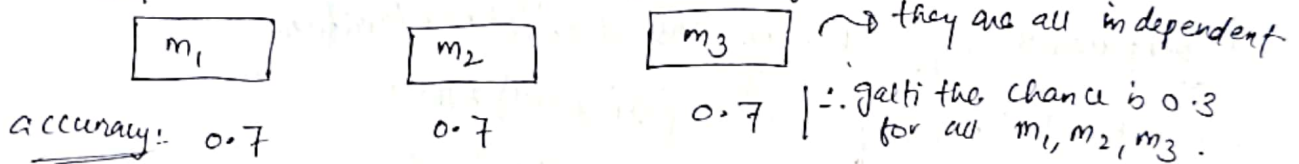
Before that let's discuss the Assumptions behind Voting Ensemble:

- All the models are independent (kind of different)
- Each ^{individual} model ka minimum accuracy should be $> 51\%$ nahi toh overall accuracy $\cdot \min\{m_1, m_2, m_3\}$ hi kharab hoga.

Why voting works :- When we have a crowd then all the individuals are from different Backgrounds & have different knowledge & hence ek ki kami ko dusra chupa leta hai & overall performance is better.

These are called base model.

↓ Mathematically proving this (probability)

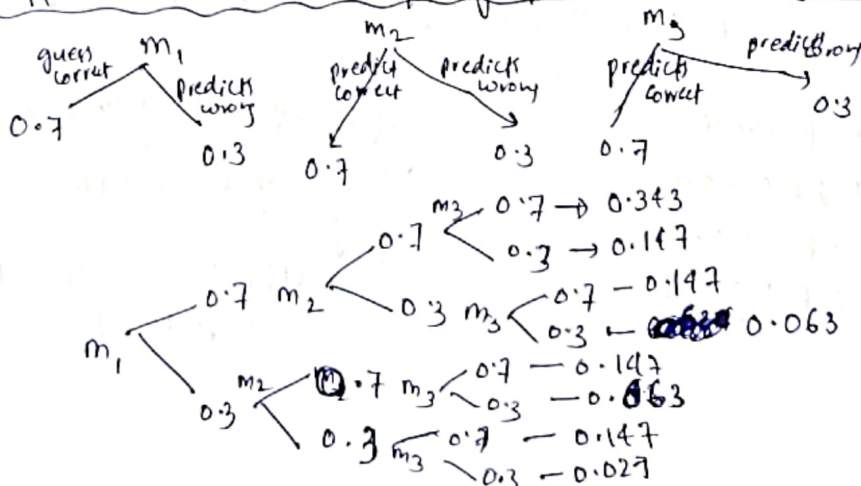


Now ab agar inke voting classifier banaye then majority jo belongs uski final prediction hoga.

To show accuracy of voting classifier > 0.7 .

Suppose we have a new query point:-

→ since they are indep.



Now:- Voting classifier will predict yes if there are at least two yes (majority count). $\rightarrow$ Hence the prob of that is:-

$$YYY + YYN + YNY + NYY$$

$$= 0.343 + 0.147 + 0.147 + 0.147$$

$$= 0.784$$

$\rightarrow$ Hence the prob to predict yes by the voting classifier is 0.784

$\downarrow$ 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 using

Voting classifier

we have discussed the core idea of majority count before.
 $\rightarrow$ See this video again from 3 minutes to understand & visualise again.

hyperparameters: sklearn.ensemble.VotingClassifier

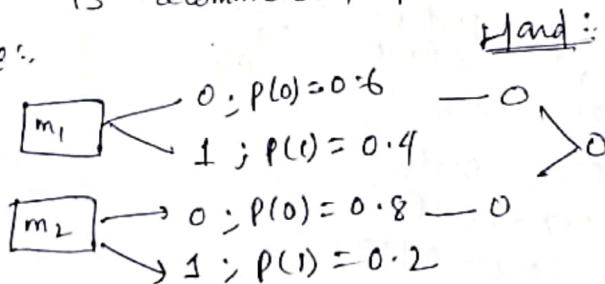
$\rightarrow$ estimators: list of ^{models} models ('str', model) type.

$\rightarrow$ voting: {'hard', 'soft'} ; default: 'hard'

if hard $\rightarrow$ uses predicted class labels for majority rule

if soft $\rightarrow$ predicts the class label based on the average of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

Example:-



Soft

$$\text{avg prob of } 0 = \frac{0.6 + 0.8}{2} = 0.7$$

$$\text{avg prob of } 1 = \frac{0.4 + 0.2}{2} = 0.3$$

$\downarrow$

$\therefore$ output = 0 $\because$ av prob of 0 > av prob of 1.

classifier 1	classifier 2	classifier 3	classifier 4	classifier 5
class 1 : 90%	10%	1%	25%	5%
class 2 : 8%	80%	85%	65%	90%
class 3 : 2%	10%	14%	10%	5%

Average :

class 1 : 26.2%
 class 2 : 65.6%
 class 3 : 8.2%

→ final prediction: class 2

Generally speaking soft voting ka result better aata hai hard voting se. But run both & see kisme better aarha hai.

* for concentric Dataset :-



→ using Log. Reg, Bayes, Gaussian Naive, & 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 are

Jupyter Notebook

estimators list & weights

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 Algorithm.

Bagging Ensemble

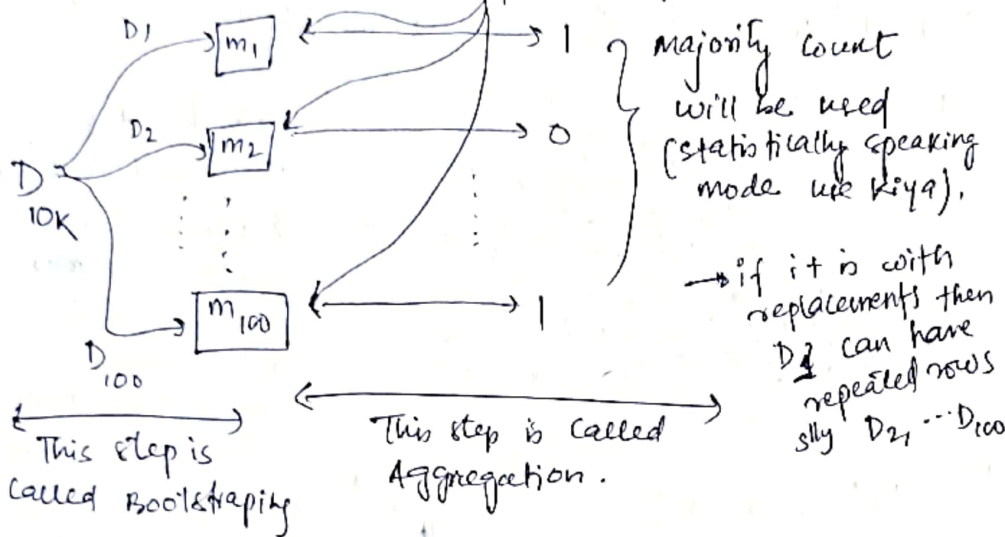
- Core idea (intuition, why use Bagging, when to use, Types of Bagging).
- Bagging classifier
- Bagging Regressor

Bagging $\rightarrow$ Bootstrapping + Aggregation.

Given population m_i
 $\downarrow$
 Samples draw k times 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 p Train p (say 1000 rows p)



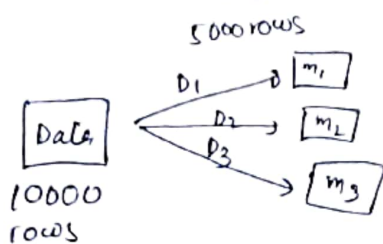
This is done by Bootstrapping :- randomly selecting 1000 rows from D_{10K} (sometimes it can be with replacement & sometime without replacement).

* why use Bagging? why it works?

- $\rightarrow$ It is based on Bias variance tradeoff (we need Low Bias & Low variance) But they are inversely prop.
- $\rightarrow$ Bagging can provide Low Bias + Low Variance.

In Bagging we take Algos (Base models) that are Low Bias + High variance (overfitting) (ex. Decision Tree with max-depth = none), SVM, KNN

Now Bagging will reduce variance &



Now if we say change 500 rows & again train then these 500 rows in the original data will be distributed Randomly. Again lab ex ki ko mil jate then that model would have changed drastically but due to random distribution among all three models the change in model is considerable & on these new set of data also the model performs similarly.

Hence, jo resultant model hai uska variance is

Low (Low variance matlab :- Agar data mai changes kardo toh bhi ~~model~~ model ke output/performance mai jyada changes nahi aate i.e. model consistent Result deta hai)

2. Low bias (accurate) jab wo the 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.
 using Decision Tree as models $\rightarrow$

→ implementations of Bagging is using Decision Trees as base models → Random Forest

for Classification problem.

Classification
an.

Jupyter Notebook

Bagging classifier (General Base models are:-
Decision Trees, KNN & sometimes SVM)

• Advantages: Decision Trees

Base estimator/model (default is Decision Trees)

- # of estimators/models $\rightarrow$ sampling
- max-samples $\rightarrow$ Har model ko Kithe row dena chahate hai.
- Bootstrap samples $\rightarrow$ means sampling with replacement
ie D_i can have repeated rows.
- max-features $\rightarrow$ Kithe features/ko use karna chahate hai.
- Bootstrap features $\rightarrow$ Kithe features use karna hai

These are hyperparameters

RANDOM SUBSPACES

$\rightarrow$ max_samples = all rows
 Bootstrap_sample = false
 max_feature = not all cols (say 2 if there are 5 cols)
 Bootstrap_Features = True.

↓ sampling on colcumn
This is Done when we have
Large no. of rows like on
MNIST Data set.

RANDOM PATCHES (Both Row & Col sampling) ^{MNIST dataset}

Say

- max-samples = 50 (50 rows for each model)
- Bootstrap Samples = True (with replacement)
- max-features = 5 (say there were 3 cols)
- Bootstrap-features = True.

Typyfer Notebook.

Bagging Regressor (for a Regression problem)

↳ In the Aggregation step $\rightarrow$ the final output is the mean of all the individual outputs & the Bootstrap step is the same as Bagging classifier.

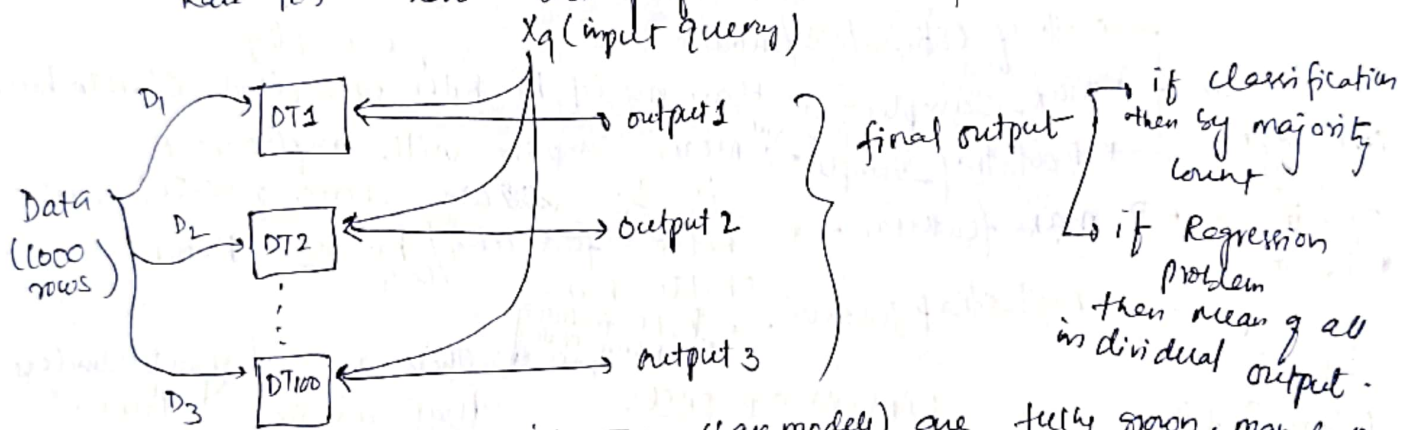
as it is a bagging technique (bootstrap aggregation) $\rightarrow$ group of Decision Trees

RANDOM FOREST (VIMP)

Bagging with base model Decision Tree. (Hence the name Random Forest)

↳ 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 Box - Bohot jyada tune karne ki jarurat nahi hai. i.e. hyperparameters Bohot jyada change nahi hti kare toh bhi iske performance kafi acche rehte hai.



↳ All the Decision Trees (base models) are fully grown; max-depth i.e. low bias + high variance. = None

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_1 can have repeated Rows due to replacement
Similarly other D_i 's can have repeated rows.

Since har decision tree (base models) ko alag-alag data diya ja raha hai toh ho sakta hai ki har decision tree alag alag dikhe.



Jupyter Notebook.

Bias-Variance & 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 by some noisy data, due to bootstrapping the new data points get's distributed into 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.

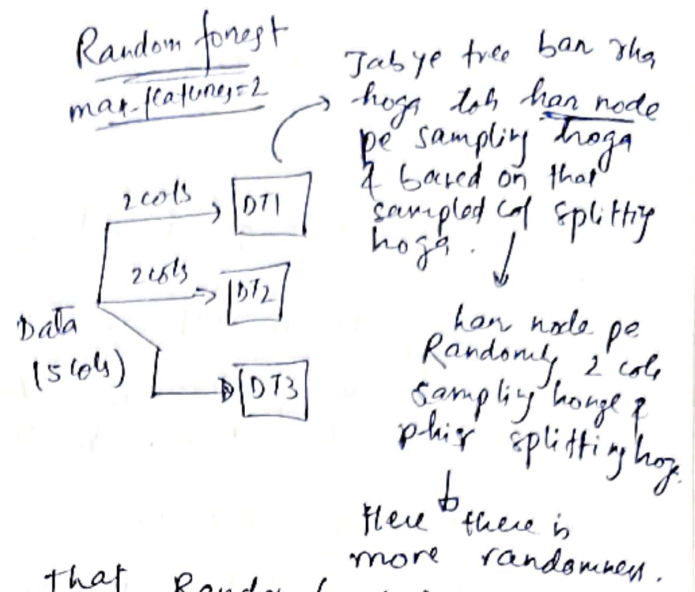
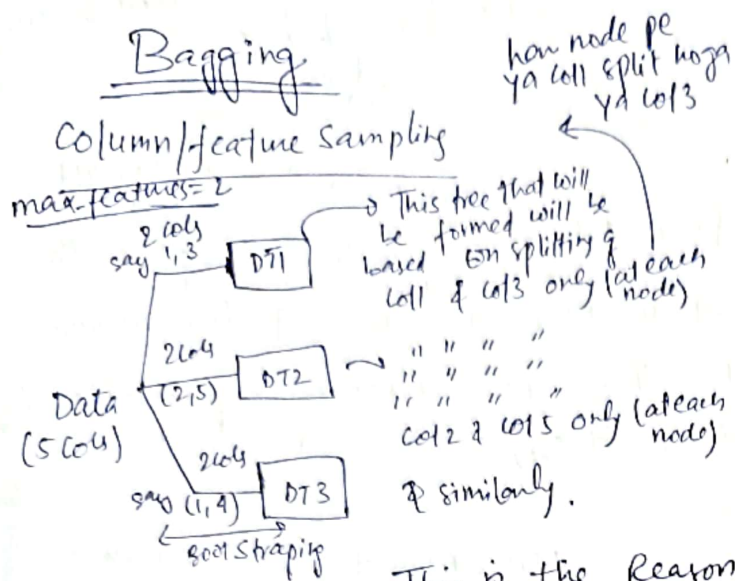
Bagging V/s Random Forest:-

The Differences:-

- In Bagging the Base models/estimators can be Decision Trees, KNN, SVM (& some other algo). But in Random forest the Base models is Decision Tree. There is a separate class for Random forest in sklearn & different class for Bagging.

Question is: Bagging with Base models — Decision Tree → Is it a Random forest → Ans: No.

The difference is in feature sampling (max-features parameter)



This is the Reason that Random forest is better than Bagging due to the extra randomness.

Jupyter Notebook.

Hyperparameters in Randomforest Classifier :-

↳ & Similar hyperparameters in Randomforest Regressor.

Hyperparameters to tune Random Forest :-

— number of estimators :- decides the No. of Decision Trees to be used to make the random forest.

— 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.

— Boot strap :- if we want to select the rows/cols randomly or not (default: True); if false the whole dataset is used for each Decision Tree.

— max sample :- # of rows we want to give to each Decision Tree (if boot strap is True)

Generally 50% to 75%.

- none (default) then use X.shape[0] samples
- if int then jo diya hai usko use karo
- if float (0.1) then 0.7 means 70% of rows use karo.

- criterion : gini (default) or entropy
- max_depth : int, None (default)
 ↳ fully grown
- min_samples_split → ithe samples agar hai node mei tohi further split hoga.
- min_samples_leaf → split karne ke baad agar leaf mei atleast ithe samples hokaye toh hi split hoga.
- max_leaf_nodes → kithe leaf nodes we want to keep in each Decision Tree.
- min_impurity_decrease : Agar node ko split karne pe atleast itna int, none (default) impurity ke hoga toh hi split hoga.
- n_jobs :- (multi processing), The # of jobs to run in parallel. fit, predict, decision-path & apply are all parallelized over the trees.
 None means : 1
 -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 < η -features)
 ↳ total no. of cols
- warm_start (default = False)
 used when stages mei model ko Train karke hai (when set to True, reuse the sor 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 :- ↳ out of Bag

in Random Forest :- while bootstrapping (in sampling with replacement (so repeated rows are possible)) 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 ^{the performance} model once it is trained. 1
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/col jyada important hai output predict karne ke liye. in a particular Algorithm. & then we keep the most important feature & Remove the Rest.

Jupyter Notebook.

How is feature importance Calculated.

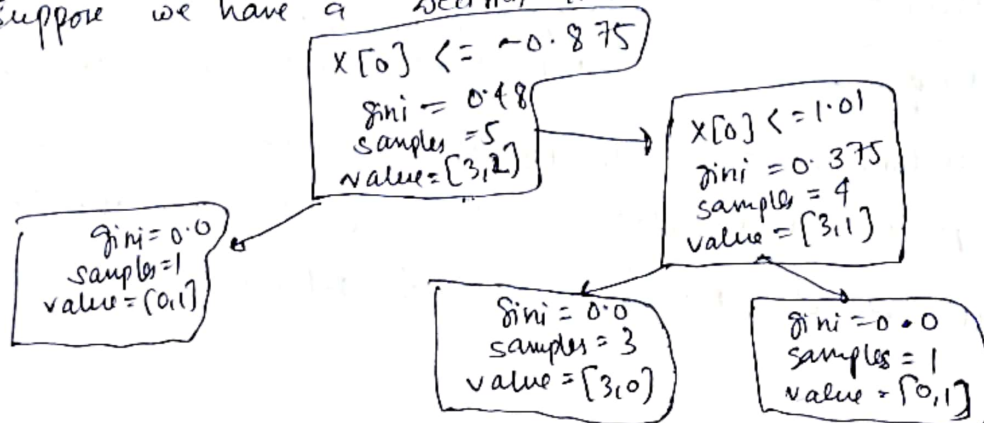
+ to understand how it is calculated in Random Forest we need to know how it is calculated in Decision Tree.

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 Notebook.

Suppose we have a Decision Tree:-



Now:- kisi bhi col/feature ka importance

Page 91.

Calculate karna hai using Decision Trees:-

of rows in node n_i

of that node (Gini value)

of rows in right node n_r

$$n_i = \frac{N_{\text{left}}}{N} \left[\text{Impurity} - \left(\frac{N_{\text{left}}}{N} \times \text{right-impurity} \right) - \left(\frac{N_{\text{right}}}{N} \times \text{left-impurity} \right) \right]$$

Le node ka importance $\rightarrow$ Total # of rows present in data

of rows in node n_i

Gini value of right node of n_i

Gini value of left node of n_i

slightly these terms have similar meanings

$$\sum n_i$$

if node split on feature k

$$\sum n_i$$

if all nodes on which splitting occurred.

feature imp of k^{th} col

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 diagram/tree where splitting $x[0]$ ko wajah $\rightarrow$ $x[0] \leq 1.01 \rightarrow$ only 1 node.

if iss node the importance find karenge pehle

$x[0] \leq 1.01$
Gini = 0.375
Samples = 4
value = [3, 1]

Hence, then feature importance of $x[0] = \frac{\text{sum of all node importance of those nodes where splitting } x[0] \text{ ke wajah } \rightarrow \text{ need hai}}{x+y}$

sum of all the node importance where splitting irrespective of kis col ke wajah $\rightarrow$ $\rightarrow$

* 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 $x[0] = \frac{0.30}{0.30 + 0.18} = 0.625$

& " " " $x[1] = \frac{0.18}{0.3 + 0.18} = 0.375$

Now Calculating feature importance in Random Forest is not that difficult:-

In Random forest :- har decision Tree (there are multiple), har col/feature ka feature importance calculate karta hai

sup
ish
ki
ab

Suppose there are 10 Decision Trees in a Random Forest.
 then each decision tree will have its own feature importance
 calculation. Then Actual/final feature importance will be
 = 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.

Ada Boost:-

opposite:- strong learners:- jiska accuracy bahot
 achha hota hai 95%, 99% etc.

- Weak learners:- Aise ML Algo jiska accuracy 50% just
 thora sa jyada hota hai.

in Adaboost:- we combine/adding multiple weak learners to
 make a bigger & better models.

- 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 -ve points).

↳ pts in +ve
 region.

↳ pts in -ve region

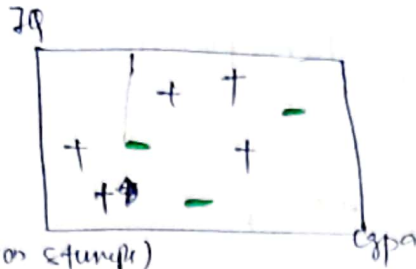
The Big Idea of AdaBoost: It is a stagewise additive method.

page 92

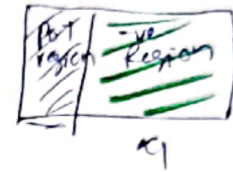
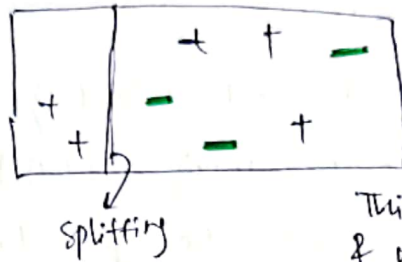
- Not placed

+ placed

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 waise jyada hoo



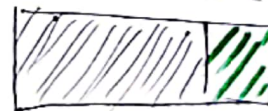
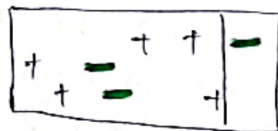
for representing the model
 $h_1(x)$
Output
(+1 or -1)

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 ki is Decision stump ka contribution weight kitna rahega in final output.

Step 2 - 2nd Decision stump:- focusing more on first decision stump mistakes makes a split such that the entropy de

is highest possible
iske perfer. ka karte
iska weightage α_2
calculate hoga.

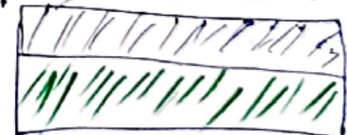
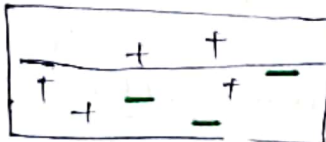


$h_2(x)$

thik kar diye but iske ab khudke mistakes hai $\rightarrow$ (-ve wale in the side)

Step 3: 3rd decision stump (the -ve in the regions) are upsampled & are focused on more here & again split is performed to de the highest entropy possible

iska weightage α_3 hai
suppose α_3 hai
iske perfer. ka karte
iska weightage α_3
calculate hoga.



$h_3(x)$

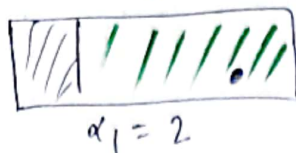
Now we have three Base models (on the right, diagram) & their weights (α_i). Now we add the three models so AdaBoost model

$$A = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

if pos A = +1
if -ve A = -1

Now let's do a prediction :- $C_{ypa} = 8.5$, $IQ = 81$

1st model.

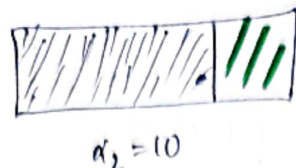


-1 (not placed)

- Not placed

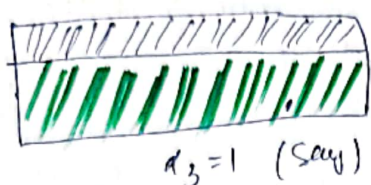
- placed

2nd model.



+1 (placed)

3rd model.



(-1) (not placed)

∴ final prediction :-

$$A = \text{sign} (\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3)$$

$$= \text{sign} [2(-1) + 10(1) + 1(-1)] = \text{sign}(7) = +1$$

∴ the student will be placed.

final Decision Boundary ke liye teeno model ke Decision Boundary ko ek ke upar ek rakho. we set.



$n = 5 = \# \text{ rows}$

Adaboost step by step

→ initially hum har apne row ko ek weight assign karte hai (1 equal value assign karte hai)
→ generally the weight assigned is $\frac{1}{n} = \frac{1}{5} = 0.2$
So there is now a weight col:-

x_1	x_2	y
3	7	1
2	9	0
1	4	1
9	8	0
3	7	0

weight
0.2
0.2
0.2
0.2
0.2

or max gini impurity use.

Stage 1 :-

1) we train a Decision stump (m_1) (split with max entropy use or max information gain).

Now iss m_1 ka performance dekhenge on the data set & we find y -pred.

x_1	x_2	y	y -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

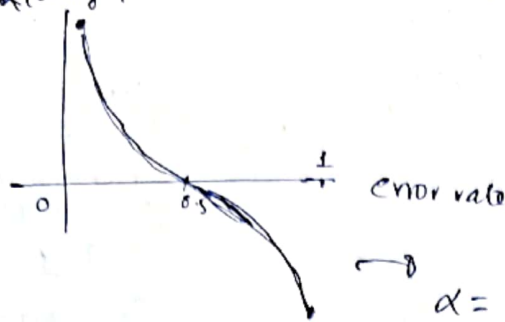
→ wrong prediction.

Now we find $\alpha_1 =$ weight of m_1 on final prediction.

Page 93.

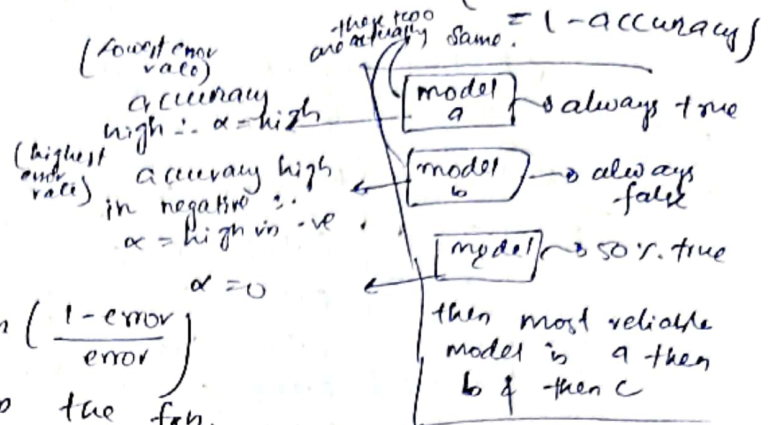
↳ it depends upon error rate (m_1 ne data pe kitni error rate jitna jyada, model utna galti ki) utna hi kam. $\therefore \alpha_1 \propto \frac{1}{\text{error rate}}$ (generally error rate)

$\therefore$ we want a fn with graph



$$\alpha = \frac{1}{2} \ln \left(\frac{1 - \text{error}}{\text{error}} \right)$$

↳ this is the fn.



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 in rows ka initial weight ka sum:- So for m_1 , error = $0.2 + 0.2$

$$\therefore \alpha_1 = \frac{1}{2} \ln \left(\frac{1 - 0.4}{0.4} \right) = \frac{1}{2} \ln \left(\frac{0.6}{0.4} \right) = 0.20$$

$$\alpha_1 = 0.20$$

Ab m_1 ke galtiyan ko upsampling karke (importance badha kar) Stage 2 (m_2) ko bhejne hai:-

(misclassified) To increase the importance of mistakes of m_1 :- Galti wali rows ka weights ko increase (Boost) kar denge & classified wale points ka weight ise kar 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 class. 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
3	7	1	1	0.2	0.16	0.166	Sum=0.96
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	

Sum=1 Sum=0.96

making compact to send to next stage

like new

we need to make sum=1

upsampling

5 random numbers generate b/w 0 & 1.

Say 0.13 → Row 1
0.43 → Row 3
0.62 → Row 3
0.50 → Row 3
0.8 → Row 4

x_1	x_2	y	new-wt	range
3	7	1	0.166	0 - 0.166
2	9	0	0.25	0.166 - 0.416
1	4	1	0.25	0.416 - 0.666
9	8	0	0.166	0.666 - 0.832
3	9	0	0.166	0.832 - 1.0

cumulative sum upper value - lower value

0.166 + 0.25
0.416 + 0.25
0.666 + 0.166
0.832 + 0.166

now we make a dataset with only these rows.

∴ Dataset for stage 2.

Row 1
Row 3
Row 3
Row 3
Row 4

this will go to the second Decision stump & Everything will be repeated.

Jupyter Notebook

Adaboost Hyperparameters:

This gives the best result.

- base-estimator (default:- Decision Tree with max-depth=1)
↳ otherwise we can use Logistic Regression, SVM, etc.
- n-estimators = # of base models 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 gives the contribution of each classifier. There is a tradeoff b/w learning rate & n-estimators parameters.
- algorithm (Samme, Samme.R) (default)

Jupyter Notebook

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

in Boosting: the base models are

High Bias + Low variance (like: shallow Tree whose depth is small) - ie. Decision Stump (or KNN with low neighbourhoods)

Sequential v/s parallel Learning

in Bagging the Dataset is split randomly into subdataset & given to different base models.

in Boosting: Data $\rightarrow$ model then to model 2 ... and so on. Sequential Learning (model ek ke baad ek train ho rhe hai).

Parallel Learning (same models sath mei train hote hai)



Weightage of base learners

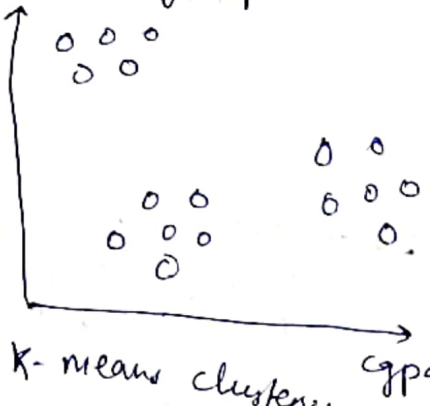
in Bagging the base models/learners have weightage = 1 (by default)

in Boosting: han model ka apna weightage (α_i) hota hai. (sath hi vote same hai) jiska weight jyada uska contribution jyada in final output.

K-MEANS CLUSTERING

Clustering is an unsupervised ML technique where we put similar things in a group.

Internal working

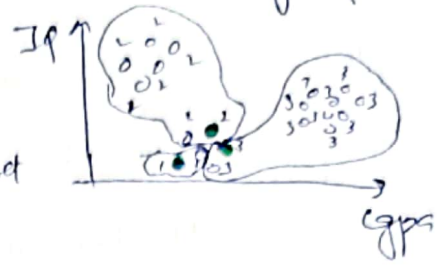
Data $\rightarrow$ IR 
to 2D mei dikh rahi hai toh hum dekh kar hi cluster kar denge & bol payenge ki konga point kis cluster mei jaige. hata then hum
But ye agar higher dim clusters dekh nahi payenge
Steps: 1. Decide n clusters & then we use K-means clustering.
2. Initialize centroids
3. Assign cluster
4. Move centroids
5. Finish.

1. Decide n clusters :- we'll have to tell the algo how many clusters we want (wired) $\rightarrow$ 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 clusters :- Ab humne har point ko Ek cluster assign karna hai on the basis of the 3 centroids chosen.

Ek ~~point~~ point ka ~~distance~~ cluster centroid $\&$ euclidean distance nikalo $\&$ jis centroid $\&$ distance sabse kam hai use us centroid ke cluster mei rakho.

Aisa hum har point in the Dataset ke liye karenge. We can see the clusters above.

4. Move Centroid.

Ab in teeno clusters ka centroid calculate karenge. Say cluster 3 ka centroid :- (mean of $Cgps$ of points in cluster 3, mean of J of point in cluster 3)

Slly. we will find centroids of cluster 1 $\&$ cluster 3.

This need not be a point on the Dataset $\rightarrow$ 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 ¹ same then no improvement in centroids $\rightarrow$ finish the algo $\&$ if they are different then algo is improving by moving the centroid $\rightarrow$ continue.

Assign clusters again with these new centroids.

Do this till finish condⁿ is met $\rightarrow$ move centroids again using the above formula

$\rightarrow$ 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.

$\rightarrow$ 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-mean clustering Alg? (Elbow method)

Page 95.
within clusters sum of squared distance
WCSS (inertia)

In this we plot a graph called elbow graph/curve
To find WCSS for a cluster to aur uska centroid ka Euclidean distance us cluster ke har points se 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 maano ki pura dataset ek hi cluster hai:-
uska WCSS nikalo & plot kardo (1, WCSS₁)

Now consider two clusters in the Dataset,
Total WCSS = sum of individual WCSS
plot (2, Total WCSS₂)

Next: Consider 3 clusters in the Dataset,
Total WCSS = sum of individual WCSS
plot (3, Total WCSS₃)

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 ~~more~~ clusters are increased & hence WCSS decreases)

$$WCSS_1 > WCSS_2 > WCSS_3 > \dots > WCSS_n$$

if all the points are centroids i.e.
of clusters = # of points in Dataset
then individual WCSS = 0 $\therefore$ 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 # of clusters. (steepness/slope decreases at that point).

Jupyter Notebook.

on salary then m_1 .

GRADIENT BOOSTING

For Regression

Consider the toy dataset:

IQ	cgpa	salary
90	8	3
100	7	4
110	6	8
120	9	6
80	5	3

Mean
4.8

Step 1.

IQ | cgpa | 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

res1
-1.8
-0.8
3.2
1.2
-1.8

Here Loss fn = pseudo residual = actual - predicted
Now is galti ko humne somehow model 2 tak bhejna hai

Our model 2 generally is Decision Tree (Gives best Result).

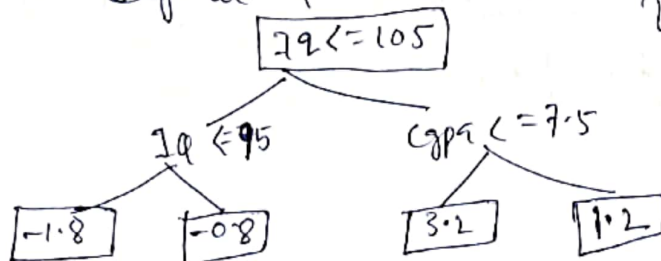
Note: To model 2 we give:

IQ	cgpa	res1
:	:	:

 output i.e. by model 2 & jitha accurately we want to predict the res1 (mistakes of m_1)

m_2 res1 pred. Karega hum utna accurately hum actual output (salary) predict kar payenge.

Say we trained m_2 (Decision Tree) & we get:-



→ sequential stage wise addition matlab data ek ke baad ek base models pe train hoga.

& hum next base model ko apne pichle wale model ka galtiyo ko pata hoga & wo un galtiyo ko nahi karega.

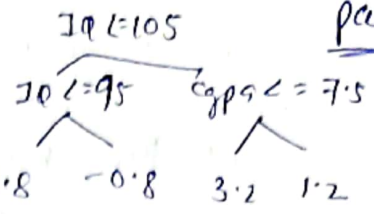
Let's say we want to use three base models:-

m_1 m_2 m_3

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

Step 2. Now we need a Loss fn that tells ki ~~model~~ model ne galti kitni ki & wohi galti hum next model ko bataenge.

Now we predict on the data M_2



IQ	Cgpa	res1	pred2
90	8	-1.8	-1.8
100	7	-0.8	-0.8
110	6	3.2	3.2
120	9	1.2	1.2
80	5	-1.8	-1.8

use this

Overfitting (since pure training data pe exactly same predict kar diya).

Now we have 3 base models but say age 2 hi hote then Actual prediction kaise hote :-

of M_2 on salary $\leftarrow$ Actual pred = M_1 pred + M_2 pred (which is obvious also)
 Ex. for input query: 90/8 $\rightarrow$ Lye pred. kaafi accha hona chahiye

M_1 pred = 4.8 (mean) $\therefore$ Actual pred = $4.8 - 1.8 = 3$
 M_2 pred = -1.8 (mistake of M_1)

$\rightarrow$ Yaha overfitting ho toh sahi hai (since exact prediction ho sha hai)
 So to avoid this we use Learning rate :-
 hence hame data pe acche prediction mahi hoga

of M_2 on salary $\leftarrow$ Actual pred = M_1 + $lr \times M_2$ pred

Now for input:

IQ	Cgpa
90	8

generally $lr = 0.1$ $\rightarrow$ ye leke se hamara step chota hota hai towards the right direction (actual Ans)
 Actual pred = $4.8 + 0.1(-1.8) = 4.7$
 Now it is not accurate enough (Actual Ans = 3)

So now we use more base models to improve it :-

Adding model 3

IQ	Cgpa	Salary	pred1	res1	pred2
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

Now we find residual 2
 $= \text{res2} = \text{salary} - (\text{pred1} + lr \times \text{pred2})$

res2
-1.62
-0.72
2.88
1.08
-1.62

clearly we see that values of res1 & res2

hence the mistakes are being hence M_2 did lesser mistakes on salary than M_1 .

Till here: (Actual pred of M_2 on)

Salary = $\text{pred1} + lr \times \text{pred2}$

Ideally we want the residual = 0 & as all of base models gives the residual ↓.

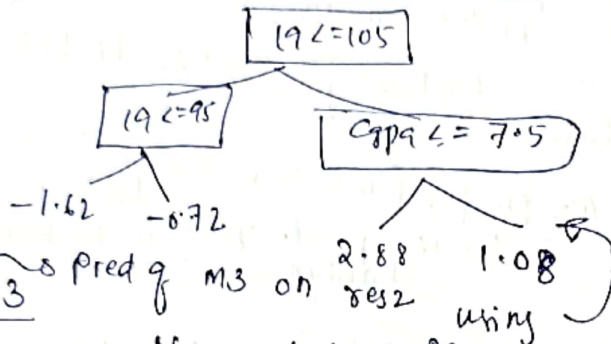
So we want the residual to be 0 or almost zero.

m3 will again be decision tree

Now the data will be

IQ	cgpa	res2
1	1	1

model 3 (Decision Tree)



ie m3 will predict the mistakes of m2 & more accurately if predicts the mistakes of m2 the better it will predict the Actual output (Salary).

Pred3
-1.62
-0.72
2.88
1.08
-1.62

Now Actual Pred of m3 on Salary

$$= \text{pred1} + \text{lr} \times \text{pred2} + \text{lr} \times \text{pred3}$$

Say input col :- IQ | cgpa
90 | 8

$$\begin{aligned} \text{then Actual Pred} &= 4.8 \times 0.1 \times (-1.8) + 0.1 \times (-1.62) \\ &= 4.8 - 0.18 - 0.162 \\ &= 4.458 \end{aligned}$$

lr = 0.1
same for all base models.

Again Actual $4.458 \approx 3$

But this is how Gradient Boosting works. So we need more base models.

Jupyter Notebook.

AdaBoost v/s Gradient Boost

1) max-leaf nodes :- in AdaBoost we use decision stumps ie 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 & they can be different & tells us ki konse model jyada imp hai. 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? $\frac{x}{\text{input}} \rightarrow \frac{y}{\text{output}}$

then ML models & fns/repr
 $y = f(x)$

if Data is:

$x_1 \mid x_2 \mid x_3 \mid y$

then $y = f(x_1, x_2, x_3)$

if the relationship is simpler then the rest is found easily (Linear, polynomial) but if the relationship is complex we use additive modelling. So in additive modelling we find the bigger fn y or $F(x)$ model.

as Sum of smaller fns: $y = F(x) = f_0(x) + f_1(x) + \dots + f_n(x)$

in Ensemble Learning

these fns 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 fn 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 fn

$$L(y, F(x)) = L(y, \hat{y})$$

$\downarrow$ L model ka predicted output

the loss fn that we will use here is Least square just $\frac{1}{2}$ of multiply tendency for mathematical convenience

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

x_i is input data (it can have multiple cols) & 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 case 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



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 $\therefore f(x) = f_0(x) + f_1(x) + \dots + f_n(x)$

Now to find $f_0(x)$.

we use $\frac{1}{2} \sum_{i=1}^N (y_i - \hat{y}_i)^2 = L(y_i, \hat{y}_i)$

& diff wrt 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.

ie $f_0(x)$ is not a Decision tree.

Note:- if we use a diff Loss fn then γ will not necessarily be a Mean.

tree fn is a Decision tree.

KNN. (K Nearest Neighbours)

Consider 100 students Dataset

Cgpa	IQ	placement
8	80	1
7	70	0
...	...	...

input → plot the input → output

* You are the average of the five people you spend the most time with.

(Jim Rohn)



The logic of KNN is ki ap 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) (100 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 ($K=three$) closest points.
Now we see Green (1) are in majority & 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

Experimentation (Better)

$$K = \sqrt{n} \quad (n = \# \text{ of observations} = \# \text{ of rows in the training data set})$$

Generally, we avoid even value for K since majority count mei dikkat hote.

Let us use cross validation.

for different KNN models with different K value & see the accuracy of each & compare them & find the best K value.

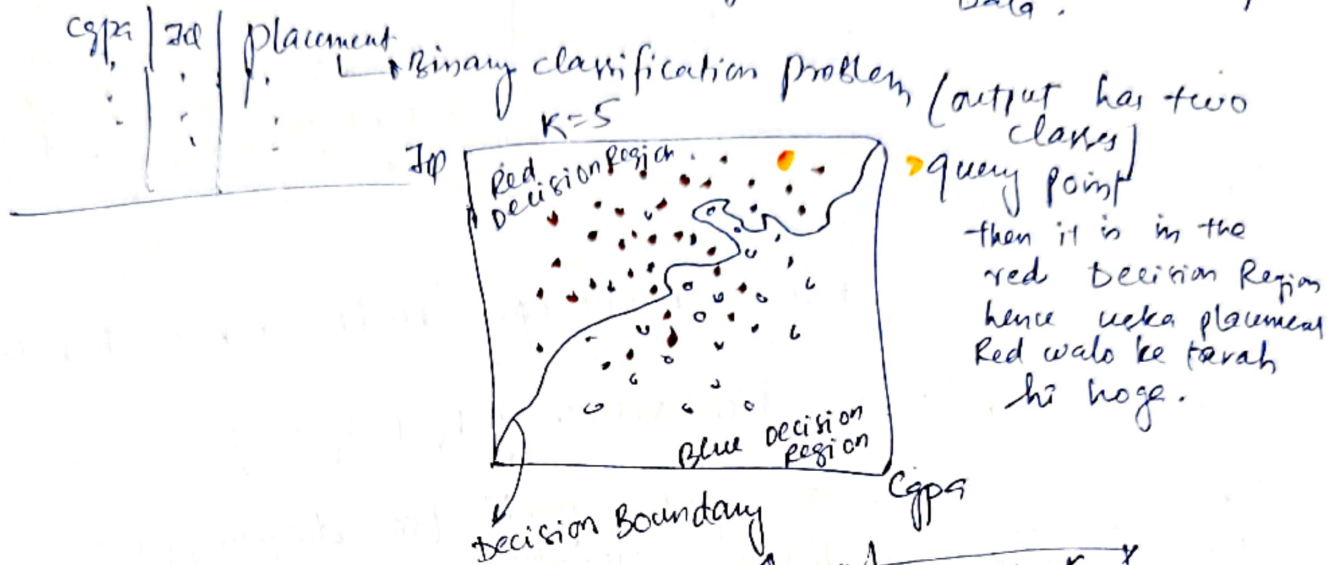
Decision Boundary / Surface used for in

- svm
- Log. reg
- Dec. Tree.
- Neural Network.

for 1D, 2D or 3D Data

for higher dim we cannot imagine the Data.

Consider 2D Data



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 predicts 0 → Blue & encode kando us meshgrid point ko

if knn predicts 1 → Red & "

& Aisa har meshgrid points ke liye karo.

& then show this points as pixels on an image

& we get the Blue & Red Decision Regions

* we have a library called mlxtend → used to plot Decision Surface.

Overfitting & underfitting

Page 100.

Say $K=1$.
(Low value)

Query point
It will be declared green although it should have been blue.

in this case the decision region will be divided into many parts

→ This is Overfitting

$K=n = \# \text{ of rows}$.
(high value)



minor changes ko hi capture kar rha hai model. (High variance) → alag alag dataset pe alag alag predictions karna.

This is a case for underfitting.
The model is unable to understand the patterns of the Dataset.

→ Here the output will be that class which is dominating i.e. jo class mei sabse jyada element hai wohi dominate karega & Decision Region bhi bas ek ussi class ka banega.
Hence output of any query point is the majority class.

Limitations of KNN

① we would not use KNN on large data sets
 $\# \text{ of rows/observations} = \text{in lakhs}$ & $\# \text{ of features/cols} = 100$ (say)
∴ query point ka har point se distance nikalna hai & that will take a lot of time & hence the prediction will be very slow.

→ KNN is a lazy learning technique

Aise ML algo jisme saavkaam prediction me toh bas data plot karna hai.

→ (slabhi distances calculate → sort them → use majority count)

② High Dimensional data ($\# \text{ of cols} = \text{say } 500$) then curse of dimensionality aagata hai i.e. in such high dimensions the distance concept is not reliable & hence kNN ke predictions bhi reliable nahi hote.

③ Cannot handle outliers nicely for lower values of K .
Which is obvious. KNN outliers ke nearest neighbours ko dekhkar hi predict kar dega.



4) Non-homogeneous feature scales

↳ Tabhi 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.

ie. if we have a query point the knn can predict ki wo konsi class mei fall karega but it cannot tell ki ye prediction karne mei kya ka jyada contribution tha ya nahi.

ie. 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 .
 (cannot tell ki konsi feature jyada imp hai prediction ke liye)

knn Hyperparameters.

- k -neighbours $\rightarrow k$ value (default = 5)
- weights :- (default = 'uniform')
 Other values :-
 → distance : all points in each neighbourhood are weighted equally. weight points by the inverse of their distance. In this case closer neighbors of a query point will have a greater influence than neighbors which are further away.
 → callable : a user-defined function which accepts an array of distances & returns an array of same shape containing the weights.
- 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.

• leaf size $\rightarrow$ default = 30

$\rightarrow$ used only when we are implementing the Ball Tree or Kd Tree algorithm

• P (integer, default value = 2)

$\rightarrow$ ~~the distance metric to use for the tree.~~

when $P=1 \rightarrow$ Manhattan distance (L1)

when $P=2 \rightarrow$ Euclidean distance (L2)

• metric :- (default : minkowski)

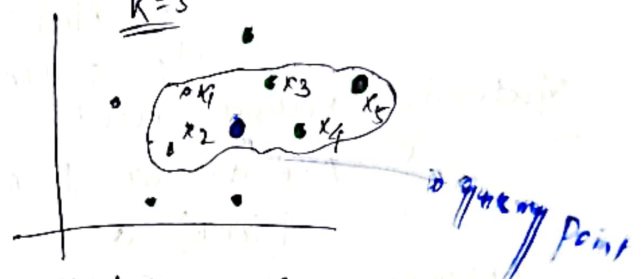
$\rightarrow$ with $P=2$ it is equivalent to standard Euclidean metric

• n-jobs $\rightarrow$ multi processing, default = 1 ; if we want to use all the cores of the CPU then use value -1.

Weighted KNN

$\rightarrow$ It 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.

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



Now we calculate weight:

Simplest weighing for is the inverse for

$$w_i = \frac{1}{d_i}$$

Based on weighing for Rule $\frac{1}{d}$ Distance $\uparrow$, weight $\downarrow$ yes.

weight

$$\frac{1}{0.2} = 5$$

$$\frac{1}{0.5} = 2$$

$$\frac{1}{0.7} = 1.4$$

$$\frac{1}{1.2} = 0.8$$

$$\frac{1}{1.5} = 0.6$$

Now weighted sum of blue points = $5 + 2 = 7$

weighted sum of green points = 2.8

Hence the query point will be labelled as Blue.

Note Simple KNN use kate for Green labelled kote ye query point (see the above Diag.)

Naive Bayes

conditional prob.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} ; \text{ provided } P(B) \neq 0$$

Independent Events

A, B are called independent if $P(A \cap B) = P(A) \times P(B)$

Aise events jo saath mei toh occur kar sakte hai but they are indep is ex ke hone ki chhote ko farak nahi padta.

Ex: Rolling a die twice, pehli baar 3 aya toh dusri baar 5 aane ki kya prob hai Ans is $\frac{1}{6}$ insp q pehli baar kya aya tha

In case of indep Events

$$P(A \cap B) = P(A) \times P(B)$$
$$\& \text{ we know } P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) \times P(B)}{P(B)} = P(A)$$

$$\therefore P(A|B) = P(A)$$

$$\text{slly } P(B|A) = P(B)$$

Mutually Exclusive Events:-

A, B are called mutually exclusive Events if

$$P(A \cap B) = 0$$

Ex: ~~Ek~~ Coin uchala $\rightarrow$ head & Tail dono saath mei aaye $\rightarrow$ not possible.

$$\text{Also } P(A|B) = \frac{P(A \cap B)}{P(B)} = 0$$

$$\text{slly } P(B|A) = 0$$

Bayes' Theorem

A, B are two events then

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}, \text{ given } P(B) \neq 0$$

↓
Posterior

↑
Evidence.

↑
likelihood

↑
prior

Intuition of Naive Bayes

Page 102.

Consider the Dataset of CSK.

loss	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 $P(W/A)$ & $P(L/A)$

& jiski ishi prob jyada hai wohi Answer hai Naive Bayes ka.

We know:- $P(A/B) = \frac{P(B/A) \times P(A)}{P(B)}$

so $P(W/A) = \frac{P(A/W) \times P(W)}{P(A)} = \frac{x \times 5/8}{1/8}$

A/W.

won	Mum	over	won
lost	Chenn	Sunn	won
won	Kol	Sunn	won
won	Chenn	Sunn	won
won	Mum	Sunn	won

∴ $P(A/W) = 0$

Now:- Naive Bayes is using a simple Assumption on the Bayes theorem

& The condition is:- $P(A/W) = P(\text{Loss, Mumbai, Sunny} / W)$

$= P(\text{Loss}/W) P(\text{Mumbai}/W) P(\text{Sunny}/W)$ (why we see later)

$= \frac{1}{5} \times \frac{2}{5} \times \frac{4}{5}$

∴ $P(W/A) = \frac{(\frac{1}{5} \times \frac{2}{5} \times \frac{4}{5}) \times 5/8}{1/8} = 0.2$

slly. $P(L/A) = \frac{P(A/L) \times P(L)}{P(A)} = \frac{P(\text{Loss, Mumbai, Sunny} / L) \times P(L)}{P(A)}$

using Naive Bayes

$= \frac{P(\text{Loss} / L) \times P(\text{Mum} / L) \times P(\text{Sunny} / L) \times 3/8}{1/8}$

$= \frac{1/3 \times 1/3 \times 1/3 \times 3/8}{1/8} = 0.3$

Hence $0.3 > 0.2$
∴ 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.
 $\rightarrow$ multiclass classification (k classes).

We need to find:-

$P(c_i | x)$ $\forall i = 1, 2, \dots, K$ and jossabhe jyada hoga
 wohi hum answer man
 lenge.

$$P(c_i | x) = \frac{P(x | c_i) \times P(c_i)}{P(x)} \quad (\text{using Bayes theorem})$$

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 ~~deno~~ so we only find the
 numerator.

so we only need to find $P(x | c_i) \times P(c_i)$ $\forall i = 1, 2, \dots, K$
 we know $P(A|B) = \frac{P(A \cap B)}{P(B)} \Rightarrow P(A \cap B) = P(A|B) \times P(B)$

$$\therefore P(x | c_i) \times P(c_i) = P(x \cap c_i) = P(x, c_i)$$

$$= P(x_1, x_2, \dots, x_n, c_i) = P(\underbrace{x_1, x_2, \dots, x_n}_A, \underbrace{c_i}_B) \xrightarrow{P(A \cap B) = P(A|B) \times P(B)}$$

$\forall i = 1, 2, \dots, K$

$$= P(x_1 | x_2, x_3, \dots, x_n, c_i) \times P(x_2, \dots, x_n, c_i)$$

$$= a_1 \times P(\underbrace{x_2, \dots, x_n}_A, \underbrace{c_i}_B) = P(A \cap B) = P(A|B) \times P(B)$$

$$= a_1 \times P(x_2 | x_3, \dots, x_n, c_i) \times P(x_3, x_4, \dots, x_n, c_i)$$

$$= a_1 \times a_2 \times P(x_3, x_4, \dots, x_n, c_i)$$

$$= a_1 \times a_2 \times \dots \times a_{n-1} \times P(x_n | c_i) \times P(c_i)$$

$$= P(x_1 | x_2, x_3, \dots, x_n, c_i) \times P(x_2 | x_3, \dots, x_n, c_i) \times \dots \times$$

$$P(x_{n-1}, x_n | c_i) \times P(c_i)$$

Now we take Naive Assumption

ie. 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

Hence:- $P(X_1/X_2, \dots, X_n, C_i) = P(X_1/C_i)$

sim. $P(X_2/X_3, \dots, X_n, C_i) = P(X_2/C_i)$

$\vdots$
 $P(X_{n-1}/X_n, C_i) = P(X_{n-1}/C_i)$

$\therefore$ 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 $i=1, \dots, K$. $= P(C_i) \times \prod_{j=1}^n P(X_j/C_i)$

$\therefore$ 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 ussi C_i class mei jaiga.

So $\hat{y} = \arg \max_{i=1, 2, \dots, K} P(C_i) \prod_{j=1}^n P(X_j/C_i) \rightarrow$ Maximum a posteriori rule - (MAP)

predicted output

Jupyter Notebook.

input

Numerical Data in Naive Bayes

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 Assumption that Height is Gaussian Distributed Random variable. using previous method sim. $P(H=185/M) = 0$ $P(H=185/F) = 0$

So to find: $P(H=185/M)$
 $\& P(W=170/M)$

Step 1 - find μ, σ of this

172	150	M
180	170	M
165	190	M
190	200	M

in case of Normal dist we know pdf (prob) = $\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

slly, for weight

find μ, σ for 150, 170, 190, 200.

& slly we will do this for female category:

to find $P(H=185/F)$ & $P(W=170/F)$

↓
 this method is called Gaussian Naive Bayes.

So what if the dist. is not Normal:

then we first plot (dist plot) and see ki kaise dist follow kar rha 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 again	positive
The movie was so 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 col. (using technique Bag of words, Bow)
- 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

Bag of words

Review column mei jitke shi unique words hai un sabke ek ek col bana dete hai.

Let's say we have only 4 unique words :-

	"Great"	"awesome"	"poor"	"pathetic"	Label (Sentiment)
Review 1	3	1	0	0	1 (pos)
Review 2	0	1	2	1	0 (-ve)

Now we pick up Review 1 and see in charo mei ki konga words aa rhe hai & konga nahi in Review 1, agar

Review 1 mei Great 3 baar aye, awesome 1 baar, poor $\rightarrow 0$ & pathetic 0 baar then we fill the row above for Review 1 as above.

& do 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 f_1	great f_2	boring f_3	label
3	1	0	pos(1)
0	0	2	-ve(0)
3	3	1	pos(1)

So what Naive Bayes do:-

$$P(\text{label} = \text{pos} \mid f_1=0, f_2=1, f_3=1)$$

$$P(\text{label} = -ve \mid f_1=0, f_2=1, f_3=1)$$

we know the formula for calculation of these.

Jupyter Notebook.

(Do it on Kaggle, Jupyter Notebook mei slow hojaga.)

input Query:-

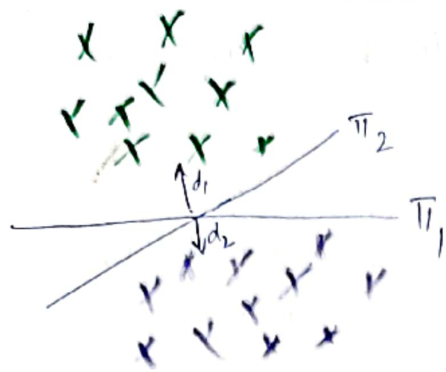
"The movie was great, but slightly boring" $\rightarrow [0, 1, 1]$

How to predict the Sentiment

This is nothing frequency count of the words "awesome", "great", "boring" in the input Review string.

& jo bada hoga wohi 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 kare & saath hi saath d_1 & d_2 ko maximise kare (for binary classification problem)

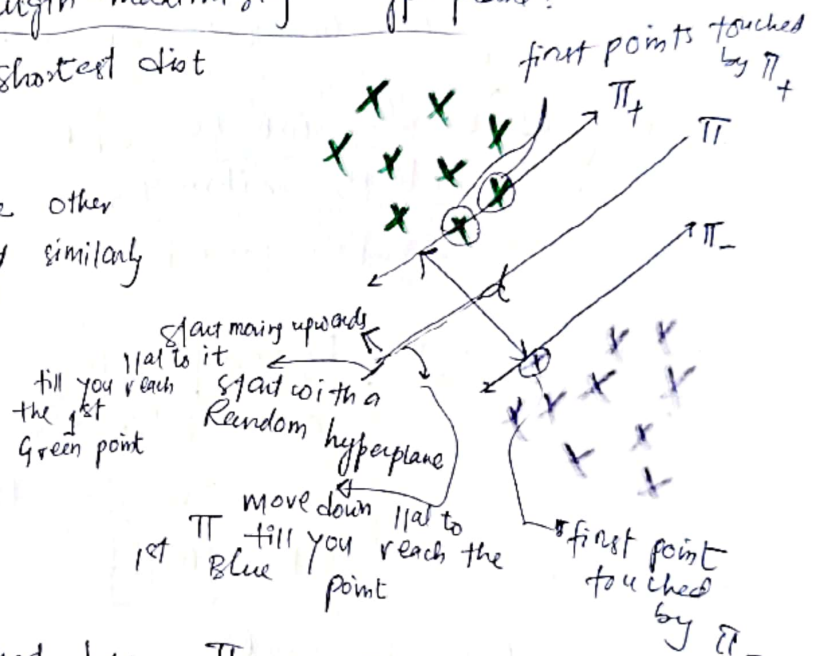
$\therefore$ The hyperplane is called Margin Maximizing hyperplane.
How to select the Margin Maximizing hyperplane:

Now we find margin = shortest dist

b/w π_+ & $\pi_- = d_1$

Then again we start with some other random hyperplane & proceed similarly & 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 π_+ & π_- are called 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.

→ eqⁿ of hyperplane.

$$\Pi: w_1 x_1 + w_2 x_2 + \dots + w_n x_n + w_0 = 0 \rightarrow w \cdot x + w_0 = 0$$

for 2-D. $w_1 x_1 + w_2 x_2 + w_0 = 0 \rightarrow$ line

for 3-D $w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 = 0 \rightarrow$ plane.

$$w = [w_1, \dots, w_n]$$

$$x = [x_1, \dots, x_n]$$

$$w^T x + w_0 = 0$$

∴ Assuming ki hamana hyperplane origin
 at pass kane, the eqⁿ is

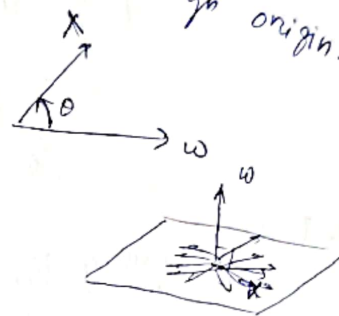
$$w^T x = 0$$

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 .

if $w_0 = 0$ then hyperplane passes through origin.



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

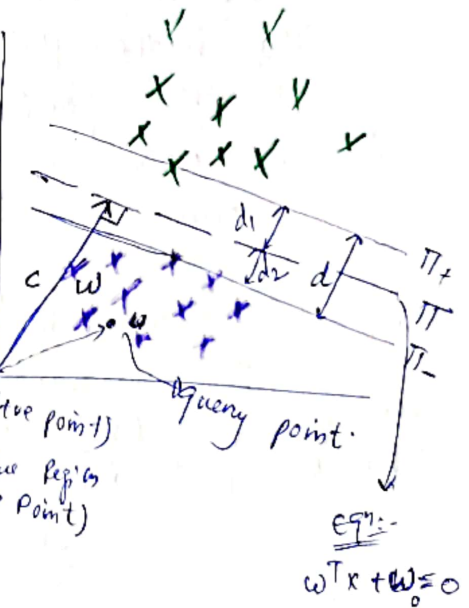
The query point. $\vec{u}$ is a vector
 And $\vec{w}$ we know from above
 is $\perp^r$ to hyperplane. Now we
 take projection of $\vec{u}$ on $\vec{w}$

$\vec{u} \cdot \vec{w}$ & we check

$\vec{u} \cdot \vec{w} \geq c \rightarrow$ Yes query point in green region (true point)
 $\vec{u} \cdot \vec{w} < c \rightarrow$ No query point in blue region (false point)

$$\vec{w} \cdot \vec{u} - c \geq 0$$

$$\vec{w} \cdot \vec{u} + w_0 \geq 0 \rightarrow \text{Decision Rule.}$$



In General: eqⁿ of $\Pi: w_1 x_1 + \dots + w_n x_n + w_0 = 0, w^T x + w_0 = 0$
 $\vec{u} = (u_1, \dots, u_n)$ 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 .

$$\hat{y} = \begin{cases} +1 & \text{if } \vec{w} \cdot \vec{x}_i + w_0 \geq 0 \\ -1 & \text{if } \vec{w} \cdot \vec{x}_i + w_0 \leq 0 \end{cases}$$

Note that: π should be exactly in b/w π_+ & π_-
 so $d_1 = d_2$ ($d = d_1 + d_2$) see previous diagram

Let the eqn of π_+ : $w^T x + w_0 = 1$
 & eqn of π_- : $w^T x + w_0 = -1$
 π_+ & π_- mei ye ~~same~~ same
 to π hai dono -
 (1, -1 to ya 5, -5 to
 ya kuch bhi ho,
 magnitude same
 hi rahega & turner
 same hi ayga).
 Parallel

Note: LHS of π_+ & LHS of π_- ko

>1 factor $\nexists$ multiply karnege to d shrink (use hoga)

<1 factor $\nexists$ multiply karnege to d $\uparrow$ se hoga

see in Desmos using 2D (line) graph

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 " " " blue " " " above it.

↓ we have to maximise d keeping this constraint in mind.

So the constraints can be written mathematically
for green points $(\vec{x}) \rightarrow \vec{w} \cdot \vec{x} + w_0 \geq 1$
& for blue points $(\vec{x}) \rightarrow \vec{w} \cdot \vec{x} + w_0 \leq -1$ (equality holds for support vectors)
Now we find & maximising it & selecting the convexp. hyperplane.

Converting the two constraints to a single constraint:-
Let $y = 1$ for not green with 4A

Let $y_i \rightarrow +1$ for pos (green points)
 $y_i \rightarrow -1$ for neg (red pts)

$\therefore$ The constraint becomes:-

∴ 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: $y_i (\vec{w} \cdot \vec{x} + w_0) = 1$.

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 $\Pi = \frac{\vec{w}}{\|\vec{w}\|}$

$$\therefore d = \frac{(\vec{x}_2 - \vec{x}_1) \cdot \vec{w}}{\|\vec{w}\|}$$

$$d = \frac{\vec{x}_2 \cdot \vec{w} - \vec{x}_1 \cdot \vec{w}}{\|\vec{w}\|}$$

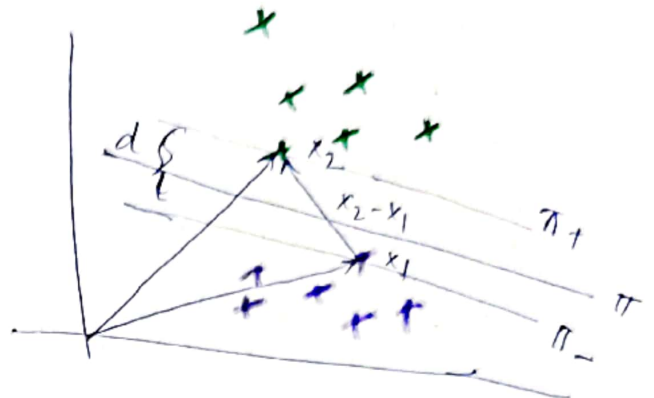
$$d = \frac{(1 - w_0) - (-w_0 - 1)}{\|\vec{w}\|}$$

$$d = \frac{2}{\|\vec{w}\|}$$

So we need to max d
So we need to minimize $\|\vec{w}\|$

So $\arg \max_{(\vec{w}^*, w_0^*)} \frac{2}{\|\vec{w}\|}$ such that $y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1$.

This formula we derived when there 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.



$x_2 \rightarrow$ support vector on Π_+

$x_1 \rightarrow$ support vector on Π_-

So for x_2

$$y_i (\vec{w} \cdot \vec{x} + w_0) = 1$$

$$1 (\vec{w} \cdot \vec{x}_2 + w_0) = 1$$

$$\vec{w} \cdot \vec{x}_2 = 1 - w_0$$

for x_1

$$y_i (\vec{w} \cdot \vec{x} + w_0) = 1$$

$$-1 (\vec{w} \cdot \vec{x}_1 + w_0) = 1$$

$$-w_0 - 1 = \vec{w} \cdot \vec{x}_1$$

Note again originally.

$$\Pi_+ : \vec{w} \cdot \vec{x} + w_0 = \alpha$$

$$\Pi_- : \vec{w} \cdot \vec{x} + w_0 = -\alpha \text{ Liya hola}$$

$$\text{then } d = \frac{2\alpha}{\|\vec{w}\|} \text{ hola}$$

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.

↳ not perfectly linearly separable.

→ Generally we will get data even that ek ya do points idhar ke udhar & udhar ke idhar honge. (outliers)
So we generally use Soft margin SVM.

In Hardmargin SVM we had

$$\arg \max_{(w^*, w_0^*)} \frac{2}{\|w\|} \text{ s.t. } y_i (\vec{w} \cdot \vec{x}_i + w_0) \geq 1$$

Also we know $\max f(x) \Leftrightarrow \min \frac{1}{f(x)}$

So $\arg \min_{(w^*, w_0^*)} \frac{\|w\|}{2} \text{ s.t. } y_i (\vec{w} \cdot \vec{x}_i + w_0) \geq 1$

$$\Downarrow$$

$$y_i (\vec{w}^T \vec{x}_i + w_0) \geq 1$$

In Softmargin SVM:

we have:- Regularization term (by comp. with Logistic Regression)

$$\arg \min_{(w^*, w_0^*)} \left(\frac{\|w\|}{2} + C \sum_{i=1}^n \xi_i \right)$$

($n = \#$ of rows/datapoints in the dataset)

ξ_i is calculated for every x_i (row/datapoint) in the dataset.

if the point is correctly classified then for that point $\xi_i = 0$.

Ek pot (green) point galat classified hai toh uska ξ_i is uska $\perp$ dot for Π_+ hyperplane & ek re (blue) point galat classified hai toh uska ξ_i is uska $\perp$ dot for Π_- hyperplane.

and minimising $\sum_{i=1}^n \xi_i$ means minimising the error (galat classified points)

So $\frac{\|w\|}{2}$ is called margin Error

& $\sum_{i=1}^n \xi_i$ is called classification Error.

& C (γ) is a hyperparameter (tuning knob) ($\alpha = \frac{1}{C}$; from Logistic Regression).

if C is big $\rightarrow$ means we are not caring about margin Error, we only want ki algo koi bhi point ko misclassify na kare.
and if C is vvv small $\rightarrow$ focus on margin Error & ignore classification Error matlab kuch misclassification chalega.

