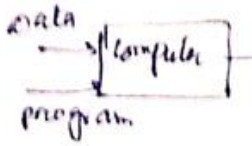
Machine Learning (ML):-

- 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:-

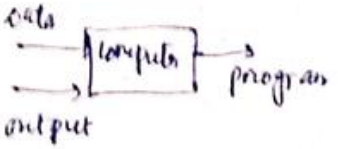


if we want to add Numbers:- → This program is explicitly coded to perform sum of 2 nos.

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) output again 3 nos and it's sum and so on (we provide some features) 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).

This is the data



→ 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 1 huge etc. present in the email then mark it as spam.

Now suppose the advertisement company knows about this and now instead of using huge ; 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 huge word to Big the ML algo will recognize it (since it has been trained on a large dataset) and will mark it as Spam

— Image classification:-

problems where there are so 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 feature 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.

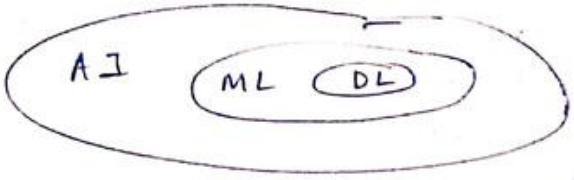
— Data mining

Kabhi kabhi kuch info in the data bohot hi jyada hidden hota 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 it's job we look for what patterns it has built, what are the keywords that it has predicted on it's 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 bare 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 iske saare data pe run karwa paye But with the evolution of Internet — These problems are solved.

AI V/s ML V/s DL



around 1950's ↑ **AI (Artificial Intelligence) :-** → Human intelligence is a very complex thing

(Alan Turing)

↳ Machines mei kisi tarike se intelligence dalna.

(Symbolic AI) → Lot's of if else

↓ isse we went to

(Expert Systems)

↓ what is it?

This is what we are trying over AI to DO ← Pattern Recognition (CV)

Creativity and Imagination

Emotional Intelligence

These are in Humans and are very complex. (Since they are very hard to quantify)

Human Expert → Knowledge Engineer → Knowledge Base → Inference Engine → User interface → User

- Human Expert → like chess master
- Knowledge Engineer → coder
- Knowledge Base → all the info about code
- Inference Engine → computer ke inferences Lena, user ke moves ke acc.
- User interface → computer
- User (not an Expert) → player

But Expert systems are very closed problems based if you give it a problem that has 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 doesnot 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

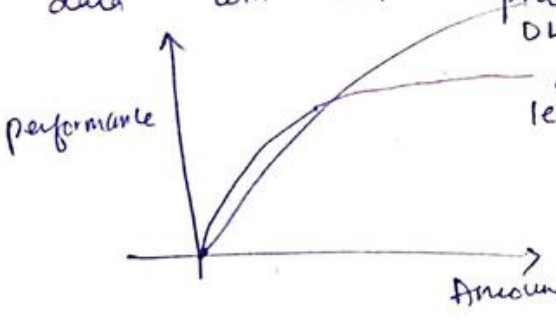
And this features will be created by the expert in the field.

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 stablizes → feeding more data will not improve the Accuracy.



- Image related Tasks
- Text related Tasks

DL works Better.

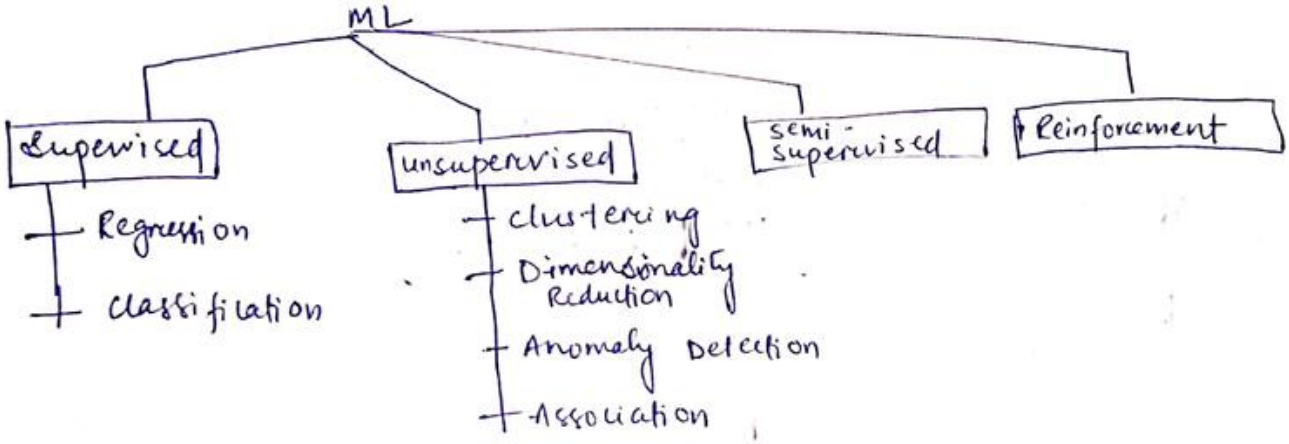
* Small data pe ML hi better accuracy dega
* on Large set of Data we use DL

TYPES OF ML

— ML ke types alag alag cheezo pe depend karte 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.
i.e. 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 pao then this is called Supervised ML

Suppose we have 5000 students Data (I/p + Output)

For each we have: IQ | CGPA | Placement Y/N →

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

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

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

→ Agar aapke paas aisa Dataset (input + output) hai & aapko input & output ke beech mathematical Rel^nship nikalna hai such that new input data ke liye output predict kar paye → Supervised ML

parts of Supervised ML → Regression, Classification

Data

Numerical
Ex:: Age, weight, IQ, CGPA

Categorical Data
- Gendre
- Nationality

Regression:

If we have a Data set (input + output) & if the output is numerical then that ML problem is called Regression.

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

IQ	CGPA	package
87	8.9	6L
111	7.9	4L

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 home price (Regression)

(ii) mails ke baare mei bohot kuch bataya hua hai you have to predict mail spam hai ya nahi (classification)

(iii) Saare wheater conditions ko dekkar predict karna ki aaj barish hogi ya nahi (classification)

(iv) Alag Alag images ko dekhkar predict karna hi image mei kitne dogs hai — (Regression)

Hence:

Supervised ML Input + output data
- output Numerical → Regression
- output categorical → classification

Unsupervised ML

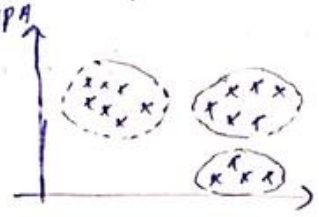
↳ Isme Basically Bas input data hoti hai output data nahi hoti.
Since output hai hi nahi toh yaha Task prediction toh ho hi nahi sakta since we don't know ki predict kya karna hai.

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

To in such case we do
- clustering
- Dimensionality Red
- Anomaly Detection
- Association Rule Learning.

Clustering :-

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



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

Infact Now we can ourselves create a Label (new col) Data
- 0 → more IQ + more CGPA category
- 1 → more IQ + less CGPA category
- 2 → Less IQ + more CGPA category.

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 Karna.

Dimensionality Reduction:-

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

Ex:- Suppose we have a large input Dimension/col's of which two are:- No. of rooms | No. of washrooms

then Dimensionality Reduction will combine this col's to a new col's Sq. feet area for room + bathroom (This is called feature Extraction) (Later)

Here we reduced the Dimension/col of input.

Algorithm used is PCA

Another use of Dimensionality Red is Vicualization :- 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 the 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.

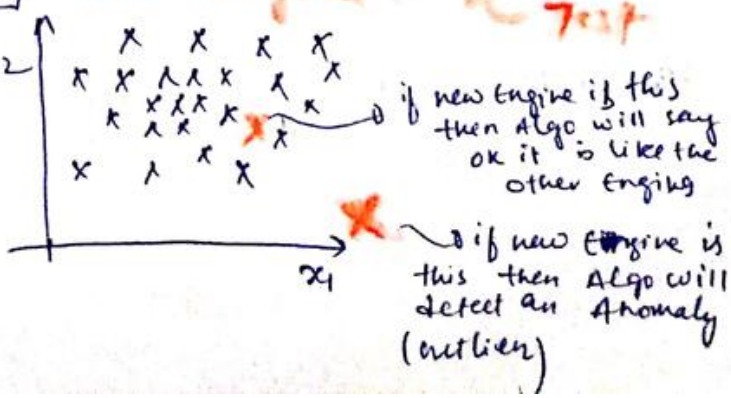
Eg Manufacturing mei defect, Credit card processing defect, Loan approval defect etc.

Data set : $\{x^{(1)}, x^{(2)}, \ldots x^{(m)}\}$

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 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 5 to 10 years and find patterns among the products bought like like out of 100 bills jab bhi 80 times milk khareeda gya toh 60 times eggs bhi liye gaye then hum milk and eggs ko ek saath agal bagal rakh denge (since there is a strong Association between milk and egg)

Interesting Casestudy in US :- Beer + Baby Diaper

SEMISUPERVISED → partially supervised & partially unsupervised

The label col^n (output col^n) data set on which Algo will be trained create karne mei for the human 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 karna difficult kaam hai but input data can be fetched from anywhere (web scraping)

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

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

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

REINFORCEMENT ML

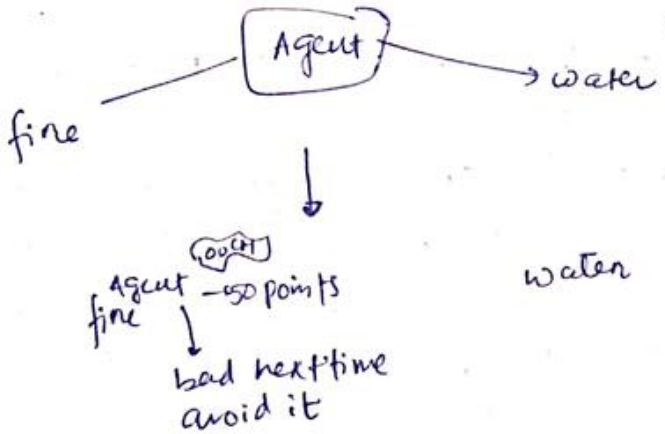
Hum Data dete hi nahi hai – system ke paas data hota hi nahi hai.

Humara Algo Exdum Scratch se start karta hai
Seekhna & dheere dheere improve karta jata hai

(Like us / Human beings) Like hum college life mei humare paas pehle se data nahi hai ki kaisa behave karna hai but hum dheere dheere galtiya karke seekhte hai.

Ex: Self Driving car :-
pehle se koi data nahi hai, wo road pe utar kar gradually learn karta hai ki kaise gadi chalni hai

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



(1) observe
(2) Select action using policy (Agent ka Rule BOOK Like it may have go to fire)
(III) Action
(IV) Get Reward or penalty (like fire has penalty)
(V) update policy (learning) (Like Don't go to fire)
(VI) iterate until optimal product is found.

* Company Deepmind (acquired by Google) made an agent (using Reinf ML) for Go Game (chinese Game more complex than chess) and that agent v the master of the game defeated straight 4 out of 5 games.

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

  Server ka environment jaha par code run kar raha hota hai usse hum production environment bolte hai and jab hum us code kar rahe hote hai apne machine pe toh hum usse development environment bolte hai.

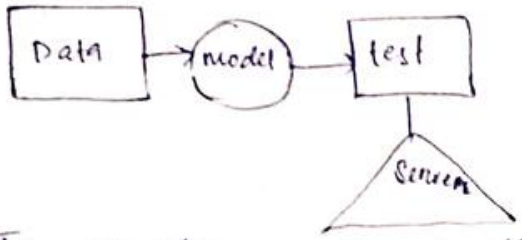
ML on servers behave differently & based on that ML is divided into two categories:

BATCH (OFFLINE) ML VS ONLINE ML

BATCH ML

(there is no incremental learning ie. chote chote hisso pe train nahi kar rahe)

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 :- Ek baar jo wo sikh gya wo sikh gya, model kuch naya nahi sikh raha hai, wo purane training ke 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 Dataset (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 nayi movies ko recommend nahi kar payega.

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

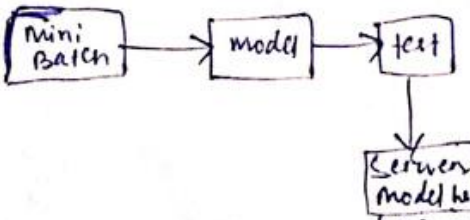
So in Batch ML after new data is added (in a frequency of 24 hrs or 1 week or 1 month or 6 months or etc.) we pull down the model Retrain it on the updated Data (old + new data) then test it & then deploy it again on the server. periodically...

Disad. of Batch ML

- Lots of Data (may be data bohot fast ↑se (update) ho raha hai & in Batch ML Limitation is ki pooore 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 sei aapka instant connectivity nahi hai toh updated data pe periodically retrain karna difficult hoga)
- Availability (let's say hum ek social networking website par kaam kar rahe hai & humne ek model banaya to newsfeed generate karta hai on basis of users interest & we have used Batch ML on 24 hr period ab agar achanak ↑ ek breaking news aagayi ya koi baat fail gyi (jisme hum interested hai & search kar rahe hai hence new data feed kar rahe hai model ko) then hamara model usse instantly show hi nahi karega, since wo 24 hrs baad new data pe train hone ke baad hi ye newsfeed pe show karega) & then hamara newsfeed us breaking news ↑ bhar jayega jo ki ab itna important bhi nahi hai.

ONLINE ML

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



continuous inflow of new data

how to doing prediction on old data & newdata & saath hi train & learn bhi kar raha hai on new data and hence with new data model ka performance improve karta jata hai.

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

- Youtube feeds personalize 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

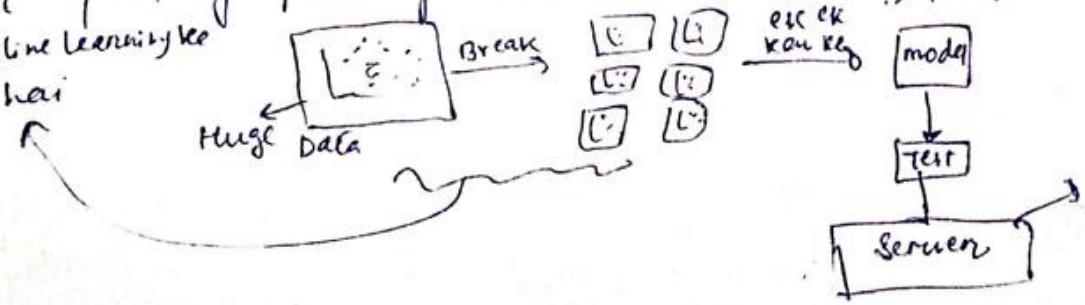
River is a python Library for online ML.
It is a result of a merger b/w creme & scikit-multiflow
River's ambition is to be the go to ~~ambition~~ library for doing ML on streaming data.

Tricky Thing to set

— Learning Rate: in online ML, Learning Rate ye decide karta hai ki kitna frequently hum model ko new data pe train karenge since new data inflow toh consistent hai Humme aisa Learning Rate rakna 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 bhi karte hai just concept online Learning ke tarah hai



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 hojaiga.

Sol^n 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

Features	offline Learning	online Learning
Complexity	Less complex as the model is constant	Dynamic complexity as the model keeps evolving over time
computational power	Feed-computation, single time batch-based training	continuous data ingestions results in consequent model refinement computations
use in production	Easier to implement	Difficult to implement & manage
Applications	Image classification on anything related to ML – where data pattern remains constant without sudden concept drifts	used in finance, economics, health where new data patterns are constantly emerging
Tools	Industry proven tools :- E.g. Scikit, TensorFlow, Pytorch, Keras, sparkMlib	Active research / new project tools- Eg :- MOA, SAMOA, Scikit multiflow, stream DM

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

- Instance Based ML
- Model Based ML

Learning in ML → memorizing → Instance Based Learning
→ generalizing (concept/underlying principle/pattern, samajhna) → Model Based Learning

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

LAZY LEARNER

— Instance Based Learning

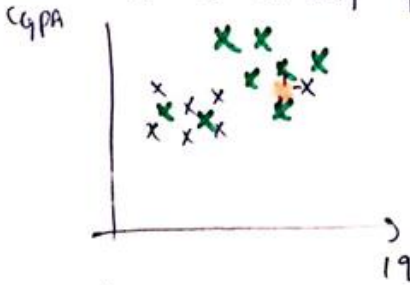
Consider we have Dataset :-

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 these are 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 based on the itself distance 103, 7.5 as shown above then we look at it's nearest neighbours (3, 5 etc depends on us) and we find near the query point which type of neighbours are more (here green) so we say the output of the query point will be like green neighbours i.e. yes.

↓

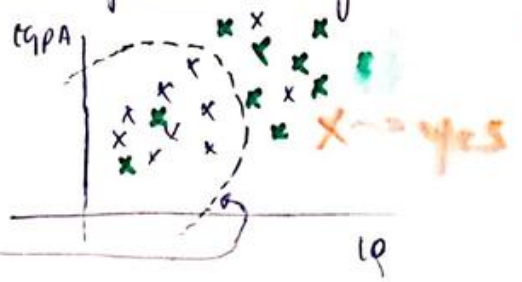
This is the logic of the KNN Algorithm (K nearest neighbour)

Note in Instance Based Learning, jab tak naya point nahi aaya tha tab tak apne model ne kuch kiya hi nahi jab uske paas ek query point (input new) aaya tab usne 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
⋮	⋮	⋮



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 (b/w input & output) → ye apne model ne data ke upar run ya train karke sikha. → called model parameter.

Most of the ML algo use this approach → finding a mathematical Ref b/w input & output of result is a decision function (esp in classification problems)

☆ Now after getting the Decision fun, even if we remove all the training Data set the Algo will be able to predict an output for a given input Data but this is not the case with Instance based Learning where we need the Dataset to find the nearest neighbours

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)	INSTANCE BASED
- Prepare the data for model training (data preprocessing)	- Prepare the data for model training (No diff here)
- Train model from training data to estimate model parameters i.e discover patterns	- Donot train model. pattern discovery postponed until scoring query received. (scoring query → new input data / query point)

MODEL BASED

- Store the model in suitable form
- Generalize the rules in form of model even before scoring instance is seen (scoring instance = 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 uss ko store karna hai)

INSTANCE BASED

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

CHALLENGES IN ML

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

- **Insufficient Data / Labelled Data**

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 & 10^6 rows to 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 bohot jyada data hai toh fatak nahi padta ki hum konsa Algo use kar rahe hai, this is called the unreasonable effectiveness of Data.

But we have mostly small or medium data ∴ Algo matter karta hai.

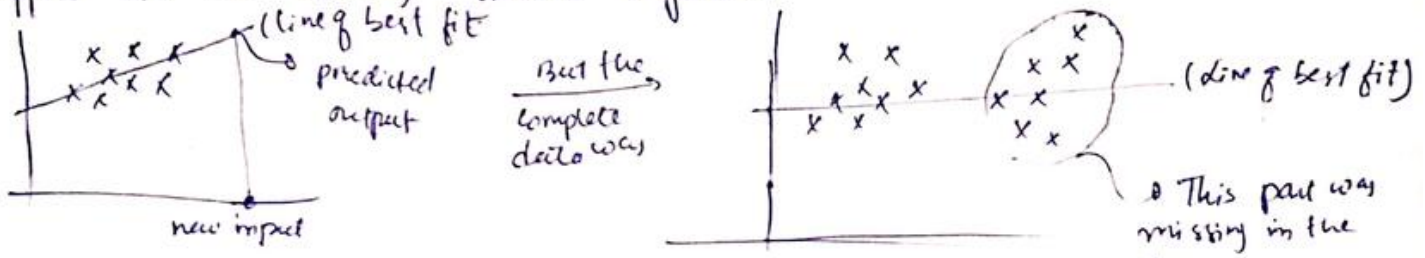
Again:- Agar hum data collect kar bhi lo. the labelling it (like cat hai ya nahi in picture) is a tedious job.

Data mila → kya wo sufficient hai — sufficient hai toh kya wo labelled hai.

— Non Representative Data:-

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

Suppose we are solving linear Regression



Kabhi kabhi jab hum Data gather karte hai toh sabhi jagah se gather na karke kisi ek jagah se utha lete hai then this data problem ka proper Representation nahi hota

Like:- we want to survey ki next T20 worldcup konsi team jitegi:- Agar hum bas indian logo se 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 karna chahiye jo teams khel rahe 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 Cause waha bhi mostly indians hi hai — This is called Sampling bias

Sahi representation of Data ke liye:- 100 Ind, 100 Aust, 100 Eng ... se pucha hoga.

(imp)

— Poor Quality Data :-

Data mei bohot saare outliers hai, missing values hai, abrupt values hai, alag alag format mei values hai and in sab cheezo ko sahi karne mei bohot 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 rahe apne analysis mei — toh apna model accha perform nahi karega

IN ML :: There is a phrase :-

Garbage IN, Garbage OUT :— Agar ML model ko kachra dhalega toh output kachra hi milega.

Suppose. we want to Analyse which age group are more fit & data we have (columns)

Age | weight | Height | Location — the location col will not add anything to our analysis hence we remove it.

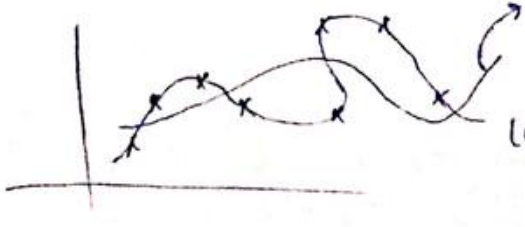
Kabhi kabhi hum do features ko mila ke ek single feature bana dete hai (yaha BMI) jo sahi analysis dega → And this is called feature engineering

— Over-fitting : (Big villian in ML Algos)

Humne ek ML model ko kisi data pe train kiya and us model ne data ko rat liya (matlab concepts/patterns 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 karna aata hai.

Ex: Guwahati gaya — saw movie — ticket — 500 ₹ — we assume Guwahati is costly But aisa nahi hai hum bas movie ticket ke basis par pure city ke prices predict kar rahe hai

Silly Even ML Algo training data set ko dil se lagaa leta hai ki jo training data set mei hai wohi sahi hai and baaki sab bhi waisa hi hai

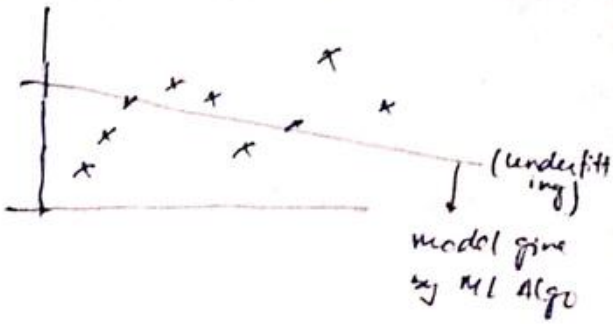


→ Aisa kuch Actually fit aana chahiye tha.

(over fitting) → jo hamara ML model ne banaya it is trying to touch every data point in the training Data set

— underfitting. (opposite of overfitting)

Toyo na training data pe accha result dega ya 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 so 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 / Deployement

Batch Learning wali problem (Ek baar model train hokar server pe gya toh wo update nahi hota - update karne ke liye wapas offline lao - new data pe train & test karo & then phir upload on server)

issme deployement mei bohot 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 they are for softwares)

— Cost Involved

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

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

* just like Devops Now there is something need 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 khoro ki konsa product ka sales increase hoga during the SALE.
And Data of previous 5-6 yrs of Greant Indian Festival sale are given to data scientists and they use ML Algo, Data mining to find insights ki konse product ka stockup karna chahiye & konse ka nahi.
(Note: yaha small mistake can lead to loses 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 (buying pattern), ki ye aadmi konsi cheeze jyada khareedta hai (ya 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 ↔ ML Algo / Loan officer — Analysis by ML algo — this profile is compared by the past defaulters (jinhone loan nahi chukaya) and agar 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 to loan officer who checks the profile manually & loan is sanctioned if good.

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

* Share market / Trading.

— Transportation — OLA

* Din ke kisi particular time pe point A से point B jane ka price bohot high kar diya jata hai jabki wohi 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 customer but those cab outside of Red region why will they go extra distance to pick customers at the same price to OLA says if they pick customers to cabs outside Red Region 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 goes due to good 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 the use IoT (Internet of Things) censors that monitors the different metrics of the arm (Temp, RPM etc.). And if an arm fails — it fails gradually & a signal is sent that a particular arm may fail to work and they are fixed even before it fails.

— Consumer Internet - Twitter (started in 2008)

Twitter was not earning money initially — it's investors were scared - for a long time (in NLP)
- then the head of Twitter decided to do sentiment analysis of the tweets & earn money from those Analysis
Ex: #WB elections 2021 tweets ko ek taraf kar dega and un tweets ke sentiments ko evaluate karega (modi vs Mamta) (social media pe jab kuch bolte ya likhte hai toh wo apke personality ko hi kind of extrapolate karta hai) (so 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 ammero से lekar) and these Stock Brokers un companies mei paisa lagayegi jo already predicted winning party ko support kar rahi hai. Ab agar seen mei predicted party jeet gya toh local public bhi unki company mei paisa lagayegi & stock prices goes and stock Brokers ko profit hoga since they already have those shares

Machine Learning Development Life Cycle (MLDLC / MLDC)

↳ just like SDLC (Software Dev. Life Cycle)

So far we focused on why & what.
Here we focus on [How?]

MLDLC — is a guidline 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 lagega, 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 itna easily available nahi hota.

Data → csv file
→ from some API in json format
→ webscraping
→ Database → we make a data warehouse (ETL: Extract Transform Load) — then fetch data
→ Big data in tool spark in clusters

- **Data Preprocessing**

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

— Remove Duplicates
— Remove missing values
— outliers handling
— scaling (standardization)

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

— EDA ** (jitna time yaha 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.
Here we make lots of experiments with Data, (extract the relnships hidden inside the data

- Visualization by plotting Graphs
- univariate Analysis (bas col ke upar independent Analysis) mean, std, variance, curve etc
- Bivariate Analysis (do 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)

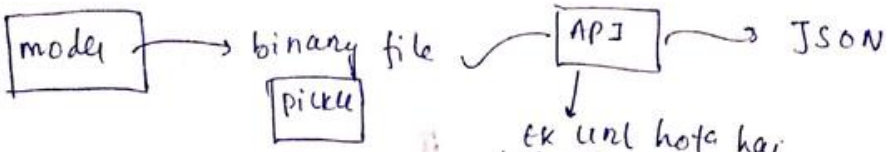
*** — Feature Engineering & Selection
Feature engg: existing features mei kuch intelligent changes karna so that Analysis easy hojai.
Features matlab — input columns
Features are imp since output input cols (features) pe hi depend karta hai
Idea is kabhi kabhi kuch naye cols (feature) hum create karte hai apne hisab se taaki apna analysis easy hojai
Ex: # of rooms + # of bathrooms → combine → eg: feature for rooms & bathrooms (new input feature)
previous input features
Feature Selection: kabhi kabhi Data set mei bohot hi jyada features hote hai (100 or 200) then hum saare features ko lekar aage nahi badh sakte because jaruri 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)

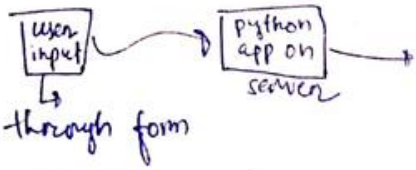
— Model Training, Evaluation & Selection
After we have the proper Data set, we bring different ML Algos and feed them the Data & train (∵ we never actually know konsa Algo kitne types acche nikalega kisi particular Data ke liye)
(naive Bayes performs better on text data but hosakta hai ki koi aur Algo iske acche perform karjai you never know)
and sabko Evaluate karte hai ki unka 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 Hyper 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 banate 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)





→ 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 ke paas ya software/ model bhejte hai and ask for feedback

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 pichle 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 aarahe hai to unko divide karna same ya kam time mei request serve karne)
- How frequent we need to Retrain (Rotting prevent karne ke liye since overtime data Evolve karne lag jata hai)
  - this needs to be automated
- Automation setup ki agar model fatgya then usse roll back karke wapas se live 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 (nearly hum characters & strings bhi store kar lete hai) (n-D array)

Ek: Vectors, matrices are Tensors.

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

General term Tensor

0 dim ←

* 0D Tensor / Scalar :-

(2) (3)

Scalar — 0D Tensor

To create 0D Tensor:

```
import numpy as np
a = np.array(4)   → To see it's dim a.ndim
print(a) → array(4)
```

This is a Scalar

* array → 1D
2-D arrays → array ke andhar array
3-D arrays → array ke andhar array & uske andhar bhi array & so. on.

* 1D Tensors :-

[1,2,3,4] → vector, 1D array, array, 1D Tensor
↳ a.ndim = 1

* in Tensors term :- what is Axis :- → means the dimension

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

↳ arr.ndim = 1

0 D Tensors → Scalars (9), (2),

Note: 1D Tensor — collection of multiple 0D Tensors → [1, 2, 3, 4]

2 D Tensor — " " " 1D Tensors → [[1,2,3], [3,4,5]] (matrices)

3-D Tensors → " " " 2D Tensors → [[[1,2,3], [3,4,5]]]

* 2 D Tensors / Matrices :-

1D Tensors: [1,2,3] [4,5,6] [7,8,9]

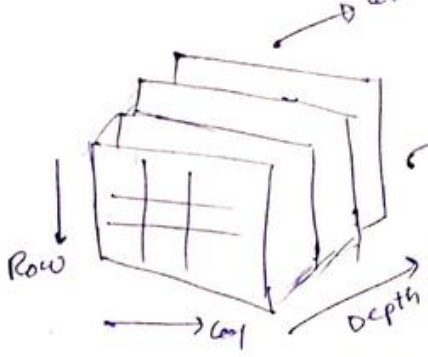
2-D Tensor matrix → [[1,2,3], [4,5,6], [7,8,9]] (col →, Row ↓)

n-dim = 2 = # of axis = Rank

creating 2-D Tensor using numpy:-

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

* 3-D Tensors :-



3-D Tensor → 3 axis (Row, col, depth) = 3 Rank

Sly :- 4-D Tensors → collection of 3-D Tensors.

IN ML 0 to 5D tak hi kaam ayega

vector: (1D Tensor) [1,2,3]

# of Axis = 1 → rank

Shape = (3,)

Size = 3

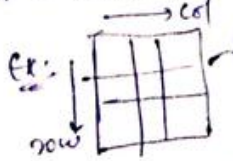
Rank, Axes & Shape, Size of Tensor

Rank = # of Axes = # of dimension = .ndim

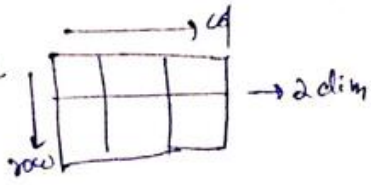
Shape ⇒ (row ki length, col ki length)

Ex:  → 2 dim

Shape = (4,2)



Shape = (3,3)



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 = np.array([1,2,3,4]) is 1D

∴ it's shape is a single value iterable

a.shape = (4,)

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

a = np.array([[1,2,3,4]])

shape = (1,4)

PRACTICAL EXAMPLES

— Examples of 1D Tensors

Students data set : (10000)

Say we have only two states WB = 0, 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] → 1D Tensor / 1D Array / Vector

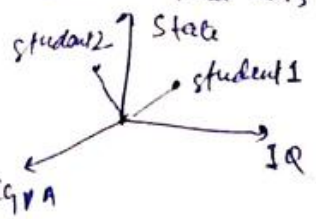
→ ndim = 1 & Tensor/array

But it is a 3-D vector

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

∴ in terms of vectors it is a 3-D vector

but when we represent them as numbers in a Tensor/array → it is a 1-D Tensor/array



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

The placement col : [1 0 0 1 1 1 0 1 ...] → 1-D Tensor

↳ 10000 entries

Examples of 2-D Tensors

Students Data (10000 students)

cgpa | iq | state | placement

input data : $\begin{bmatrix} [9.2 & 101 & 0] \\ [8.7 & 97 & 1] \\ \vdots & & \\ [- & - & -] \end{bmatrix}$ → 2 dim Tensor.

→ student 1 input data

→ 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 ...] → 1-D Tensor or 10000 dim vector

* Examples of 3D Tensors

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

Ex : Hi Gautam
Hi Neeraj
Hi Jatin

→ This is the text we provide for training to the model & we know ML Algo is mathematics (doesn't 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 → Hi
0	1	0	0 → Gautam
0	0	1	0 → Neeraj
0	0	0	1 → Jatin

Now : Hi Gautam → [[1,0,0,0], [0,1,0,0]] → 2-D Tensor ∵ collection of 1-D Tensors

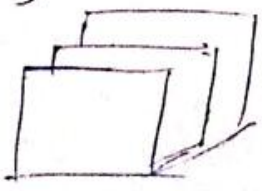
[[1 0 0 0]
[0 1 0 0]]

Sim. Hi Neeraj → [[1 0 0 0]
[0 0 1 0]]

Hi Jatin → [[1 0 0 0]
[0 0 0 1]]

→ ndim = 3 = # of Axis.

∴ Hi Gautam
Hi Neeraj ⟹
Hi Jatin



→ 3-D Tensor

3 x 2x4 → shape

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

Aisa Data jo EK frequent time period pe collect karte ho

Ex:- Sharemarket Data

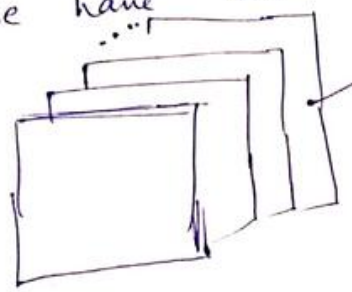
Kisi stock ka Highest | Lowest price Every day for 365 days.

Day1 [(– –)]
Day2 [(– –)]
⋮
Day 365 [(– –)]

→ 2-D Tensor, shape (365, 2)

→ 1-year

Now: say we have data of 10-years Data



→ 3-D Tensor

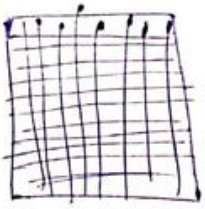
shape = (10, 365, 2)

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

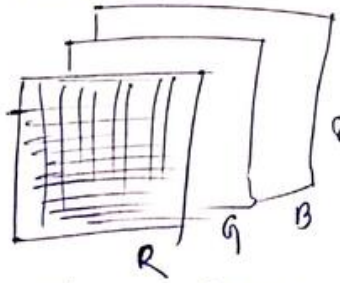
~ Examples of 4-D Tensors

(Collection of Image) data → in Computer Vision Domain

Image Processing.

Any image is a collection of pixels :-  & har pixel ko wajah se hume koi cheez dikhayi deti hai

1 Colour Image :-



→ 3-D Tensor

shape = (3, 1200, 800)

R G B

What if we have 50 colour image :- shape :- (50, 3, 1200, 800) → 4-D Tensors

(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 60 fps (frames per sec) i.e 1 sec me 2 diff images ko hi classify kar sakta hai

1 video → 60 sec at 30 fps, 480p (480 x 720 Resolution) → each image has 3 shades / channels

1800 images ← (1800, 3, 480, 720)

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

each shade has this many pixels

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 rahe hai Netflix unko rok le. ✓

Churn rate:- jitne bhi apke platform ke given period (monthly/yearly) kitne customers ke jaa rahe hai.
means churn rate ↓ users hai unme ↑ in a platform chor

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

Say Churn rate of Netflix 4%.

Hence:- Increase Revenue ⇒ Churn rate 4% → 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 to will be leaving the platform and then we need to stop them.
(by offering instant discount on next month subscription)

Problem is user will leave the platform Y/N → Supervised → classification problem.

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) → Hence sabhi ko barabar discount kyu dena

∴ kam man hai chor ke jaane ka → kam discount do
jyada " " " " " " → jyada " ".

∴ Now we would 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?

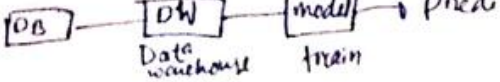
1./ Watch time ⟶ Agar wo browsing mei jyada time laga raha hai toh apne man pasand ke cheeze nahi mil rahi
2./ Search but did not find (matlab mei kitne searches kiye & nahi mile)
3./ Content Left in the middle → means content is not much tempting
4./ Clicked on recommendations (order of recommendations)

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

5./ Metrics to Measure:- ⟶ jo hum kar rahe hai woo 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 chor kar gaye isme diff bohot kam hai then model sahi hai.
→ jitko predict kiya chor kar jaiga wohi gya
— means model sahi hai.

6./ online vs Batch:-

Here online will be better since bohot saare factors Churn rate ko affect karte hai & wo upar niche hota rehta hai.



(However we should also consider Batch Learning ki han hafte phir se train karenge).

7./ Check Assumptions:

- cols | features decided → are they available
- will the model work same for US people and Indian people (geographical based changes in model)

2./ Data Gathering:-

Note:- "Bad Algo with huge Data will work better than / perform
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 nikalege then it is SQL
- fetch Data from An API
- Webscraping

Laptop → folder Machine Learning on Desktop → Jupyter Notebook.

3./ understanding your Data:

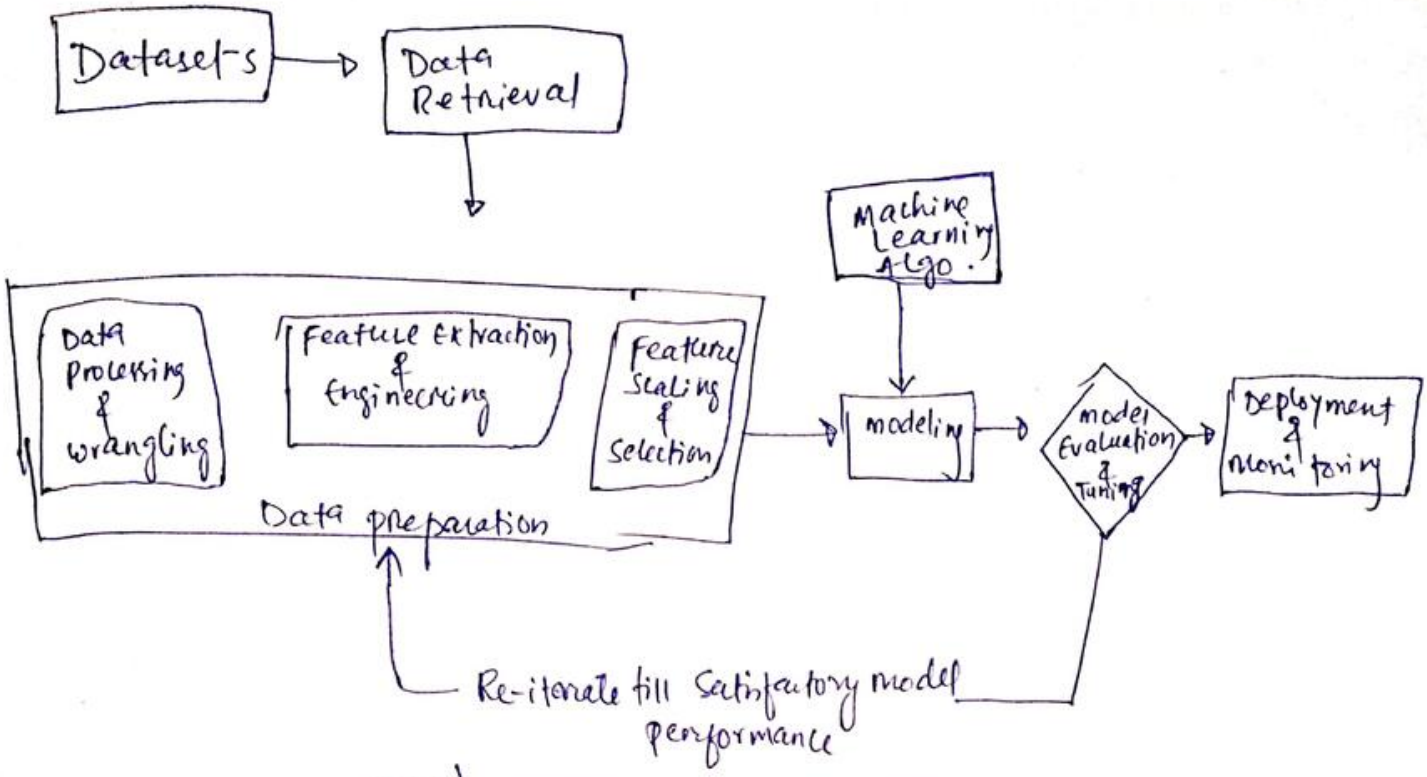
- Asking Basic Questions
- EDA
  - univariate Analysis
  - Multivariate Analysis
- pandas profiling (Automation for EDA)

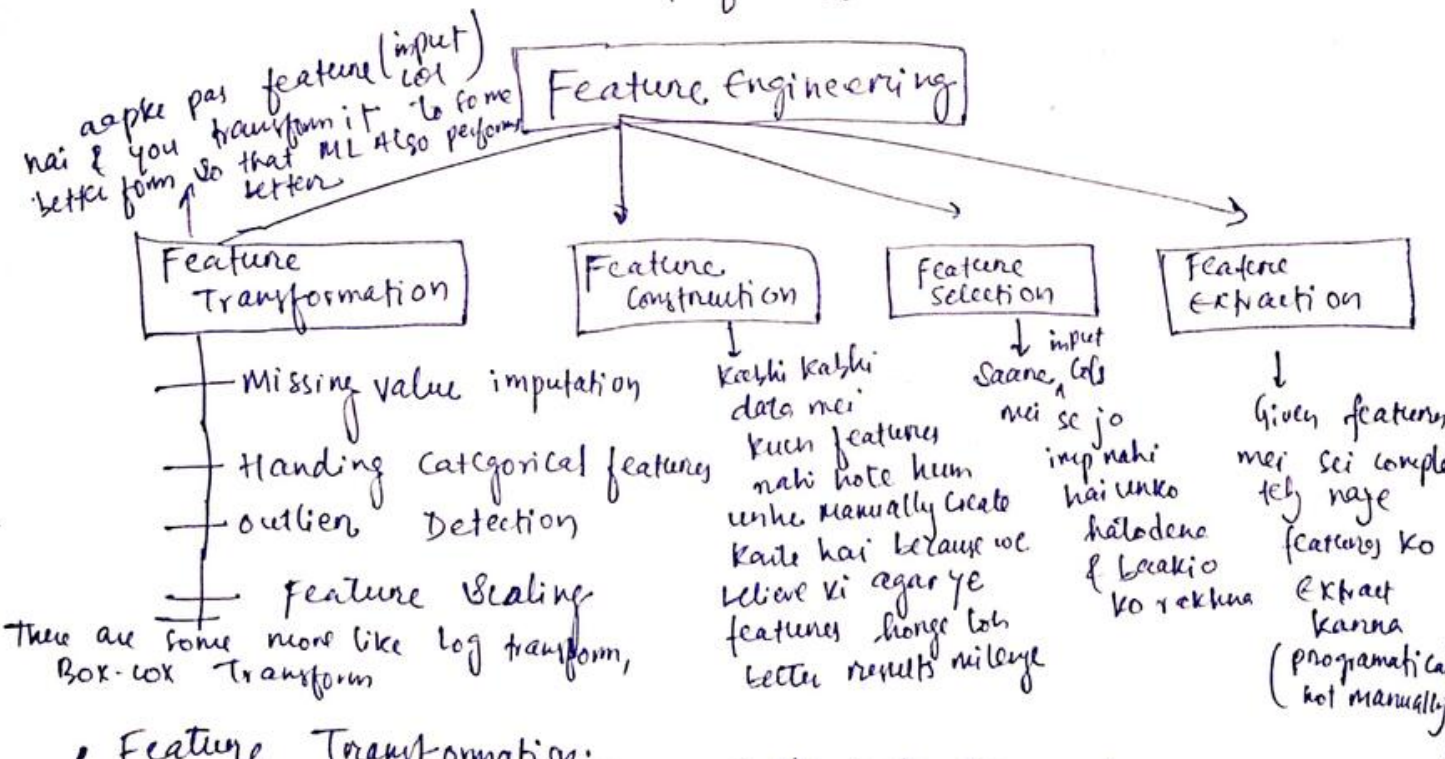
4./ ⭐⭐⭐ Feature Engineering: (It is an art sab apne apne tarike se karte hai).

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

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

Av age = 26 → data NAN mei 26 bhar denge

- kabhi kabhi bohot kam values missing hoti hai then hum row remove kar deta hai ∴ farak nahi padega tab.

Kabhi kabhi bohot jyada Data missing hota hai toh we need to fill → we can replace missing values by Mean, Median in case of Numerical feature & by Mode (most frequent category) in case of categorical Data. There are also other ways to fill NaN.

— Handling Categorical values

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

Categorical values

one Hot Code →

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

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

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

Sometimes we convert Numerical values to categorical :.

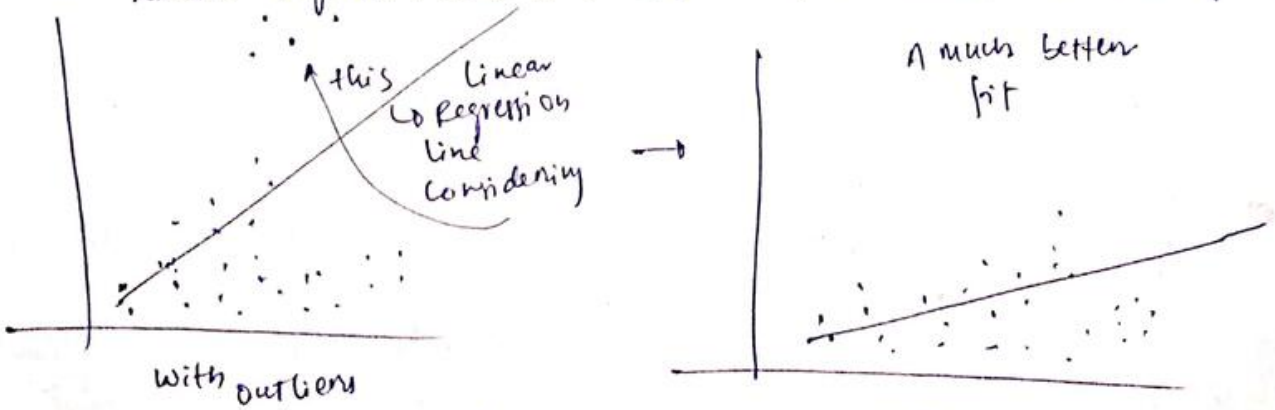
Like :- Age (Numerical) — This is called Billing.

0 – 12 | 13 – 19 | 20 – 35 | and so on.

Child | Teenage | Adult

— 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

Age	Salary
44	72000
27	48000
30	54000
38	61000
40	93000

Eg: 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 feature will be dominant in the distance calculation & hence age will not be able to control / monitor the ML behaviour

So Generally we scale the features i.e. features ko ek given range mei le aate hai -1 to 1 (min max scaling are some techniques, standardization, Normalization, min absolute scaling)

- **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 family to categorical feature

family (categorical feature)
0 → alone
0-4 → small family
4> → Large family

Some techniques are:
Feature splitting
Feature Grouping.

- **Feature Selection.**

Bohot saare input cols hai & aap important input cols ko select karte ho.

MNIST data set → Deep Learning (Convolutional Neural Network)
Lo 50000 images of handwritten digits.
& Each image is 28 x 28 resolution = 784 pixels
most pixels are white & the black/grey pixels se 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.

— Feature Extraction → mostly when we are dealing with high dimensional data etc. (many input cols.)
→ this is not feature extraction :: yaha tk completely new feature create hua.

ex: House pricing

#q rooms	#q Bathrooms	price
:	:	:

suppose kisi ek input col ko hatana hi hai & ek hi col rakhna hai

sq. feet area	Price
:	:

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

Feature Scaling: → It is a technique to standardize the independent (input) features present in the data in a fixed range.
→ It is the last thing we do in the feature Eng. pipeline. ML model ko data dene se 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 Distances)

(50, 83000) (27, 47000)

$(y_2 - y_1)^2 = 1225000000$

$(x_2 - x_1)^2 = 529$

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) Age → Age'

$x_i \rightarrow x_i'$

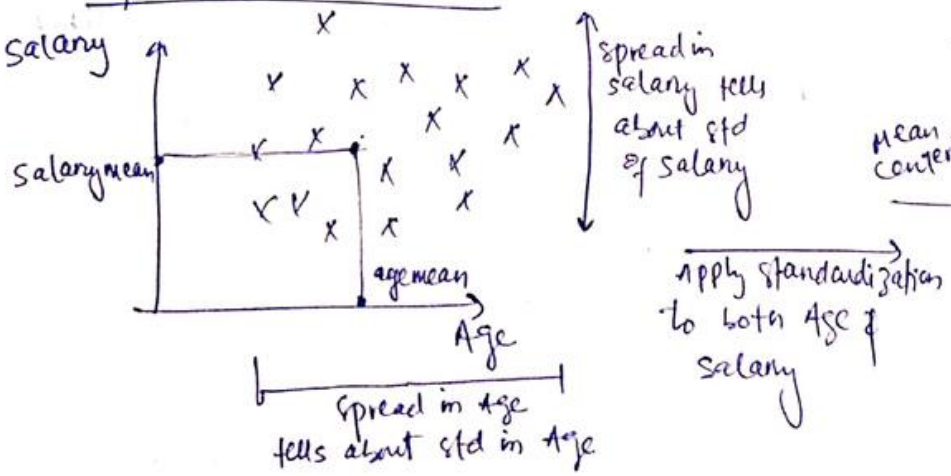
$$x_i' = \frac{x_i - \bar{x}}{\sigma}$$

$\bar{x}$ → mean of age col

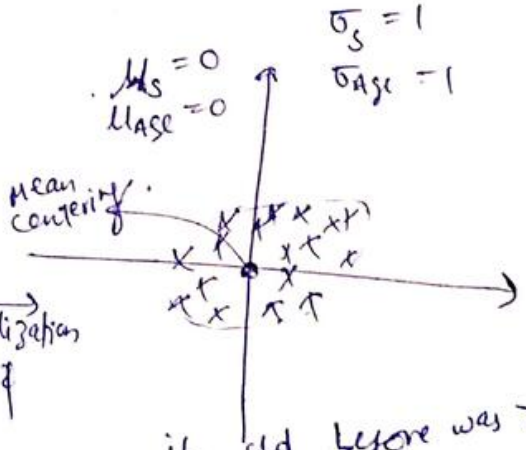
σ → std of age col

Note After standardization the mean of Age' = 0, std of 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 → 1. Feature Scaling → 1. standardization

When to use Standardization?

Algorithm(s)	Reason of applying feature scaling :-
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 donot do standardization:-

Decision Tree, Random forest, Gradient Boost, XG Boost ↓ inpe standardization ka koi farak 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 cols → It has Numerical value. It emits (magnitude)
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
54
⋮

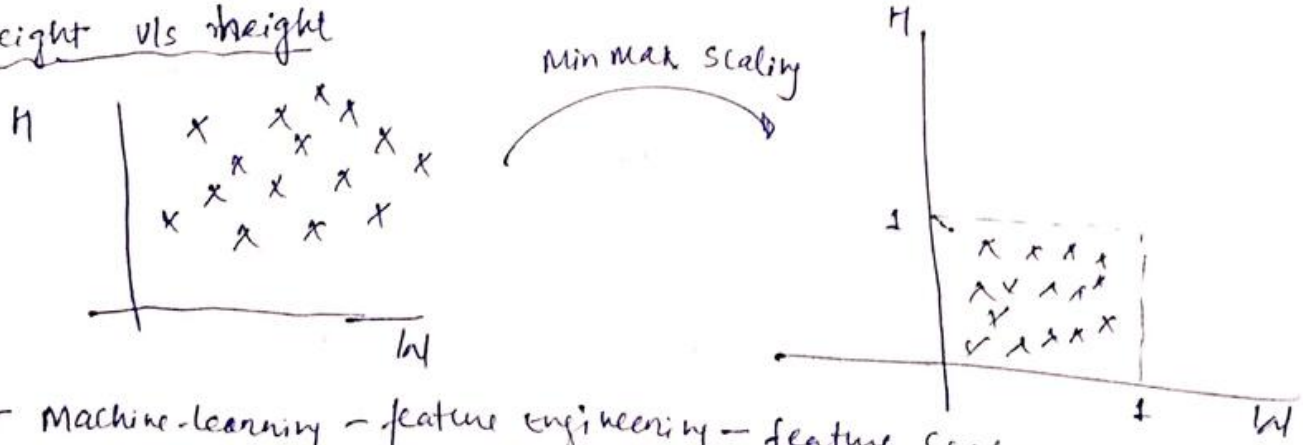
Normalization / Min-Max Scaling

$$X_i' = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

→ all pot
→ Smallest value 0
→ Range :- [0,1]

Geometric Intuition

Weight vls Height



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

— Mean Normalization

wt (X_i's)
73
86
54
31
⋮

Mean Normalization

$$X_i' = \frac{X_i - X_{mean}}{X_{max} - X_{min}}$$

→ Mean centering.
→ Range -1 to 1

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

This is used in those algos where we need centered data → But we often 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 :- $X_i' = \frac{X_i - X_{median}}{IQR}$

IQR = inter Quantile Range = $Q_3 - Q_1$ = 75% value - 25% value

class in Sci-kit Learn :- RobustScaler.

It is used when
- Data has outliers (it generally performs good with outliers)

Standardization v/s Normalization

1st Qn:- Is feature Scaling reqd? (check konse Algos mei reqd. hai & konse mei nhi)

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 :- (vvv imp)

Encoding Categorical Data

ML algos expect numbers & categorical Datas are in string/object format :. It's our duty to convert them to Numbers.

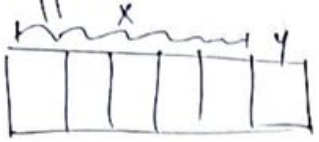
Techniques
- Ordinal Encoding (on ordinal data)
- One Hot Encoding (on nominal data)

There is also Label Encoding Similar to Ordinal Encoding

Types of Categ. data
- Nominal: There is no order (comparison) b/w the data. Ex (i) States — we cannot say UP is better than MP. (ii) Gender — we cannot say Male is better than Female
- Ordinal: There we have order. Ex. (i) Distinctions (Ranks in class) (ii) 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

Like: Email-spam classifier, placement hogg ya nahi etc. & X walope ordinal Encoding

Ordinal Encoding

Suppose we have an input col

Categorical Data ↓ It is ordinal

Education	Education.
High School	0
UnderGraduate	1
UnderGr..	1
Post Grad..	2
Post Grad..	2
High School..	0
Post Grad..	2
High School	0
UnderGrad..	1

HS → 0
UG → 1
PG → 2

PG > UG > HS

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

Jupyter Notebook.

One Hot Encoding :- (for Nominal

Consider a Nominal Categorical Data :- One-Hot Encoding

Color	Target
Yellow	0
Yellow	1
Blue	1
Yellow	1
Red	1
Yellow	0
Red	1
Red	0
Yellow	0
Blue	0

Color_Y	Color_B	Color_R	Target
1	0	0	0
1	0	0	1
0	1	0	1
1	0	0	1
0	0	1	1
1	0	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	1	0	0

in the color (Nominal Categorical col) we have Three Categories [Y, B, R] then in One hot Encoding we Make col for Each Category

		Y	B	R
Now first Data is Yellow	→	1	0	0
2nd Data is Yellow	→	1	0	0
3rd Data is Blue	→	0	1	0

and so on.

∴ [1,0,0] → Y
[0,1,0] → B
[0,0,1] → R

∴ Human String ko vector mei convert kar diya.

Qu:- What if we have 50 diff Categories?

Ans:- Then we make 50 cols.

Dummy Variable Trap (jo cols bante hai in one hot Encoding are called dummy variables)

What we do is after one-Hot Encoding we drop the 1st col (Generally, kisi bhi col ko kar sakte hai) Hey 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 Multicollineanity.

If we donot drop then Σ Each one = 1 → A mathematical rep" of the cols

we donot want this ∴ they create Problems for Linear ML models (Regression models)

∴ we drop col color-Y :- Now:-

Y →	[0,0]
B →	[1,0]
R →	[0,1]

Before

Y →	[1,0,0]
B →	[0,1,0]
R →	[0,0,1]

Say Car Brand → Nominal data

to say there are 40 Brands ∴ in one hot Encoding there will be 40 cols. (This is the way but this will make 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 bohot jyada & Kuch kam aarahi hai.

Column Transformer :-

Suppose we have input col's & in one col^n it is Ordinal, other is Nominal & other has missing values, then we cannot take the three col's as seperate parts Perform the operation & then Combine them back

Ex: 100 datapoints/rows

Age (20 values missing)	city	gender	review
∴ we'll have to use imputer to fill the missing values.		Nominal ∴ we'll use one-hot Encoding	Ordinal ∴ we'll use ordinal Encoding

↳ Saare col's pe alag alag transf. lagane hai hence again alag karke kiya then 3 diff numpy arrays banenge & unn teeno ko jodna padega → This is not efficient

∴ we use Column Transformer (Class in Scikit-Learn)

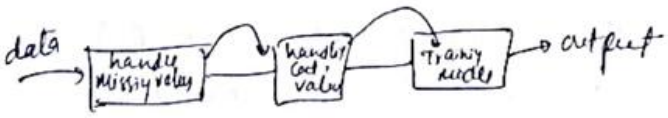
Jupyter Note book :-

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 → 1st we handle missing values then handle categorical data then say we train the model



Project

without Pipeline	with pipeline.

yaha jo series of steps hum train kr rhe time likhte hai wohi series of steps humne production time pe bhi likhne hote hai (which is not good)

Jupyter Notebook

Mathematical Transformations:

Feature Eng → Feature Transf. → Mathamatical Transformations.

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

- log Transf
- Reciprocal Tranf
- power Transf (Power & sqrt. Transform)
- BoxCox Transf (very famous)
- Yeo - Johnson Transf

→ we'll apply these transf. on columns & see ki model ka perf improve hota hai ya nahi of that col

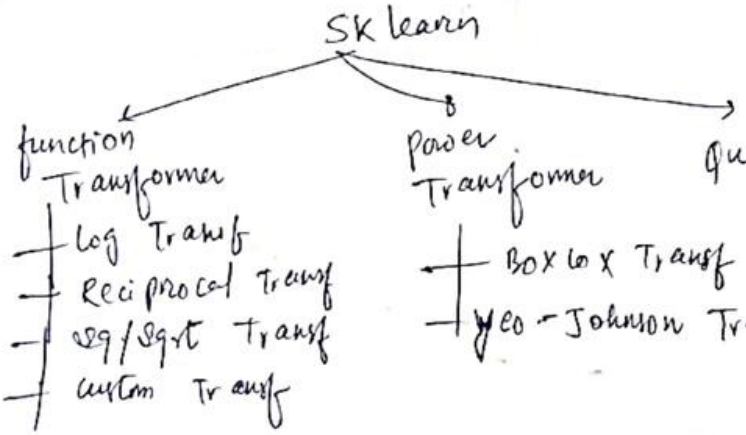
we can also define a custom transf like (x^2+2x) but after that we should get Normal dist.

After these transformation the pdf (distribution) becomes

Normal dist → vv imp 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.

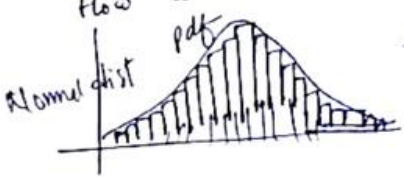
Algos like Decision Tree, Random Forest ko farak nahi padta ki data ka dist Normal hai ya nahi.

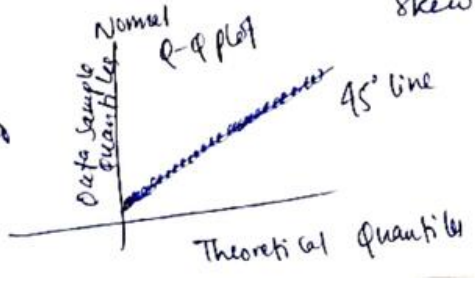


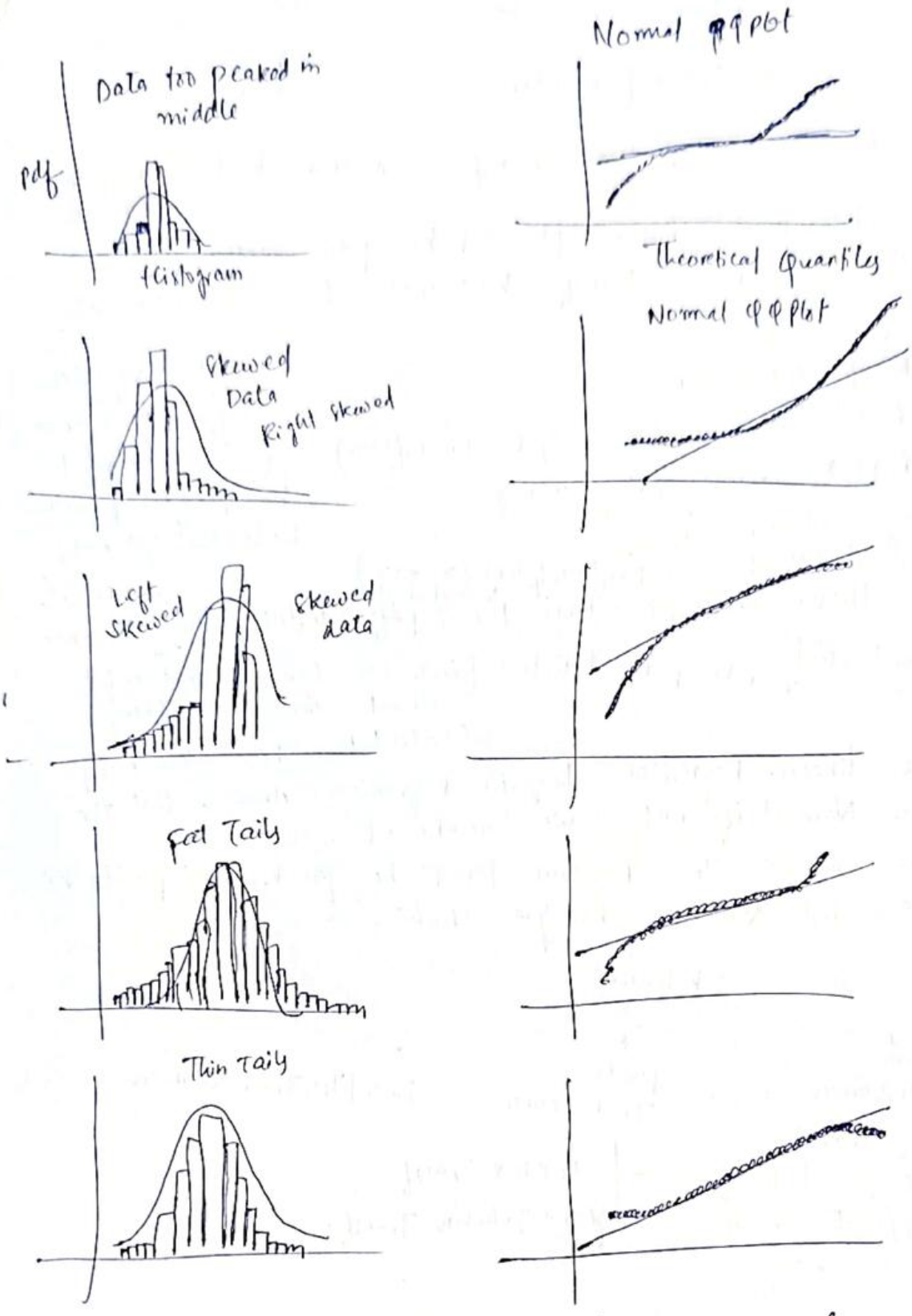
How to find if data is Normal?

- Use → Seaborn → sns.distplot()
- → pandas → pd.skew() → 0 ✓, +ve ya -ve aaye then skewed hai dist.
- → QQ plot

How to read







Hence. in QQ dist jitna aap line se repan ho utna aap ka data Gaussian Normally dist hoi & jithe dots line se QQ plot dekho bhag raha hai utna data kar Normally dist hai.

Log Transform:-

↳ To apply log transform on a col → we take each entry ka log (base 10 yaa e)

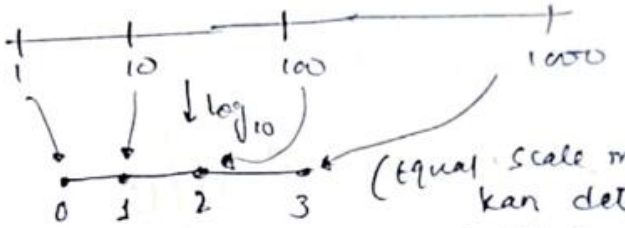
↳ isse apna wo column jo current state hai usse jyada Normally dist. ho jata hai

When to use:- • -ve values pe nahi lagega

• Right Skewed data  pe log transf. lagane pe wo usko center pe leke aajata hai.

(basic sagj)

What log does



(equal scale mei dist. kan dete hai) equidistant

Reciprocal Transform $(\frac{1}{x})$	Square Transform (x^2)	sqrt Transf. $\sqrt{x}$
small transfs big values big transf small values	↓ used for left skewed data then wo jo data hai wo pehle se jyada Normally dist hojata hai	

Jupyter NoteBook

Box-Cox Transformation :-

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

The Exponent here is a variable called Lambda (λ) that varies over the range -5 to 5 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 each variable (input col/feature)

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

$\log \Rightarrow \lambda = 0$; $\sqrt{x} \Rightarrow \lambda = 1/2$; $x^2 \Rightarrow \lambda = 2$

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

& This problem is solved by Yeo-Johnson Transf

Yeo-Johnson Transform:-

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

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

So if we have a col that is not Normal distributed & we are working with Algo (like Linear, logistic regression, knn 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)

- without Box-Cox or Yeo-Johnson
- with Box-Cox or Yeo-Johnson.

ENCoding Numerical -features :-

Ex: google playstore dataset

Apps # of downloads	Bins
23	100+
10102 3499	1000+
9927	1000+
11	1000000+
101	
34	10+
27598 62	
13 40516 3	

(its data type can convert to categorical data)

There are two famous Techniques for (Numerical → categorical)

- Discretization (Binning)
- Binarization

Discretization (Binning)

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

Why use Discretization :-

1./ To Handle outliers
2./ To improve the value spread (data ke dist ko thoda uniform karna)

Ex:- Age :-

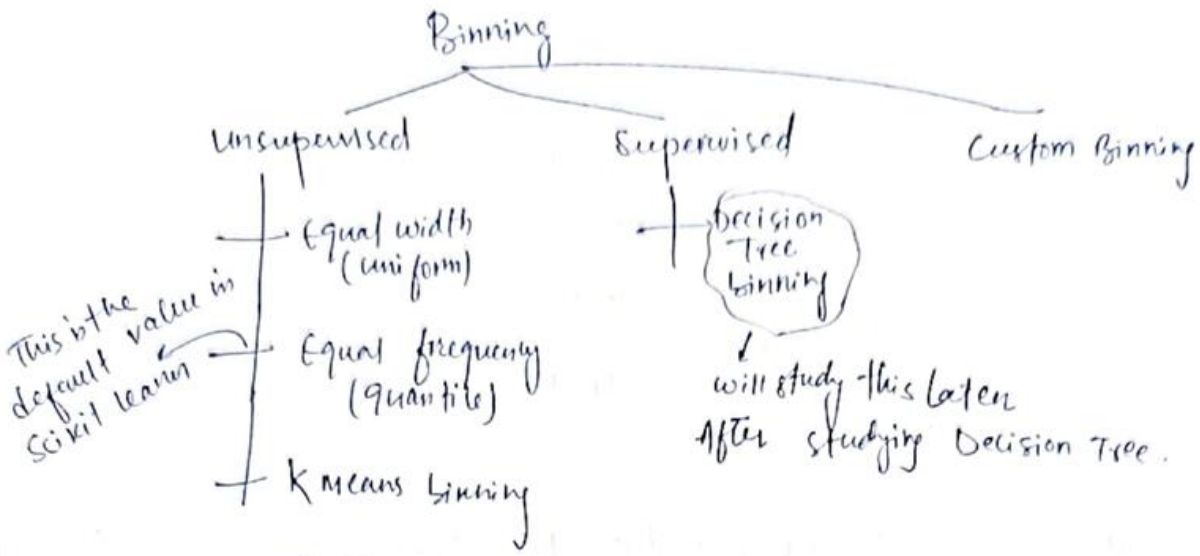
23, 1, 9, 41, 52, 37, 48, 51, ---- 82, ...

↓

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

5 4 6

Types of Discretization :-



Equal width (uniform binning) :-

Age :- 27, 32, 89, 56, ...

first we give #-g bins = 10 (say)

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

∴ the bins/intervals are :- 1st bin 0 – 10, 2nd bin 10 – 20, 3rd bin 20 – 30, ... 10th bin 90 – 100

and now put your values into these bins.

↓ and then we can plot like Histogram

— outliers handle ho jate hai ie. woh last ya first bin mei aajate hai.

— No change in spread of the data.

Equal frequency / quantile binning :-

say intervals / bins = 10

Each 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 bhi 10 log hai ↓ yaha bhi so on ...

but each bins has same #-g obsn / data.

Quantile binning performs better than uniform binning because.
- handles outliers better
- Values spread ko uniform banata hai.

K Means Binning.

Here we use a clustering Algorithm called kMeans.

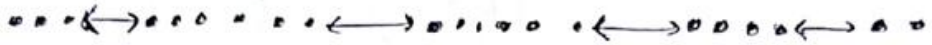
→ This clusters are formed by k means in any dimensions.

This is used jab hamara data clusters mei ho.

Kuch logo ka age | phir thore door ke baad kuch aur logo ka age | ... and so on.

Here :- intervals ko Centroid bolte hai

Age:



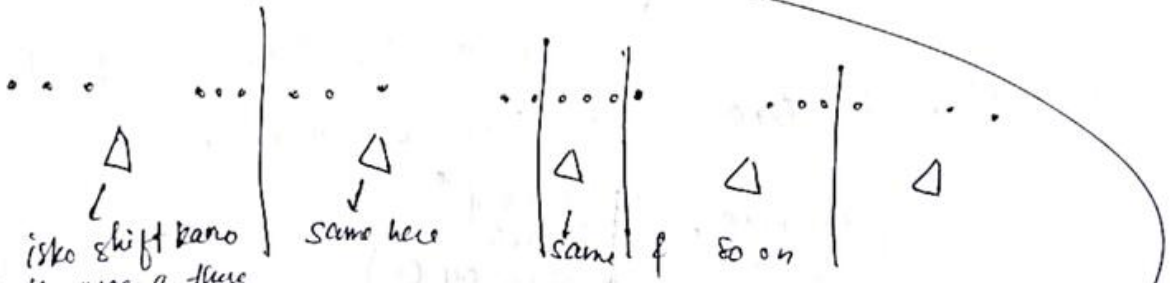
say #-q bins = 5 then 5 Arrows kahi bhi randomly place kardo

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

Now there are two techniques, → Har do centroids/arrows ka Bisection lelo

↓ use any one we get say

→ ~~Ek~~ har point ka arrow se distance nikalo wo point jis arrow ke sabse kareb 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 ~~arrow~~ centroids of that will finally give us the bins.

Hoco we use these three?

sklearn
↳ class → KBinsDiscretizer ( )

- bins = ?
- strategy
  - uniform
  - quantile
  - KMeans
- encoding (matlab bin no. of data; konsa data kis bin mei gya uska rep. kaise hoga.)
  - ordinal
  - ~~OneHotEncoding~~ One hot Encoding (if Nominal)

Jupyter NoteBook.

✳ custom / Domain Based Binning

Lo 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 < 6Lakhs → 0 → no need to tax (Taxable zone)
income ≥ 6Lakhs → 1 (has to tax)

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

- threshold Parameter (iske niche sab 0 and upar sab 1)
  - iska value by default '0' hota hai
- copy
  - say salary col pe Binarise lagi the hai:-
  - if copy = True then tk nayi col create karega & if copy = False then usi salary col^n 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: ① in Titanic Data set:-

Cabin.
B5
C23
D41
:

→ These are not categories Bcoz there will be many categories

B5 → we split this

cat	num
B	5
C	23
D	41

B → Categorical, 5 → Numerical.

Ex ② In a single col we have:

7
3
1
A
C
4
D

Again we split into two:-

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

Jupyter Notebook.

Handling DateTime based cols:

date:- 23 Aug 2012 → ye date weekend hai ya nahi etc. ya ye date weekdays hai.

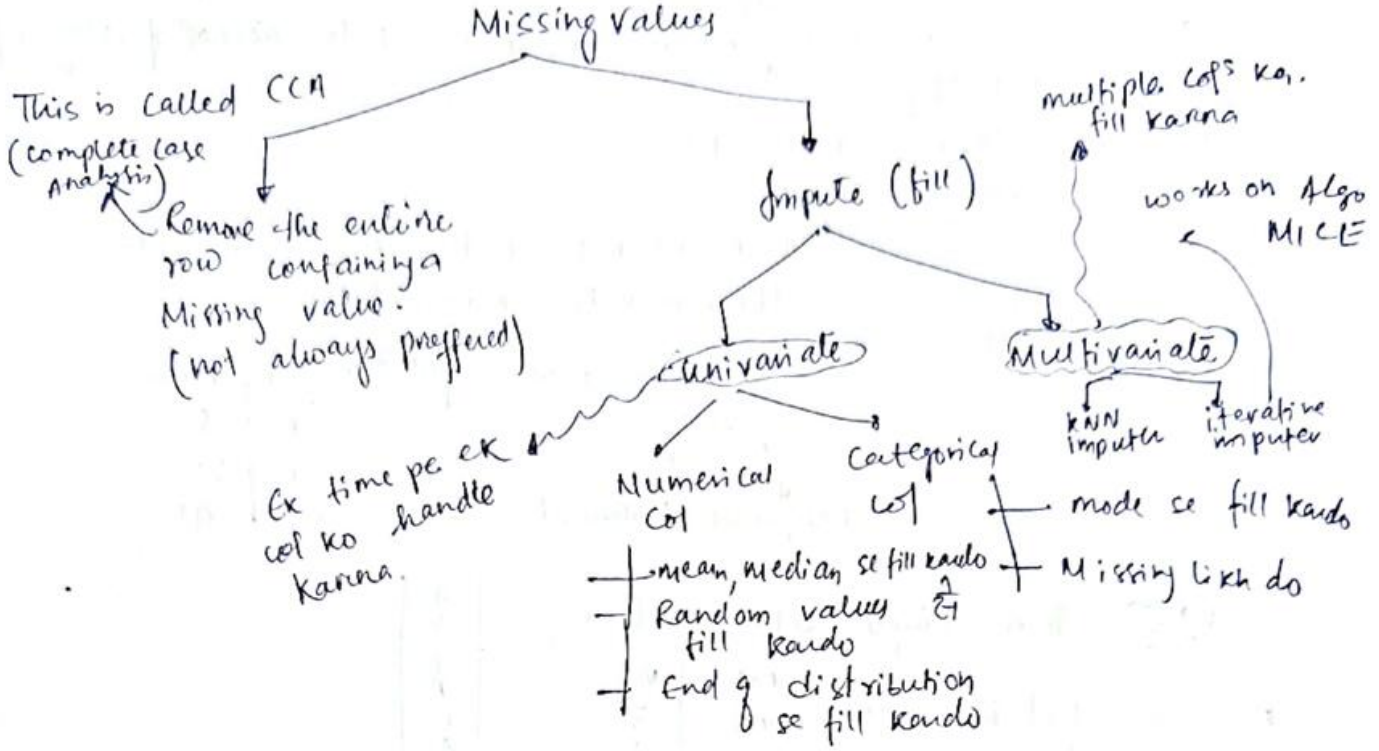
- 23 → Date (23)
- Aug → Month
- 2012 → year
- → days of week
- → quarter of year
- → which semester

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 sirf/ kuch 50 ya last values missing hai. (Tab hum CCA Apply nahi kar sakte)

Adv / Disadv. of CCA

Adv.

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

Disadv.

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

When to use CCA?

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

Jupyter Notebook.

Imputing (filling) missing values.

- Handling Missing Numerical Data

Imputing Numerical Data

- univariate Imputation
  say col2 mei kuch values missing hai & unko fill karne ke liye agar hum bas col2 ke baaki data ka 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 rahe 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 value fill kar dena.

} How to select automatically the best one using scikit Learn?
Ab:- Jupyter Notebook

Mean/median se imputation → very obvious.

- if dist is kind of Normal then fill with Mean
- if dist is skewed then fill with median.

Advantages:-

- Easy to use

Disadv.

All these should not happen.

- changes the dist shape
- outliers aajate hai.
- Jis col mei impute kiya usko rel^{n}ship with other cols mei changes aajata hai. This is called changes in covariance/correlation.

When to use:-

(i) Data is MCAR (missing Completely at Random)

(ii) Missing data < 5%.

Jupyter Notebook.

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

↳ Mostly this technique is used for categorical variables

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

↳ 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 one's idea :- we are creating a difference b/w the existing data and the missing values.

Adv:-
- Easy to apply

Disad.
- pdf becomes distorted
- variance, covariance changes.

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

— End of Distribution Imputation
- It is an extension of Arbitrary value imputation.
- Arbitrary value (valet value) dhundna thoda difficult hota hai
- Yaha we choose a value from the end of dist.

for Normal dist the arbitrary value = Mean + 3σ or Mean − 3σ (σ = std)

if Dist is skewed, we use IQR Proximity Rule to Calculate the arbitrary value:-

arbitrary value = $Q_1 - 1.5\,IQR$ (IQR = $Q_3 - Q_1$)

or arbitrary = $Q_3 + 1.5\,IQR$

adv:-
- Easy to apply (same method as above)

Disadv
- pdf & variance, covariance change hojaenge.

When to use?
→ When Data is not missing at Random.

Random Sample Imputation & How to select the best Technique for imputation
- It can be used on both Numerical & Categorical data.

Then the missing values are filled by random Numbers & the random numbers belong to the Non missing values in the col. ie. col ke data mei se hi randomly ex numbers select karke fill karenge.

advantage:-
→ It is easy to apply. almost
→ Data dist & variance remains intact doesnot get distorted.

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 works good on Decision Tree based Algos.
- However the covariance of the col with other cols gets disturbed.
- This technique memory heavy for deployment, as we need to store the original training set to extract values from & replace the NaN in the coming observations.

  i.e. X_train age mei se ek random value dalegi koi bhi usse incoming data mei agar age missing ho toh fill kardenge.

Jupyter Notebook.

Missing Indicator

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

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

→ isse ML model missing & non missing data mei diff create karna sikh jatahai and hence uska perf improve ho jata hai.

How can we automatically select the best parameter values for any imputation Technique?

Here we use a technique called Grid Search CV.
↳ yaha sciKit-Learn saare combinations try karta hai and best results 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) and missing data < 5% → Replace with mode of the col data. jo sabse jyada bar appear hua hai.
And one more imp thing is the mode should appear in the data much more than the remaining categories.

- adv:- Easy to use
- disad:- 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.

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

Jupyter Notebook.

MULTIVARIATE IMPUTATION TECHNIQUE

KNN imputer Iterative Imputer

KNN Imputer

- It works on algo called KNN (apne neighbours ki sunaho)

Isme agar kisi row mei koi value missing hai toh waha us row ki value fill hogi jo sabse jyada similar hai row to be filled se.

- Similarity is calculated using Euclidean distance.

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	—

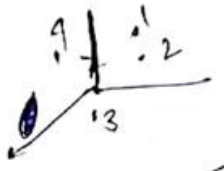
NaN

dist(x,y) = sqrt(weight * sq. of distance from present coord.) ← wo point jo dono mei present hai in x & y

where weight = Total # of coord / # of present coord.

This can be represented in a 4-D coord system

(say)



this we do if we take k = 1

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

Similarly again 4 fill kare then dist(4, y) → jo sabse close uske feature 2 ki 4th row mei fill karenge.

In KNN, k → # of neighbours we want to select with

Say k=1 then sabse closest neighbour ka feature 1 ka value ko replace karenge row 2 mei

if k=2 Then two nearest neighbours from point/row 2 ka feature 1 ka mean ko fill karenge in row 2 and so on.

Calculating NaN distance.

	F1	F2	F3	F4	SI NO
we want to fill this →	33	—	67	21	1
	45	68	12		2

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

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

Total pairs = 3, pairs possible = 2 → weight = 2/3

Row 3	51	71	18

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

weight = # of total pairs / # of possible pairs = 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 of row 4 is 40 will be filled at row 2 feature 1

if K=2. then 2 nearest neighbours are row 3 & row 4

∴ $\frac{\text{feature 1 of 3} + \text{feature 1 of 4}}{2} = \frac{23+40}{2} = 31.5$ will be filled at feature 1 of row 2 & so on.

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

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

Jupyter Note Book.

There is a parameter called weights = uniform or distance in KNN imputer class (default)

uniform is the normal what we calculated above to fill.

if weights = distance → to fill this → K=2. say nearest neighbours are 1 & 4 (dist = 10, dist = 50)

then value to be filled at row 3 = $\frac{\frac{1}{10} \times 70 + \frac{1}{50} \times 50}{2} = 4$

isse jo jyada weightage jyada kare hai. (close wali ka)

Multivariate imputation by chained equations for Missing value | MICE Algo | Iterative Imputer

MICE stands for Multivariate imputation by chained eq^{n}s. (Learner)

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

for this there is not much rel^n b/w the col's so other cols cannot be used to fill the missing data in a col.

Assumptions for using MICE

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

MICE Algo ↑ ML Algo

Adv: • It is quite accurate

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

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

1) Actual Dataset (Startup Dataset)

R&D spend	Administration	Marketing spend	Profit (Target col.)
8	15	30	11
4	5	20	9
15	10	41	19
12	15	26	13
2	16	3	7

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	36
NaN	5	20
15	10	41
12	NaN	26
2	15	NaN

Now we need to predict the Missing values using MICE Algo.

Predicting the Missing values using MICE Algo

step 1. Fill all the NaN values with the Mean of resp cols.

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

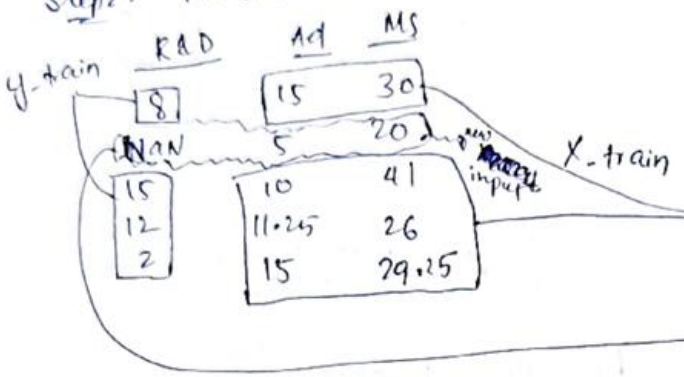
→ Iteration 0

Mean of R&D

Mean of Ad

→ mean of MS

step 2. Remove all cols missing values.



step 3. predict the Missing values of cols using other cols

Algo → using any ML Algo (Regression, Decision tree)

Here: over input training data

is: with R&D as target col

& test data → we provide

Ad	MS
5	20

to the input the Algo after training & it predicts the output. say Here the prediction is 23.14.

we have

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 cols 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)

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

with output Ad

Ad
15
5
10
15

& then provide input

R&D	MS
12	26

& predict the NaN output of Ad.

step 5. predict the Missing values of cols using other cols say we got 11.06

step 6. Continue for col 3.

After that say we have:-

RED	Ad	MS
8	15	30
23.14	5	20
15	10	41
12	11.06	26
2	15	31.56

→ Iteration 1

Diff b/w iteration 1 & iteration 0

Iteration 1 − iteration 0 =

RED	Ad	MS
0	0	0
13.89	0	0
0	0	0
0	−0.19	0
0	0	2.31

→ the diff should be 0 or almost 0.

So again we repeat the process:-

By taking iteration 1 as Base Table:-

we Remove all NaN from Col1 & predict it using the reg & so on

& get iteration 2 table & Calculate iteration 2 − iteration 1 if all 0 or about 0 then done. otherwise continue.

Say we got iteration 2

RED	Ad	MS
8	15	30
23.78	5	20
15	10	41
12	11.22	26
2	15	31.56

12 − 11 → difference

RED	Ad	MS
0	0	0
0.64	0	0
0	0	0
0	0.16	0
0	0	0

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

Outlier (Sharmji ka Lelo)

↳ Ek aisa data point jo baaki data se alag behave karta hai & aise data points ML Algos ko disturb kar sakte hai.

Suppose Ek class hai.. we are finding the ~~mean~~ of that Salary

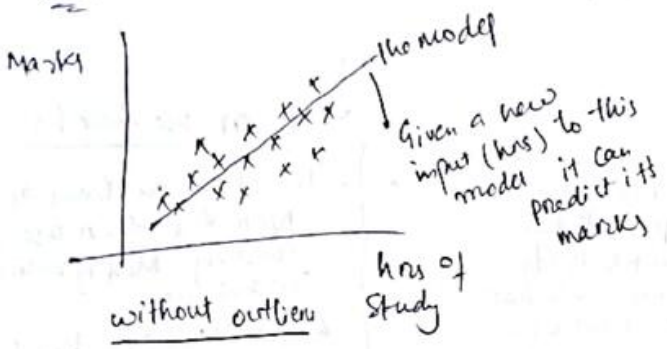
Class → say ₹75000 per month

Ab usi class mei Ek outlier aaya (Bill Gates) then

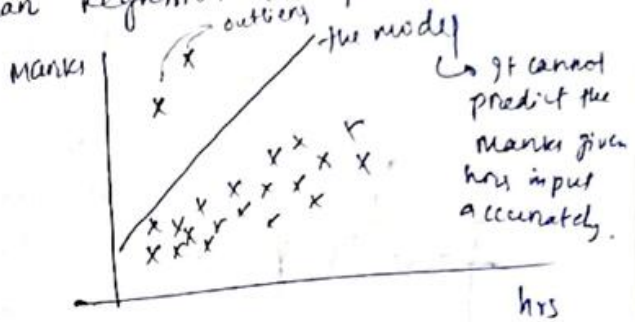
Now the Mean Salary will be 50 crores per month

Now this new Mean is can not a true representation of the Salary of that class.

Ex: Consider we are training Linear Regression Model



without outlier



with outliers

* 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 col of each student & this may justify the outliers (having high IQ)

Effect of outliers on ML Algos

if data has outliers then the performance of following ML algos will decrease →
- Linear Reg
- Logistic Reg
- Adaboost
- Even in Deep Learning Algos.

in these algos we calculate weights : outliers hampers weight-based ML Algos.

Tree Based Algos

Decision Tree
Random Forest
Gradient Boosting
XG Boosting

} inpe outliers ka utna farak nahi padta.

How to treat outliers?

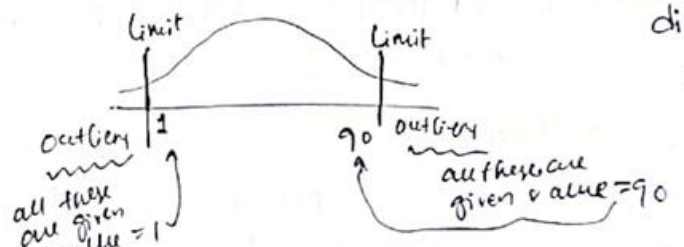
Trimming (Removing)

adv:- Very fast

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

Capping

outlier hamesha ends pe milte hai ie: ya toh jyada marks wala hai ya bahot kam marks wala hai so we set Limit



Kam use hote hai

- Treating outliers as NaN & then use Handling Missing Values Techniques
- Discretization karde (say Age) Numerical data ko discretize karna :- 0-10, 10-20, ... 90-100

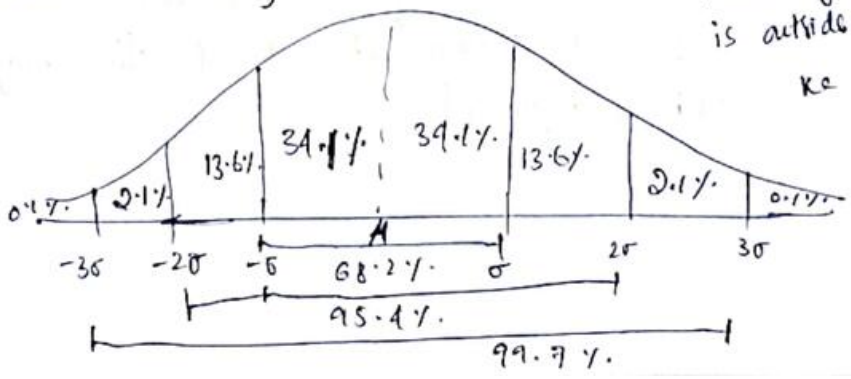
then 90 ke upar jo outlier the toh wo ab 90 ke saath merge ho gya & ab wo outlier nahi rha.

How to detect outliers.

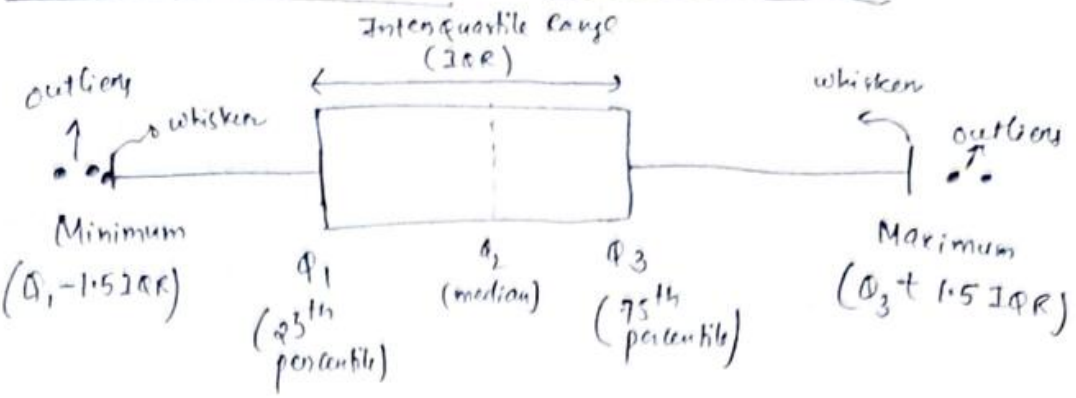
Three imp methods.

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

Hence agar apna observation is outside $(\mu - 3\sigma, \mu + 3\sigma)$ ke bahar hai then we can treat it as outlier.

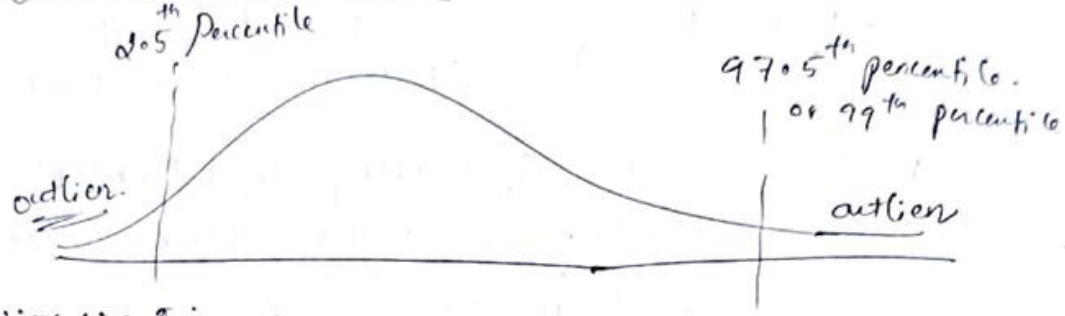


2./ Again Data is skewed (Left ya Right skewed)
Then we use Box Plot (IQR Based filtering)



3./ other Distribution (percentile Based filtering)

percentile Based approach



4./ Winsorization (Later).

* Z-score technique for outlier Detection & Removal

↳ The col with outlier must be Normally Dist. (or almost Normally dist).

↳ outside $(\mu - 3\sigma, \mu + 3\sigma)$ the value is an outlier

Age
27
32
⋮

$\xrightarrow{\text{Z-score}}$ $x_i' = \frac{x_i - \mu}{\sigma}$ Say $x_i' = 3.5$

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

$\therefore x_i$ is an outlier.

$\therefore$ if $x_i' \to (-3, 3)$ the data (x_i) is ok

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

correspondingly.

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 $x_i = -3\sigma + \mu$

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

then $x_i = 3\sigma + \mu$.

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 it's distribution then we decide a amount like upon and needhe values ko we declare Outlier! after that we do

- Trimming
- Capping (In this Technique Capping is called Winsorization).

Jupyter notebook.

Feature construction

- Given features se Manually new features create karte hai.
- feature splitting.

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

Ex:- In Titanic → sibsp, parch → family (categorical feature)
- 0 → Alone (0)
- 1 → Small family (1-4)
- 2 → Large family (5-9)

This makes the accuracy of model better.

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

Ex:- in Titanic dataset: 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 building that tells ki feature hatane pe kuch fayda hota hai ki nahi ML algo performance pe).

In Dataset, we have cols/features (Dimensions).
When we use 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 → n-dimensional (n → 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 → Analogy :- wallet lost on road (1d like - 1D)
playground (2D)
Building (3D)

then finding wallet on 1D is easier. (since wallet to fewer cases me 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.

Dimensionality Reduction

Feature Selection

Say we have features
$\{f_1, f_2, \cdots f_n\}$
we take a subset :-
$\{f_1, f_3, f_7\}$ &
run the model on it & see it's performance & do it for many (prob all) subsets to see if we can remove some features & improve or maintain the accuracy.

- Forward Selection
- Backward Elimination

Feature Extraction. $\{f_1, \cdots f_n\} \xrightarrow{ET}$

- PCA (Pricipal component analysis)
- LDA (Linear Discriminant Analysis)
- TSNE.

completely new set of cols created kaise har $\{e_1, e_2, \cdots e_k\}$ these cols are actually linear comb of the $\{f_1, \cdots f_n\}$ and use these new features.

PCA (Principal Component Analysis)

↳ It is an unsupervised ML problem (we have input col only & no output)
↳ It is an useful, old, reliable technique & it's 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.
↳ ie. PCA best possible lower dimension mei lekar aata hai while keeping the essence/behaviour of the data intact

Benifits 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 10 Dim, PCA reduces it to 30 or 20 & 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

output

→ Agar yaha dono input col mei se ek ko rakhna hai then we will keep the one jispe output jyada depend kar raha hai ie No. of rooms and remove the other one.

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

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

plot: Scatter plot

→ 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

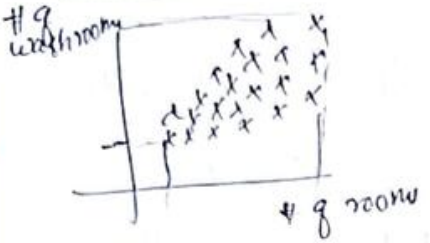
Now suppose we have changes in Data

#q rooms	#q washrooms	price
:	:	:

→ the DF 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 wala trick).

in this case if we Scatter Plot



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

In such case → we take help of Feature Extraction

#q rooms	#q washrooms	price
:	:	:

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

Collect your data in a different way

size (area)	price
:	:

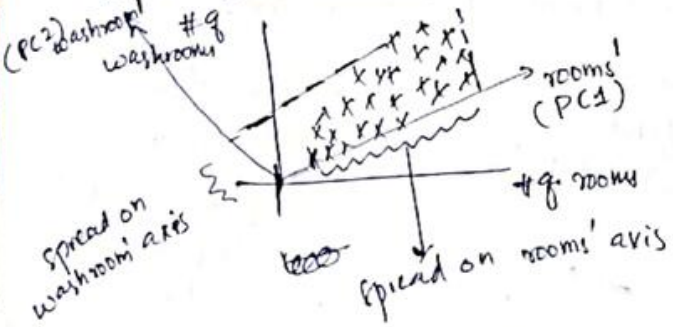
since more the #q rooms + #q washrooms more is area & Now we have only one col.

↓ This is exactly what PCA does

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

→ upar to Broker ne bata diya ∵ he had Domain knowledge:-

As a Datascientist How do we do it:-



→ yaha feature selection technique fail kar gya tha since variance dono cols pe same aaraha tha.

↓ So what will PCA do:-

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

rooms' → PC1 → Principal Component1, washrooms' → PC2 → Principal Component2.

& Now rooms' (PC1) ko rakhenge & washrooms' (PC2) ko hata denge.

↳ ∵ variance of data on PC1 is more.

Now we transform the Data according to the PC1

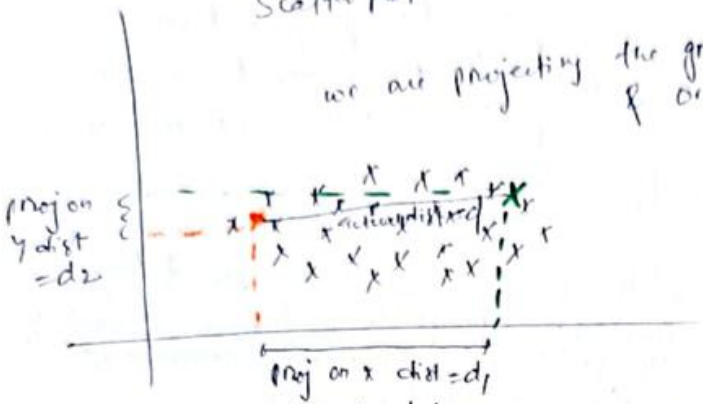
Note:

No. of principal components = ≤ no. of original cols / features / Dimensions

* Why is Variance important for PCA

Scatter plot

we are projecting the green & orange pts.



Now X axis data has more spread hence on PCA we chose the X axis data

Why?

∴ when we choose x-axis data then the 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 lete hai toh datapts ke aapas mei distance/behaviour nai bdlna chahiye, isliye hum variance ko maximize karte hai.

What Variance Represents

↓

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

if we have Dataset:

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

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

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

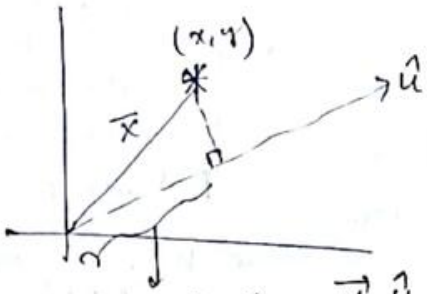
Problem Formulation in PCA

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

→ Humme ek single aisa axis find karna hai jiske upar hum apne data ko project kar paye & humme utne hi accche results mile jitne 2D mei mil rahe hai.

This direction we need is the unit vector we need. ($\hat{u}$)

Consider only one point:..



→ Silly 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.

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

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

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

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

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

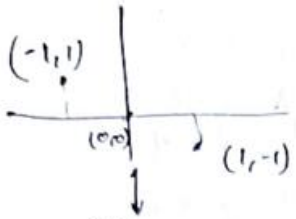
Now this becomes an optimization problem.

This is called mathematical objective fn.

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

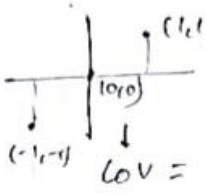
Covariance & Covariance Matrix

- Variance is a metric of a single axis, a single col/feature.
- Covariance is a metric of two axes, two cols/features. Lit gives the rel^n ship b/w x & y axes.



$$Cov = \frac{(-1\times1)+(0\times0)+(1\times-1)}{3} = -2/3$$

↳ -ve means as x ↑ses y ↓es.



→ Done this data sets ka variance same aayga.

→ So acc.to variance they are same but they are not.

$$Cov = \frac{(-1\times-1)+(0\times0)+(1\times1)}{3} = 2/3$$

→ +ve means as x ↑ses y ↑ses

↓ Cov is similar to Corr; except Corr is restricted to [-1, 1] but Cov has no restrictions.

Covariance Matrix:-

say we have:- $X_1 \mid Y_2$ ⋮ ⋮ then Cov Matrix → will be 2×2 matrix

$$\begin{matrix} X_1 \\ Y_2 \end{matrix}\begin{pmatrix} Cov(X_1,X_1) & Cov(X_1,Y_2) \\ Cov(X_2,X_1) & Cov(X_2,Y_2) \end{pmatrix} = \begin{pmatrix} Var(X_1) & Cov(X_1,Y_2) \\ Cov(X_2,Y_2) & Var(Y_2) \end{pmatrix}$$

X_1 Y_2

Sly. if $X_1 \mid Y_2 \mid Y_3$ ⋮ ⋮ ⋮ then Cov Matrix.

$$\begin{pmatrix} Var(X_1) & Cov(X_1,X_2) & Cov(X_1,Y_3) \\ Cov(X_1,Y_2) & Var(Y_2) & Cov(Y_2,Y_3) \\ Cov(X_1,Y_3) & Cov(X_2,Y_3) & Var(Y_3) \end{pmatrix}$$

Square Symmetric Matrix

$\because Cov(a,a) = Var(a)$ & $Cov(a,b) = Var(b,a)$

→ This Matrix is important since it has all the variances $Var(X_1)$, $Var(Y_2)$, $Var(Y_3)$ that tells the spread of data on each axis.

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

Eigen Decomposition of Covariance Matrix :-

Basics: Eigen vectors & Eigen values :- (Matrices as Linear Transformation)

→ for this Read Article 100 days of ML
lecture 47 PCA part 2 Live in Descrip. vision dummy.com.

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

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 equal to the corresponding Eigen value.

The second largest Eigenvector is always orthogonal to the largest Eigenvector & points into the direction of the 2nd largest spread of Data.

Step by step soln :-

input Dataset :-

f1 | f2 | f3

step 1 :- Mean centering of Data
(not a mandatory step)
but PCA improves if done mostly.

step 2 :- find cov Matrix. → (np.cov using numpy)

	f1	f2	f3
f1	V_{f1}	C_{f1f2}	C_{f1f3}
f2	C_{f1f2}	V_{f2}	C_{f2f3}
f3	C_{f1f3}	C_{f2f3}	V_{f3}

step 3 :- find Eigen values & Eigen vectors of the covariance matrix. → (np.linalg)

& jis Eigenvector ka Eigen Value sabse jyada hai vaahi PC1 banta hai, then PC2 then PC3 (all these are orthogonal to each other)

Now it's our choice if we want PC1, PC2 → 2D or only PC1 → 1D.

Now after getting the principal components then how do we transform the points. → f1 | f2 | f3 → pc1 | pc2

i.e. say originally in 3D ⟶ 2D if we choose pc1, pc2

" ⟶ 1D if we choose pc1.

f1 | f2 | f3 ⟶ pc1

3D → 1D. Take the projections of the points on pc1 i.e. dot product of pts with pc1.

Say there are 1000 pts in 3-D

the shape of pts is (1000, 3) matrix

& the pc1 is (1, 3) matrix

∴ dot product gives (1000, 1) matrix which will be also pc1

3D-2D. → pc1 & pc2 (plane)

projection on the pc1 pc2 plane.

again shape of datapoints is (1000, 3) & shape of pc1, pc2 plane is (2, 3) → take dot product i.e. $(1000,3) \cdot (2,3)^T$ (this is U^T)

$= (1000, 2)$ → pc1 | pc2

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 → 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 ∴ 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 → 784 cols. (input)

then the Eigen values of the Covariance matrix are $\lambda_1, \lambda_2, \ldots \lambda_{784}$

λ_1: this denotes the spread/variance of data on PC1.

λ_{784}: This explains the spread of data on PC2.

Convertig them to %

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

this is how much % of data variance is explained by λ_1

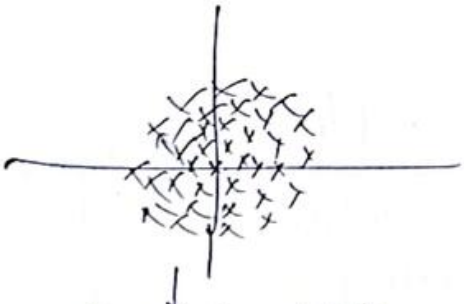
and so on & humme tota 90% variance explain karwana hota hai.

Hence humme utne values (λ_i's) lene hai jitne mei jaate jaate hum 90% pohuch jaye.

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

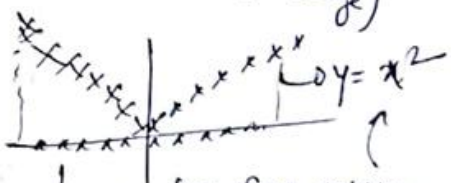
Jupyter Notebook (kaggle)

When PCA Doesnot work? (ie data is aisa hai ki wo PCA-ko kaam hi nahi karne dega)



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

Dono ka Projection le toh wo ekdusre pe overlap karjaiye & PCA do points ko diff. hi nahi kar payega.



↓ data hai some pattern iska projection in lower Dim the patter 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
- Multiple Linear Regression
- Polynomial Linear Regression
- Regularisation

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	State	IQ	Package
⋮	⋮	⋮	⋮	⋮

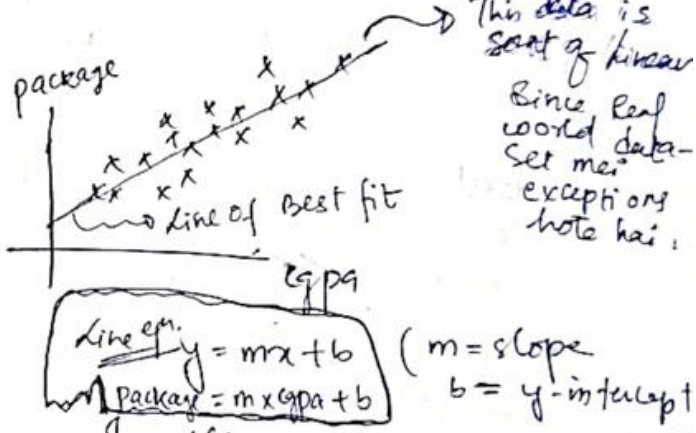
(CGPA, Gender, State, IQ → input; Package → output)

Polynomial Linear Regression → this helps when data is not Linear

- 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. $y = mx + b$

Package $= m \times CGPA + b$ (m = slope, b = y-intercept)

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

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 finding m and b.

Linear Regression Model.

Jupyter Notebook

finding the optimum no. of principal components :-

Say we have MNIST Dataset → 784 cols. (input)

then the eigen values of the covariance matrix are $\lambda_1, \lambda_2, \ldots \lambda_{784}$

λ_1: this denotes the spread/variance of data on PC1.

λ_{784}: This explains the spread of data on PC2.

Comentify them to %

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

this is how much % of variance of data is explained by λ_1

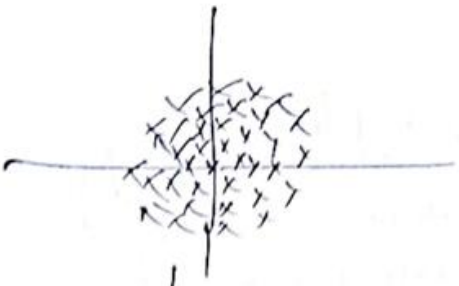
and so on & humme tota 90% variance explain karwana hota hai

hence humme utne values (λ_i's) lene hai jitne mei jaate jaate hum 90% pohuch jaye.

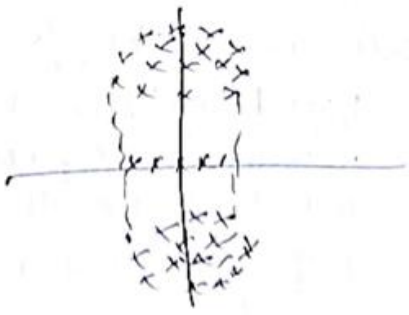
Say $(\lambda_1\% + \ldots + \lambda_{15}\%) = 90\%$ then we take the PC1 ... PC15 → ie 15 principal components are the Optimum components for good/best accuracy.

- Jupyter Notebook (kaggle)

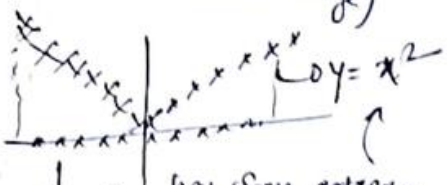
When PCA Doesnot work? (ie data is aisa hai ki wo PCA ko kaam hi nahi karne dega)



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 ko projection le toh wo ekdusre pe overlap karjaenge & PCA do points ko diff. hi nahi kar paayega.



Data has some pattern, iska projection ki 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
- Multiple Linear Regression
- Polynomial Linear Regression
- Regularisation

Simple Linear Regression → only one input col & one output col.

Ex:- Consider the data :-

cgpa (input)	package (output)
6.66	3.01
⋮	⋮

Multiple Linear Regression → more than 1 input col

Ex:-

cgpa (input)	Gender (input)	state (input)	IQ (input)	package (output)
⋮	⋮	⋮	⋮	⋮

Polynomial Linear Regression → This helps when data is not linear

• Simple Linear Regression

cgpa (input)	package (output)
7.1	3.5
4.7	1.2
8.9	4.2
8.1	3.9
⋮	⋮

plot this data

This data is sort of linear since real world dataset me exceptions hote hai.

Line eqn. $y = mx + b$

$package = m \times cgpa + b$

(m = slope, b = y-intercept)

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

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

finding m and b.

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

Jupyter Notebook

Mathematical model of Linear Regression

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

Closed form soln

↓ in this case we have formulas to calculate values/soln's

This technique is called OLS (ordinary least squares)

↳ Scikit Learn uses this internally when we use the class LinearRegression()

↓ Here the formula is:-

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

y = package ($\bar{y}$ → mean)

x = cgpa ($\bar{x}$ → mean)

m = slope

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

Non-closed form soln.

↓ Here we use approximation techniques to reach to the soln's.

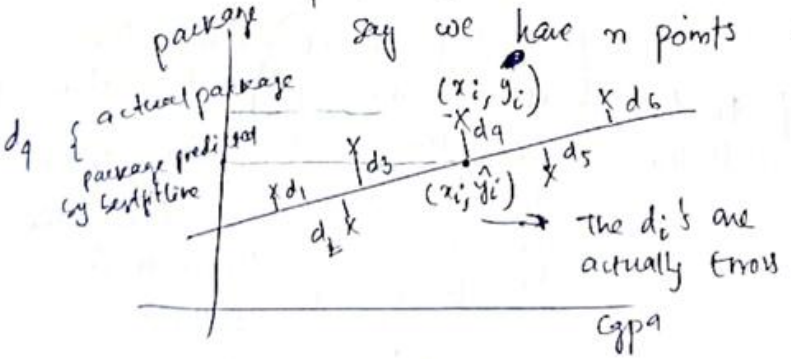
↳ Here we use Calculus

↳ Using Gradient Descent in ML.

↳ In higher dim input data Gradient Descent works better

↳ in ML :- there is a class in scikit Learn called SGDRegressor() → in it Gradient Descent is implemented.

* Best fit line woo hota hai jo saare points ke pass se hokar gujarne ka try karega.; ye line Minimize karta hai.



each point ki distance

say we have n points

Total Error

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

or $E^2 = d_1^2 + d_2^2 + \cdots + d_n^2$

need to minimize this

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

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

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

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

$\therefore E = \sum_{i=1}^{n} (y_i - \hat{y}_i)$ (→ average Error ke liye 1/n karlo)

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

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

note :- $\hat{y}_i$ → predicted value

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

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

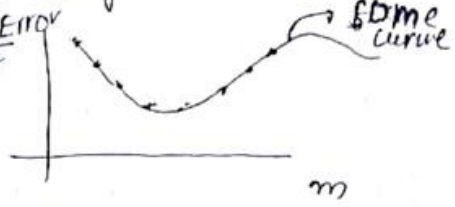
$$\Rightarrow E(m,b) = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

; x_i, y_i are the input & output data resp. & they are fixed.

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

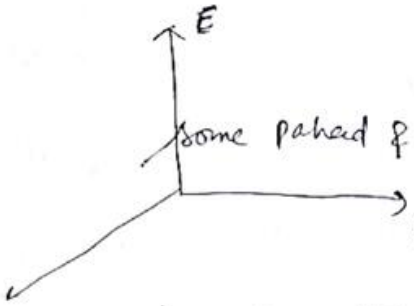
Note: m fixed ; b → change → Translation parallel.
m change, b → fixed → Rotation about b.
m change, b change → Translation.

Error v/s m (fixing b)



Error v/s b (fixing m)

∴ in 3-D.



some pahad & khai diagram see google.

(multivariable calculus)

and using Maxima & minima we can find the pt. of minima.

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

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

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

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

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

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

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

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

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

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

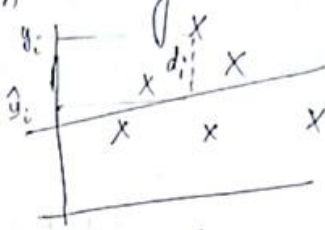
Jupyter Notebook.

Regression Metrics

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

(1) MAE :-

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



Regression Line (Line of Best fit)

We know what d_i is → d_i → i^{th} point Regression line ke respect mei kitni galti kar raha hai. (we saw: $d_i = y_i - \hat{y}_i$)

y_i → Given in the data, $\hat{y}_i$ → predicted.

total absolute Error

then = $|y_1 - \hat{y}_1| + \cdots + |y_n - \hat{y}_n|$

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

→ 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)

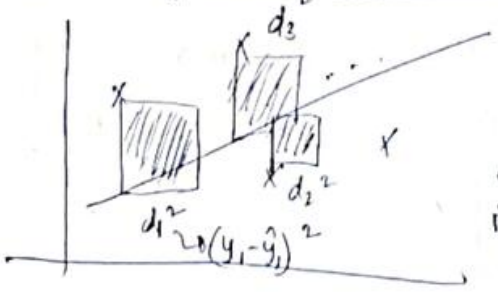
Disadv :- (i) MAE is not diffble at '0' (∵ mod)

(2) MSE (Mean Squared Error)

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

→ 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 is a bit difficult to interpret like: MSE = 11.25 $(LPA)^2$ → Loss = $\sqrt{11.25}$ LPA

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

Hence: agar Data mei jyada outliers hai then go with MAE if kam hai - then go with MSE.

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

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

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

↳ Not Robust to outliers, just like MSE.

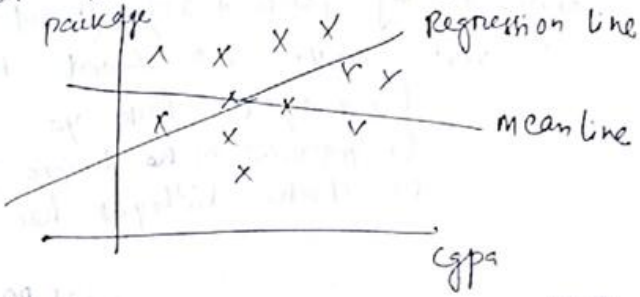
→ As a Beginner → techno metrics viral lena.

(4) R2_Score → (Coeff of determination or Goodness of fit)

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 (not a good guess) 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: $R2\_Score = 1 - \frac{SSR}{SSM}$

SSR = sum of squared d_i's wrt Regression line

SSM = " " " " wrt Mean line.

$$\therefore R2\_Score = 1 - \frac{\left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2\right]_{\text{Regression Line}}}{\left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2\right]_{\text{mean line}}}$$

Interpretation of R2_Score.

- if R2_score = 0 ⇒ $SS_R = SS_M$ → not good
- if R2_score = 1 → $SS_R = 0$ → Best/perfect case scenario i.e. Regression line passes through all the Data points.

Hence: jitna Reg. line perfection ke taraf move karta hai utna R2_Score 1 ke taraf move karta hai & jitna reg line kharab hota hai utna R2_score 0 ke taraf move karta hai.

* what if R2-score is -ve ⇒ $SS_R > SS_M$ ⇒ Regression line, hamare mean wale line (worst case) se bhi jyada galtiya kar raha hai.
This should not happen.

Another interpretation of R2-score:

say we have cgpa | package

& R2-score = 0.80

this means cgpa explain the 80% of variance in the package (lpa) (y-axis) (output)
(x-axis) (input)

i.e. 80% jo variance hai in package (i.e. kisi ka 3.1 lpa hai & kisi ka 4.2 lpa hai etc).
that is due to cgpa

& rest 20% variance in package → we donot know.

eg: cgpa | IQ | package → R2-score = 0.80

Then again 80% variance of package is explained by (cgpa + IQ) (input cols)
& rest 20% we donot know.
↳ may be luck ya
↳ interview accha hogya ya
↳ chacha vidhayak hai etc.

(5) Adjusted R2-score:-

say we are using R2-score with dataset cgpa | package & the R2-score = 0.80 & now if we add further input col say IQ → cgpa | IQ | package then now the R2-score will increase say 0.90. → which is OK
package may depend upon IQ.
But say if we add an irrelevant col ie temperature (this should not have any impact upon package) but even then sometimes R2-score increased indicating temp is affecting package.
Hence we use adjusted R2-score:-

formula :- $\text{Adj. } R2 = 1 - \left[\frac{(1-R2)(n-1)}{n-1-k}\right]$

n → # of rows | datapoints in X_test
k → total no. of indep. cols (input)

(only cgpa → k=1
cgpa + IQ → k=2
cgpa + IQ + temp → k=3)

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

- Since faltu ke col add karne pe k ↑ses (like temp) Deno decs
  ∴ $\frac{(1-R2)(n-1)}{n-1-k}$ ↑ses
  ∴ R2 adj ↓ses.

- Adding Relevant col say IQ

k ↑ deno ↓, again in 1-R2 in num ↓ses rapidly ∵ R2 ↑ses hence Num ↓ses. & in such case the dec in Num is greater than the ↓se in Deno, Hence R2-adj ↑se hoga.

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

Jupyter Notebook

MULTIPLE LINEAR REGRESSION

Real world Data are of multiple Linear Regression. Ex:

↳ more than 1 input col Ex:

cgpa	Gender	IQ	package
:	:	:	:

input / independent

x_1	x_2	x_3	y
:	:	:	:

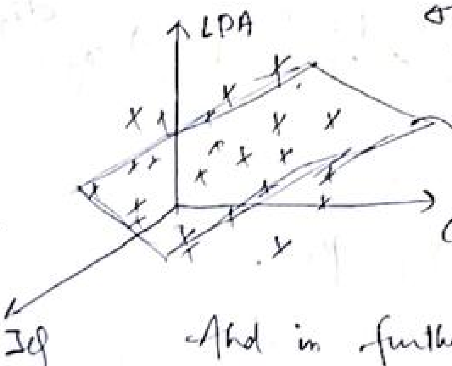
y → output / dependent

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

Consider the dataset :-

cgpa	IQ	package
:	:	:

plot it in 3D



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

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

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

$$y = mx_1 + nx_2 + b$$

or

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

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

Sly in n-dim:- we would be searching the eqn of (n-1) dimensional hyperplane:-

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

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

The coeff $\beta_1, \cdots, \beta_{n-1}$ 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

↳ finding out $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ (i.e $\beta_0, \beta_1, \beta_2$)

for dataset:-

cgpa	IQ	LPA
⋮	⋮	⋮

and if Data set is:-

cgpa	Gender	IQ	LPA
x_1	x_2	x_3	y → actual y

we find $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$.

($\hat{y}$ ← prediction wala y)

Then plotting this in 4D will be difficult.

Suppose there are 100 students ∴ 100 rows:-

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

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{100} \end{bmatrix} = \begin{bmatrix} \beta_{0,1} & \beta_{1,1} & \beta_{2,1} & \beta_{3,1} \\ \beta_{0,2} & \beta_{1,2} & \beta_{2,2} & \beta_{3,2} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{0,100} & \beta_{1,100} & \beta_{2,100} & \beta_{3,100} \end{bmatrix} \cdot \beta .$$

This is the predicted stud. of all 100 students.

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

The step is iterative. So when we are writing the same in n rows & (m+1) cols

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

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

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

Now: The output Data:-

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

And Here:-

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

in simple Linear Regression:- $E = \sum_{i=1}^{n} [y_i - \hat{y}_i]^2$ → Error, or loss fn

$\Rightarrow E = e^T e = (Y - \hat{Y})^T (Y - \hat{Y}) = (Y^T - \hat{Y}^T)(Y - \hat{Y})$

$= (Y^T - (X\beta)^T)(Y - X\beta) \quad \because \hat{Y} = X\beta$

$= (Y^T - \beta^T X^T)(Y - X\beta)$ → All are matrices so dist. prop holds.

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

$= Y^T Y - Y^T X\beta - (X\beta)^T Y + (X\beta)^T X\beta$

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

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

so $Y^T X\beta = (X\beta)^T Y$ → or Basically we need to prove $(Y^T X\beta)^T = Y^T X\beta$ → & $Y^T X\beta$ is symmetric
↳ as it is 1×1 matrix hence it is symmetric

$\Rightarrow \boxed{E = Y^T Y - 2Y^T X\beta - \beta^T X^T X\beta}$
↳ Error or loss function

$E = y^T y - 2y^T X\beta + \beta^T X^T X \beta$ → we need to minimize this choose β so that this gets minimized.

same method (diff wrt β) :-

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

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

↳ matrix differentiation

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

$$y^T X = X^T X\beta^T$$

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

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

$$\beta = (X^T y)\,(X^T X)^{-1}$$ → Nav:

Shape of β: $((m+1), 1)$

Shape of $(X^T y)(X^T X)^{-1}$: $(m+1, 1)$

& $X = X\_train$, $y = y\_train$.

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

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

Why Gradient Descent ?

Scikit-Learn uses 2 methods to solve Linear Regression problems
- OLS (Closed form) → Here we find formula The one we found above.
- (non closed form) → Gradient Descent

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

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

Hence if we have :- text data or image based data (which have many cols) and we have to apply Multiple Linear Regression then we use Gradient Descent.

scikit Learn
- OLS LinearRegression class
- GradientDescent SGDRegressor class.

approximation Technique.

Jupyter Notebook.

Gradient Descent

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

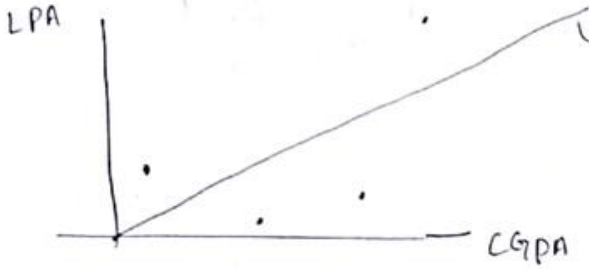
The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at a current point, because this is the direction of the steepest descent. Conversely, stepping in the direction of the gradient will lead to a local maximum of that function, this procedure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

* Gradient Descent is a General Algorithm
  - ↳ Linear Regression
  - ↳ Logistic Regression
  - ↳ 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 (i.e. CGPA / LPA of 4 students)



→ Regression line (say) (best fit line).

⇓

mean the Error = loss fⁿ

Error (Sq of Actual y − predicted y) should be minimum.

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

say m = 78.35

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

↳ depends only upon b.

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

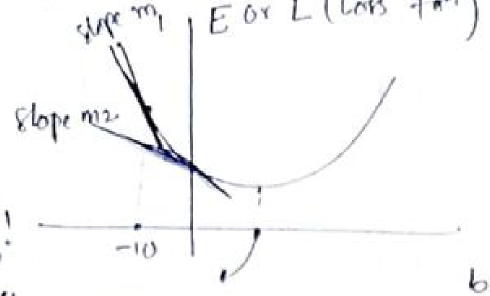
$$E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

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

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

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

The Rel^n b/w ~~w~~ and b is quadratic ∴ (infact upward parabola)



clearly $m_2 < m_1$
∵ after θ > 90° θ ↑ses tan θ ↓ses ∴ slope ↓ses.

→ This is the value of b where L is min.
→ This is what we want.

Now we start with Gradient Descent

Step 1: select a Random b 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 ↓ses. There are two directions to move, how would the Algo know which direction to move so that the f^n value ↓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 → +ve then ↓se b
if slope is −ve then ↑se b.

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

This eq^n will bring drastic changes in values of b, ∵ we use iterations. Hence we use a transformed eq^n:

$$b_{new} = b_{old} - \eta \, \text{slope}$$; where ; η = Learning Rate
↳ generally it's value is 0.01 (we can keep it 0.1 or 0.0001 etc)

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

- 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^n vv slowly.

The step is iterative, so when do we stop.

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

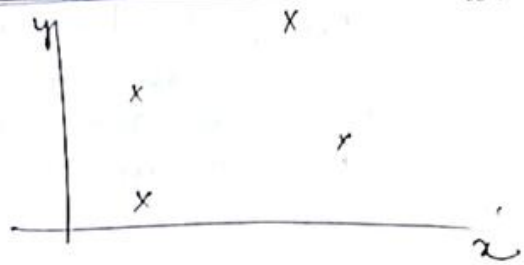
$\Rightarrow b_{new} - b_{old} \rightarrow -\eta \cdot slope$ (which should be very small)

means the slope is becoming 0 or 1 to x-axis hence we are reaching our point of min.

Or: We can fix the no. of iterations (called epochs) (say 100 or 1000), i.e. how the 'hi' iterations converge & the 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 (no of iterations):-

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

$b_{old} = b_{new}$

How to find?

This is Gradient Descent

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

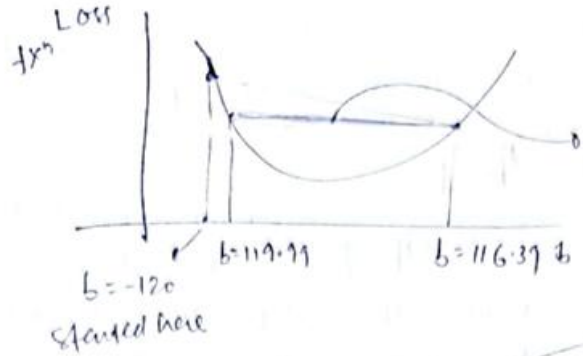
$$\frac{dL}{db} = \sum_{i=1}^{n} -2(y_i - mx_i - b)$$

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

Note: $\eta \cdot$ slope at b_{old} = step size.

JupyterNotebook for 1 upto this part.

Gradient Descent → file 2 → Drawing 1:



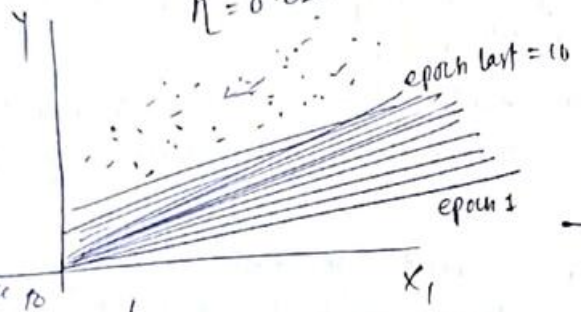
Free Discussion.

Say epoch = 10

$\eta = 0.1$

① Effect of Learning Rate

$\eta = 0.02$

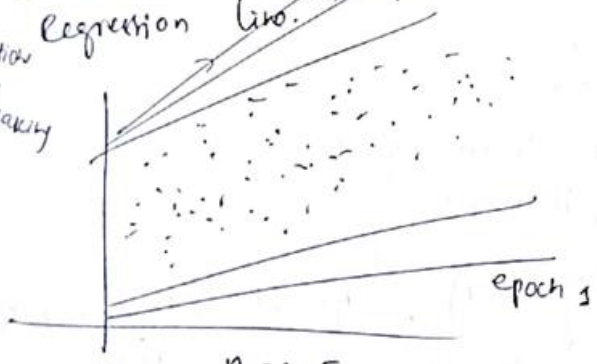


in this case to reach the actual reg. line we need more epochs means more calculations & more time taking

↓

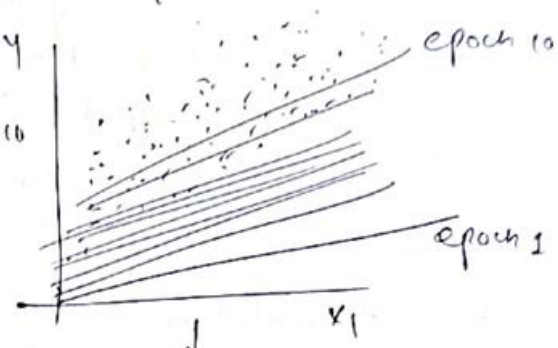
Learning rate too low

Cannot reach the actual Regression line. epoch 10



$\eta = 0.5$

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. chalakar hi dekhar lege. 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 wrt Linear Regression, but it can be applied along with other Algorithms also.

In some other Algo, the loss function may be different & then again we can use the Gradient descent similarly to find the parameters so that the loss fn is minimum. The only cond^n is 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

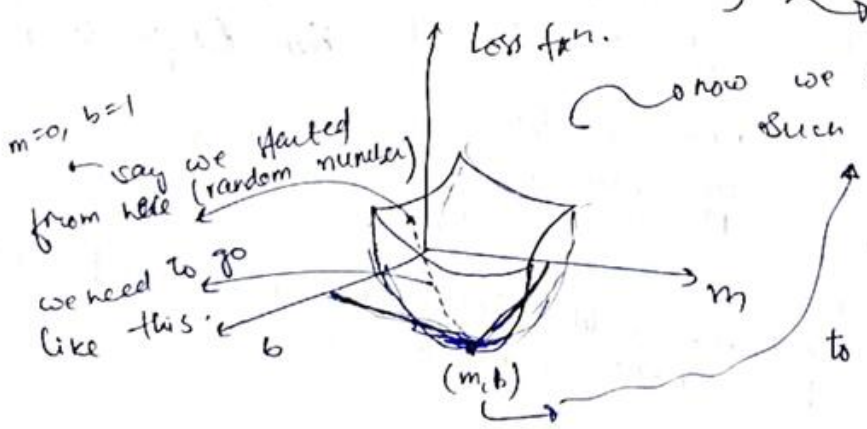
for i in epochs:.

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

update m:. $m = m - \eta \cdot \text{slope}$ ← 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 → it will have two components; one component in direction of m & other in direction of b.

& diff of Multivariable func is gradient. ∴ In the formulas.

$b = b - \eta \cdot \text{slope}$ ← this is slope wrt b i.e. $\frac{\partial L}{\partial b}\Big|_{b=b_{old}}$

& $m = m - \eta \cdot \text{slope}$ ← This is slope wrt m i.e. $\frac{\partial L}{\partial m}\Big|_{m=m_{old}}$

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

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

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

(m_{old}, b_{old}) is particular point jaha pe currently hai

Nao same can be extended to higher dimensions → dataset with more than one input col.

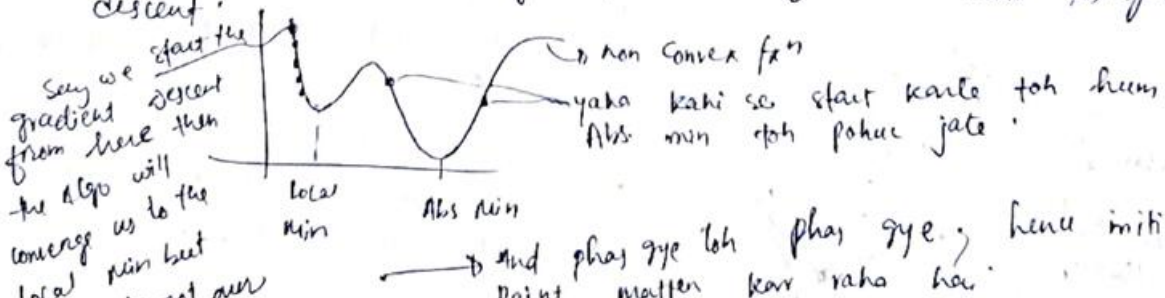
Jupyter Note Book.

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. ← Such fns are good for gradient descent

Since they have only one minima (Global minima).

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



If for higher Dimensions say we have 10 cols, we cannot check for convex hai ya nahi & agar initialisation agar aisa hua ki humara Algo local minimum pe convergence kar gya then we have to accept it.

↳ isse bachne ke kuch tarike hote hai like trying for initialization, depending on cost function. (this we do in deep learning :) wahi pe jyada how convex for milte hai.

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

Effect of Data on Gradient Descent:

Agar features in the data are equally scaled then the contours plot of the cost fn/loss fn 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

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini Batch Gradient Descent.

What is the basis of this Types/distinction:

we had the eqn:.

$$m_{new} = m_{old} - \eta \left(\text{slope} = \frac{\partial L}{\partial m}\Big|_{m=m_{old}} \right) \quad \because L = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

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

Now in Batch Gradient Descent :- To calculate the slope we use the entire data. & the 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 fn/error & make changes in m & b → faster with large data set but Error prone.

∴ pure dataset pe ek baar ghumne pe we make 300 updates considering there were 300 total rows/data

Again there is

Mini Batch Gradient Descent (b/w 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:-
→ It is used generally when we have convex loss fn like in Linear Regression. but still there is one cond^n that the Dataset should not be big.

Batch Gradient Descent :-

We have already studied Batch Gradient Descent.

So Let's study Batch Gradient Descent for more than one input variable/feature

Say the dataset is:-

	input		output
Cgpa	IQ	Gender	LPA

Now the Regression hyperplane will be:-

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$ → 4 dimensional.

(y → LPA, x_1 → Cgpa, x_2 → IQ, x_3 → Gender)

So if we have n input col^s:

then $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$ → This will be the eq^n of the Regression hyperplane

Hence we need to find

$\beta_0, \beta_1, \cdots, \beta_n$.

mathematically formulating the n-Dim Batch Gradient Descent (Multiple Linear Regression)

To make it easier consider :-

rows = 2
col = (2+1)

	CGPA	IQ	LPA
	x_{11} 8.1	x_{12} 93	3.2
	x_{21} 7.5	x_{22} 95	3.5

→ we will extend this.

→ eqn of reg plane : $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ → To find $\{\beta_0, \beta_1, \beta_2\}$

(y ↓ LPA, x_1 ↓ CGPA, x_2 ↓ IQ.)

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

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

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

Then updating the values for each epochs:

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

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

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

Hence if we have n input cols then we will need $n+1$ such eqns 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 fn = $L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

{ 1/n Lagane se we are calculating Mean squared Error (MSE) } ↓ jo $\beta_0, \beta_1, \beta_2$ MSE ko minimise karegi wo with $\frac{1}{n}$ wale total error ko bhi minimise karegi.

for our above data

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

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

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

$$= \frac{1}{2} (3.2 - \beta_0 - \beta_1 \times 8.1 - \beta_2 \times 93)^2 + \frac{1}{2} (3.5 - \beta_0 - \beta_1 \times 7.5 - \beta_2 \times 95)$$

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

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

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

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

Now:

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

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

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

for n - rows)

$\frac{\partial L}{\partial \beta_0} = -\frac{2}{n}\left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + \cdots (y_n - \hat{y}_n)\right]$

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

Again if we have the same data set: $L = \frac{1}{2}\sum_{i=1}^{2}(y_i - y_i)^2$

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

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

for n rows.

$\frac{\partial L}{\partial \beta_1} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_{i1} = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_{i1}$

x_{i1} is the 1st column ie. Cgpa col. (X_1)

↓ sly.

$\frac{\partial L}{\partial \beta_2} = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_{i2}$

x_{i2} → is the 2nd col of the the data set ie. IQ col (X_2)

So Now if we have (n) input cols: (& n rows)

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

and

slope wrt $\beta_K = \frac{\partial L}{\partial \beta_K} = \frac{-2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_{iK}$ → ★★★ General formula

Jupyter Notebook:

Stochastic Gradient Descent :- (most imp)

The problem in Batch Gradient Descent is that hum coeff (of hyperplane of best fit) ke respect mei Loss fⁿ (L) ka derivative nikalna hota hai to make one update in the coeffs.

& again the Loss fⁿ is a Summation over the rows.

Now Suppose there are 1000 rows (which is vv easy to get on real life problems)
& cols = 5 (input cols)
And also say there is epochs = 50.

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

∵ we have 5 input cols ∴ $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_5 x_5$

∴ to find $\frac{\partial L}{\partial \beta_0}$ → we need to do 1000 derivatives (open L & diff)

sily for $\frac{\partial L}{\partial \beta_1}, \frac{\partial L}{\partial \beta_2}, \cdots, \frac{\partial L}{\partial \beta_5}$

So there are 6000 derivatives in each iteration

∴ for 50 epochs/iteration: there are

$6000 \times 50 = 3.00000 = 3$ lakhs derivatives.

Now if we have :- rows = 10^5 ; & cols = 10^2
& epochs = 10^3

then Total derivatives to be calculated till the loop complete = 10^{10} → huge computation

hence our Algo will become very slow on Big Data.

Hence for Big Data sets → we use stochastic Grad. Descent.
(like in Deep Learning + CNN (on images) + RNN (on Texts))

On Small 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 rakha hoga tabhi dot product ho payega but sometimes X_train may be huge

then the Calculation of y_hat will not be possible.
So there is a Hardware problem (i.e. if the system RAM is small)

In Batch Grad Descent :- After Reading the entire data (X_{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. (randomly selected (not sequentially))

So in a single iteration/epoch → there are # of rows iterations.

& Because of these frequent updates we reach the correct answers in less epochs. (converges to mean Stochastic)

Hence in Stochastic Grad Descent :-

+ faster convergence (since less # of epochs)
+ we do not have to load the entire data at a time hence no hardware problem.

Stochastic Meaning :- having a Random prob. distribution that may be analysed statistically but may not be predicted precisely.

Problem with Stochastic GD is → if we run the algo on the same dataset we get different answers each time & all are closer to the actual answer — this is due to the random selection of rows.

Jupyter Notebook.

In Stochastic :- we select a Random row (say k^{th} row) &

Loss fn becomes :-

$$L = (y_k - \hat{y}_k)^2$$

$$\& \quad \frac{\partial L}{\partial \beta_0} = -2(y_k - \hat{y}_k)$$

$$\& \quad \frac{\partial L}{\partial \beta_j} = -2(y_k - \hat{y}_k) \cdot x_{kj}$$

$x + y$ → then is a quad. eqn

Time Comparison:-

Say we use same epochs = 100 for both stochastic & Batch GD with Batch GD will be faster than Grad. desc. Since in BGD there are only 100 update to be made but in SGD there are 100 × # of rows update to be made.

But since SGD will converge to a soln. before 100 epochs ∴ 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

- 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.

but for this learning rate to monitor karna padega.

& due to this it may jump out of the local minima of the convex fn [The problem we faced with Batch GD).

line joining these two points cuts the fn hence fn is Non Convex.

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 use:- $t_0, t_1 = 5, 50$ → we can define def^n our own accordingly.

due to this as epoch goes ↑, $\frac{t_0}{t+t_1}$ ↓

∴ t ↓

∴ Lr ↓; & hence fluctuation stabilizes towards end.

```
def learning_rate(t):
    return t0/(t+t1)
for i in range(epochs):
    for j in range(X.shape[0]):        # of rows
        lr = learning_rate(i*X.shape[0]+j)
```

multiply

Hence now Lr is a fn of epoch i

The SGDRegressor() class provided by Sklearn has a Parameter called learning_rate = 'constant', 'optimal', invscaling', 'adaptive' → to handle Learning schedule.

↓

The differences in the types of grad. descent is based on how frequently we update the intercept & slopes

It is a comb of Batch & stochastic. ← used mostly in deep learning

Mini Batch Gradient Descent

→ suitable for small & large loss fn. data

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 → there are random batches
total rows = 1000 ∴ total batches = 10 of 100 rows out of 1000 rows,

∴ in each epoch. #of batches update ie 10 updates in 1 epoch.

↓

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

↓ Mini Batch improve (less) the Randomness in the Stochastic GD.

Jupyter Note Book.

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.
& $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$ → 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:- | X | Y | 35 | 12 | & there is a quad. rel^n b/w them

then we find | x^0 | x^1 | x^2 | Y | 1 | 35 | 35^2 | 12 → & send this as x_train & y_train.

And the coeff we are finding is:-

$y = \beta_0 + \beta_1 x + \beta_2 x^2$ → $(\beta_0, \beta_1, \beta_2)$.

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 |

& eqn is:- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$

& we have to find $(\beta_0, \beta_1, \beta_2, \beta_3)$

How to find which degree we need to impose:-

↓ degree is a Hyperparameter

ML Engineer predicts this.

if this value is less than the Regression curve will underfit & data ki saari attributes/properties fetch na kar paye.

if this value is high than the Regression curve will overfit (it tries to touch every point)

What if Dataset is:- | X_1 | X_2 | Y | → Then

& degree = 2

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

So the DataSet becomes:-

| x_1^0 | x_1^1 | x_1^2 | x_2^1 | x_2^2 | y |

Now Consider this:- ↓ copy just change the col names

| $\bar{x}_0$ | $\bar{x}_1$ | $\bar{x}_2$ | $\bar{x}_3$ | $\bar{x}_4$ | Y | → 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 solving Multiple Linear Regression

Slope na vaam →

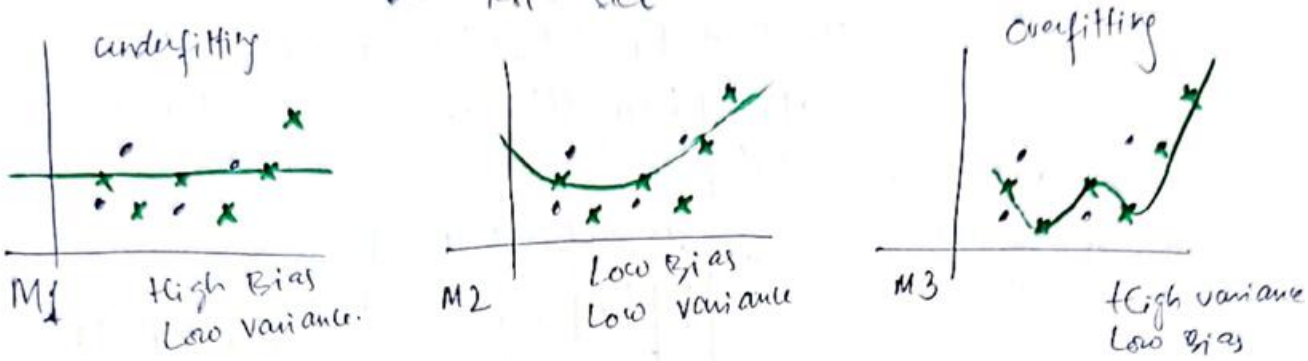
***

Bias Variance Trade off

This concept is applicable on Every ML Algorithm.

Consider a data set & on it we apply three different ML models:-

x - training set
• - 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 rel^n ships in the training data.

Le ie. the ML model training data pe bhi accha perform nahi kar pa raha. Then we call it bias (since wo training data ko samajh hi nahi paa raha).

M3 → low bias

M2 → There is curve in the training data & the model is able to capture that curve.

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

Variance :- It is the difference of fit on different data set.

Say we have training data set & test data & the model give error of 10 on training data & error of 100 on test data → This means variance is 90. (100-10)

model 1 & model 2 (M1 & M2) :- will have low variance since they are closer to the test data points also hence error in train data - error in test data is not much.

in M3 :- The error in training data is almost 0 & error in test data set is high → ∴ High variance.

Overfitting: When the model works good on the training data but doesnot work good on the test data. => Training Error is extremely low but the test Error is high => Variance is high.

Underfitting: Model jo training data pe hi accha kaam nahi kar pa rha → High bias.

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

* Kabhi kabhi ke keep a bit bias so that variance kam. 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

↳ 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)

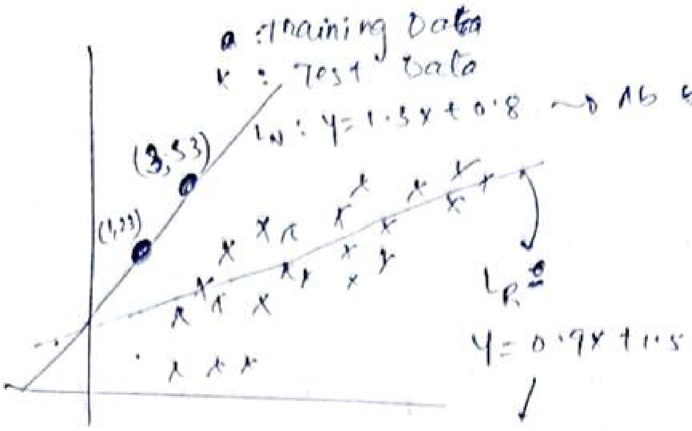
Let's talk overfitting wrt Linear Regression :-

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

in Linear regression if we have overfitting the the value of m will be extremely high. and if value of m is very low then it will be underfitting.

so Regularization for Linear Regression means that slope ka value ko thora kam karna hai.

considers.



→ Ab bas do Points pass Calculate/train karke Line of Best fit nikalna hai then un dono sei pass karne wala line hi nikelega.

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

Consider this the hypothetical line jo training Data pe kuch galtiya karega but test data pe better perform karega.

So humme ab kissi tarike se ML 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 f^n :- → 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 f^n :-

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

→ m is the slope

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

Now Let's see using this how we can convince our ML model to choose L_R instead of L_N.

Loss L_N ($\lambda=1$)	Loss L_R ($\lambda=1$)
$L = 0 + 1.5^2 \times 1$ $= 2.25$ Since L_N pass hi un dono ke upar se kar raha hai	$L = \sum_{i=1}^{2} (y_i - \hat{y}_i)^2 + m^2\lambda$ $= (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (0.9^2 \times 1)$ $= (2.3 - 0.9\times1 - 1.5)^2 + (3.3 - 0.9\times3 - 1.5)^2 + 0.9^2$ $= 2.03$

$y_1 = 2.3$ & $y_2 = 3.3$
$x_1 = 1$, $x_2 = 3$
$\hat{y}_i = m x_i - b$

Hence Now the ML model will choose the LR 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.

→ [goose although Bias there bath are but Variance significantly that are. → And this is called Bias-Variance trade off]

Note:- if there was two input cols & a output col^n then :-

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda (m_1^2 + m_2^2)$$ → since sum square $\frac{1}{2i}$ multiply kar rahe hai hence it is L2 Norm hence the name L2 Regularization.

n = tq data in training dataset

sily if there are k input cols then :-

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda (m_1^2 + m_2^2 + \cdots + m_k^2)$$

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 :-

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

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

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

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

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

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

- minima point

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

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

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

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

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

→ This is the new m ; note that $\lambda \geq 0$

earlier $m = \dfrac{\sum_{i=1}^{n} (y_i - \bar{y}) + (x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$; hence

Since now we are adding $\lambda \geq 0$ to denominator, m decreases & ∴ bias ↑ but variance decreases & 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 | ... | X_n | Y | ~ $(n+1)$ cols & m rows

$Y = w_0 + w_1 x_1 + \cdots + w_n x_n$

So we need to find $(w_0, \ldots w_n)$

$w_0 \rightarrow$ intercept & $w_i \rightarrow$ slopes/weights

Loss fn in this case :-

we learnt earlier that

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

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

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \quad \& \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & & \\ 1 & x_{m_1} & \cdots & & x_{mn} \end{bmatrix}$$

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

$$L = (Xw - Y)^T (Xw - Y) + \lambda \|w\|^2$$

$\|w\|^2 = w_0^2 + w_1^2 + \dots + w_n^2$ → L_2 Norm square.

$$L = (Xw - Y)^T (Xw - Y) + \lambda w^T w$$

↳ This is the Loss fn for Ridge Reg for n Dim Data.

We need to find W matrix:

$$L = ((Xw)^T - Y^T)(Xw - Y) + \lambda w^T w$$

$$L = (w^T X^T - Y^T)(Xw - Y) + \lambda w^T w$$

$$L = w^T X^T X w - w^T X^T Y - Y^T X w + Y^T Y + \lambda w^T w.$$

↳ These two terms are the same (did earlier)

Now:

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

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

Now $$\frac{dL}{dw} = 2X^T X w - 2X^T Y + 2\lambda w = 0$$

$$w(X^T X + \lambda I) = X^T Y \quad ; \quad I_{nx1, nx1}$$

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

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

↳ earlier it was:-

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

Some common vector derivatives:-

$X^T B \xrightarrow{d/dX} B$

$X^T b \rightarrow b$

$X^T Y \rightarrow 2X$

$X^T B X \rightarrow 2BX$

code for this part in my Jupyter Note Book.

Now: What if the data set is huge & we have multiple input cols → Then Calculation for the inverse will take a lot of time & hence we'll therefore

Ridge Regression Technique using Gradient Descent

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

Loss fn with added regularization term

$$L = (Xw - Y)^T (Xw - Y) + \lambda \|w\|^2$$

choose starting $w = [0, 1, \ldots, 1]$ for Gradient descent.

$$L = (Xw - Y)^T (Xw - Y) + \lambda w^T w$$

where the terms have the same meanings as discussed in previous section

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & & \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}; \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}; \quad w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \begin{matrix} \rightarrow \text{intercept} \\ \} \text{ slopes.} \end{matrix}$$

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

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

↓ in a single formula

$$w_{new} = w_{old} - \eta \frac{\Delta L}{\Delta w} = w_{old} - \eta \, \text{grad} L$$

$$\begin{bmatrix} w_{0_{new}} & \cdots & w_{n_{new}} \end{bmatrix} = w_{old} - \eta \nabla L = \begin{bmatrix} w_0 & \cdots & w_n \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial L}{\partial w_0}, & \cdots & \frac{\partial L}{\partial w_n} \end{bmatrix}$$

we have:- $L = \frac{1}{2}(Xw - Y)^T (Xw - Y) + \frac{1}{2} \lambda w^T w$

($\frac{1}{2}$ multiply kiya for mathematical convinience ↓ minimization karna hai.)

$$L = \frac{1}{2}\left[w^T X^T X w - 2 w^T X^T Y + Y^T Y \right] + \frac{1}{2} \lambda w^T w$$

$$\frac{\partial L}{\partial w} = \frac{1}{2}\left[2 X^T X w - 2 X^T Y \right] + \frac{1}{2} 2 \lambda w$$

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

Choose start $w = [0, 1, 1, \ldots, 1]_{1, n+1}$ then in each epoch 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 bcz & Bias ↑ses & variance ↑ses.

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 ↑ λ then the coefficients (coefficients i.e. slopes) starts ↓ing towards '0' but never '0'.

see jupyter Notebook.

2./ Higher coefficients are affected more.

Say:

x_1	x_2	x_3	y

→ this means they y calculate karne ko like x_1 is most imp. then x_2 & then x_3 is least important.

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

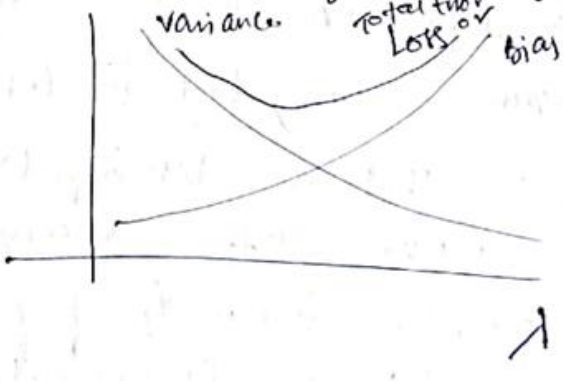
Now as we ↑ λ then w_1 will be affected the most (↓ses faster) & w_3 is the least affected (↓ses slowly)

3./ Effect of Regularization on Bias Variance Tradeoff:-

As λ ↑ses the Bias ↑ses but the variance ↓ses upto some point

but if we continue ↑sing it then Bias will ↑ but the curve may underfit.

if λ is very small then Bias ↓ses & Variance ↑ses 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 fn.

The Loss fn:- $L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \|w\|^2$ (this if we divide by n then it becomes mean squared error (MSE)).

$\lambda \|w\|^2 \geq 0$ always positive, hence, the loss fn ↑es upon regularization

* Mostly Ridge Reg is applied when # of input cols is ≥ 2.

Lasso Regression or L1 Regularization → It also helps in reducing the overfitting.

in Ridge Regression: we added a Reg. term

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \|w\|^2 ; \quad \|W\|^2 = w_0^2 + w_1^2 + \cdots + w_n^2$$

w_i → coefficients.

↓ in Lasso also we do the same..

here we just use L1 Norm:-

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

Jupyter NoteBook — using Scikit Learn.

as λ ↑es we try to min $\|w\|$ → & decreasing too much may cause underfitting.

In Ridge Reg → no matter how much we ↑sed alpha/lambda, the weff never became zero, but in Lasso, as we ↑ lambda, the x_i's that are less important to determine y (output) wrt coeff becomes zero.

↳ Hence Lasso regression inherently performs feature selection & helps in dimensionality Reduction.

Hence Lasso is preffered over Ridge for high dimensional data.

Why Lasso Regression creates sparsity?

We see as the value of λ ↑ then the coefficient (slope) in case of Lasso regression becomes zero → Thus creates sparsity

As the λ value becomes very high → all the coefficients become zero → error does this happen?

Consider one input col & one output col to understand:.

x | y → Linear Regression case:- we have to find the line $y = mx + b$

& $b = \bar{y} - m\bar{x}$ we know. $\bar{y}$ = mean of y, $\bar{x}$ = 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

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

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

Now Let's see what the formula becomes for the Lasso Regression:. $b = \bar{y} - m\bar{x}$ (same)

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

This 2 is added just for mathematical conv.

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

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

$m > 0$.

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

$m < 0$.

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

$m > 0$.

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

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

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

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

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

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

(this is fixed)

clearly as $\lambda \uparrow^{es}$ the value of m $\downarrow$es. & we can $\uparrow^{se}$ λ s.t. $m = 0$ & if we further $\uparrow^{se}$ λ then m gets -ve. value !

but -ve value not possible since in that case we use this formula

if $m = 0$. then we'll have Simple Linear Reg & then $m = \dfrac{\sum (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$

& 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 $m > 0$. $m = \dfrac{Yx - \lambda}{x^2}$

Say $Yx = 100$ & $x^2 = 50$

$\lambda = 0 \rightarrow$ we get $m = 2$

$m = \dfrac{100 - \lambda}{50}$

as $\lambda \uparrow^{es}$ m $\downarrow$es & as $\lambda = 100$ $m = 0$

& if $\lambda > 100$ then m gets -ve but as m gets negative this formula is used

& if $m < 0$. $m = \dfrac{Yx + \lambda}{x^2}$

$m = \dfrac{100 + 150}{50} = 5$ → Real slope increased even more than the simple linear ($\lambda = 0$) reg case → not good

Hence the algorithm stops increasing or using m after $m = 0$.

Nao if $m < 0$.

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

$\because \lambda \geq 0 \therefore$ for $m < 0$; $Yx < 0$ say $Yx = -100$ & $x^2 = 50$

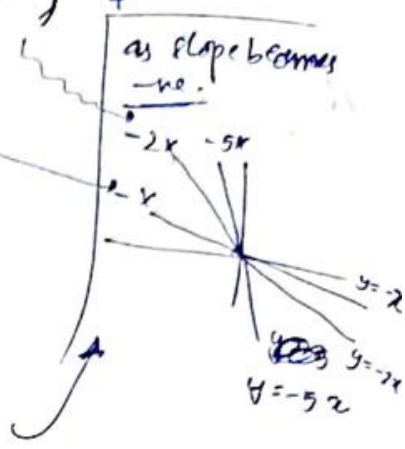
$m = \frac{-100 + \lambda}{50}$ → Nao as $\lambda \uparrow$ seq $m \downarrow$ seq (simple Linear case: $\lambda = 0$ $m = -2$)

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

$\lambda = 100 \rightarrow m = 0$

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

$$m = \frac{Yx - \lambda}{x^2} = \frac{-100 - 150}{50} = -5$$ → 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 since. waha Lambda (λ) deno mei divide ho raha hota hai ; upar num mei add ya subtract nahi.

Hence: λ kitna bhi bada karo. coeff zero nahi hoga. chaen na kuch 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 → then we use Elastic Net Regression.

The formula for Loss fn in Elastic Net Regression is :-

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \overset{0.5 \times}{\lambda} a \|w\|^2 + b\|w\|.$$

in Scikit Learn we two hyper parameters :-

$\lambda = a + b$ & l1_ratio $= \frac{b}{a+b}$.

By default:- $\lambda = a+b = 1$ & l1_ratio = 0.5)

& $\therefore a = 0.5, b = 0.5$

But say: l1_ratio = 0.9 → 90% we use Lasso Reg & 10% Ridge Regg.

$\therefore$ The Scikit learn with $(\lambda, l1\_ratio)$ → is used in Elastic net Reg., kitna % Ridge lagaye & kitne % Lasso lagaye.

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

↳ Ex: weight | height

Jupyter Notebook.

$Ax+By+C=0$ → decision line

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?

1. Dataset which are Linearly Seperable or almost Linearly Seperable (i.e. ek do data points yaha wala ho wala ho sakta hai)

Line seperating the categories of the output

in 3-D its a plane & in higher dim. it's a hyperplane.

Perceptron Trick

Consider the same above dataset & we have to predict for a given new child ki uska placement hua ya nahi.

The data set is Linearly seperable & Eqn of line used in Logistic Regression is $Ax+By+C=0$ (General Eqn)

This is the line that seperates the dataset.

$\Downarrow$

$Ax_1+Bx_2+C=0$

x_1 → cgpa, x_2 → IQ.

$\Downarrow$

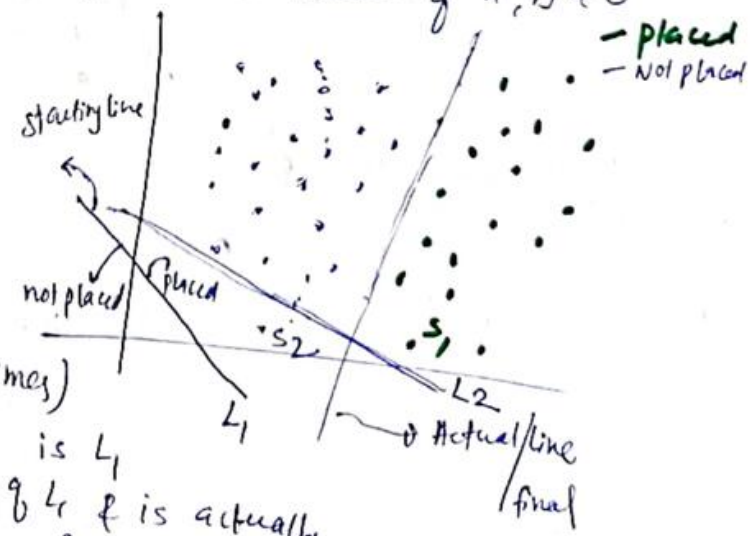
Hence to find A, B & C.

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 accordig to this line everyone is placed.



Now we run a loop (mostly epoch 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 donot change the value of A, B, C ; Again in 2nd iteration of loop we pick S_2

& ask S_2 : is L_1 OK → 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 L_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 got our actual Answer/Line.

↓

or we can also run the loop till convergence. (i.e. No. of mis classified points = 0)

1st On:- for the initial line, how do we decide which side is green & which side blue (not placed)
(placed)

How to decide pot region & -ve Region.

say initial line is :- $2x+3y+5=0$

the pot region is :- $2x+3y+5>0$

& -ve region is :- $2x+3y+5<0$.

↓ using this we decide if a point is in pot region or a -ve region.

Transformations :- (wrt a point → jis pt ke taraf move karna hai)

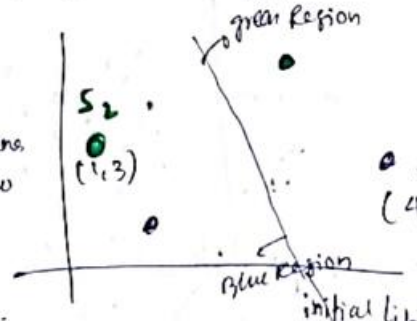
Line is :- $Ax+By+C=0$

change in C → parallel shifts ↑ se C we shift ||ly down
↓ se C " " " up

change in A → rotation about y-intercept

change in B → Rotation about x-intercept.

Consider.



Jab line ko -ve direction mei move karna hai then add karo

↑

Again. S_2 says the initial Line is faulty :-
so transforming :-

$$\begin{array}{r} 2\ 3\ 5 \\ +\ 1\ 3\ 1 \\ \hline 3x+6y+7=0 \end{array}$$

↳ for this line S_1 will be on pot/green side.

say $2x+3y+5=0$

Jab line ko pot direction mei move karna hai tab subtract karo-

↑

S_1 says line is not good so we move line towards $S_1 \rightarrow (4,5,1)$

$$\begin{array}{r} 2\ 3\ 5 \\ -\ 4\ 5\ 1 \\ \hline -2x-2y+4=0 \end{array}$$

↳ for this line S_1 will be in blue region (-ve. Region)

$Ax+By+C=0$ → existing line

Thus Summary of Transformation is :- current

Agar apka pot point (x_1, y_1) -ve side mei hai then we need to move the line to -ve side to make (x, y) on pot side for that new line :-

A	B	C
+ x_1	y_1	1

$(A+x_1)x + (B+y_1)y + (C+1) = 0$

& if B -ve (x_1, y_1) point is on pot side then we need to move the line to pot side to make the point -ve →

Transformation is :-

A	B	C
− x_1	y_1	1

$(A-x_1)x + (B-y_1)y + (C-1) = 0.$

But these transformations are huge hence we use η = Learning rate

$\eta = 0.01$ or 0.1 generally & Transformations then become :-

A	B	C
± ηx_1	ηy_1	η

$(A \pm \eta x_1) + (B \pm \eta y_1) + (C \pm \eta) = 0$ → with this the change / transformation is not abrupt → but it still does the job.

Algorithm for perceptron trick :-

	cgpa	IQ	placed
S_1	7.5	81	1
	8.9	109	1
	7.0	81	0

→ The eqn of line will be :- $Ax + by + c = 0$

→ for this Data :- $w_0 + w_1 x_1 + w_2 x_2 = 0$ (cgpa, IQ)

$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$ (assuming there is a col x_0 of all ones)

$\Rightarrow \sum_{i=0}^{2} w_i x_i = 0$ → General eqn of line.

How does the make prediction :- This is our model.

say for S_1 :- (we have w_0, w_1, w_2)

$w_0 \times 1 + w_1 \times 7.5 + w_2 \times 81$ → ≥ 0 means placed, < 0 means not placed.

$(w_{eff} = (w_0\ w_1\ w_2))$ ▪ $((1, 7.5, 81) = (1, cgpa, IQ) = (1, x_1, x_2))$ dot product → ≥ 0 placed, < 0 Not placed

$$\begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

Algorithm is:

epochs = 1000 generally ; η = 0.01 (Generally) ← Learning rate

for i in range (epochs):

randomly select a student

if y_i ∈ Negative Region and $\sum_{i=0}^{2} w_i x_i \geq 0$ → i.e. student actually belongs to +ve Region but model (current line) is predicting ↑ $\hat{y}_i$

↳ $y_i = 0$

update coeff.

placement nahi hua but model ne bola ho gya

$W_{new} = W_{old} - \eta X_i$

$= [w_0, w_1, w_2] - \eta [x_0, x_1, x_2]$

C A B

This is 1

placement hua but model ne bola nahi hua.

if X_i ∈ Pot Region and $\sum_{i=0}^{2} w_i x_i < 0$

↳ $y_i = 1$ ↳ $\hat{y}_i$

$W_{new} = W_{old} + \eta X_i$

$= [w_0, w_1, w_2] + \eta [x_0, x_1, x_2]$

→ Point ∈ Pot Region but current line shows it is in -ve Region so the line should move to -ve direction to make the pt. in +ve Region ∴ add the coeff of current line to (coor of pt, 1).

↓ Simplifying the Algo

for i in range (epochs)

select a random student:

update: $W_{new} = W_{old} + \eta (y_i - \hat{y}_i) x_i$ ← without any if else

predicted output

Why it works.

Actual output

y_i	$\hat{y}_i$	$y_i - \hat{y}_i$	
1	1	0	$W_{new} = W_{old}$ ✓ actual placement hua & model ne bhi bola placement hua.
0	0	0	$W_{new} = W_{old}$ ✓ placement nahi hua & model ne bhi bola nahi hua.
1	0	1	$W_{new} = W_{old} + \eta X_i$ → placement hua but model ne bola nahi hua.
0	1	-1	$W_{new} = W_{old} - \eta X_i$ → placement nahi hua but model ne bola ho gya. (as per Algo)

Jupyter Notebook.

So far we had perceptron trick :-

using $W_n = W_0 + (y_i - \hat{y}) x_i$, 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 paas bulayenge (as always). ↓

ye line ko apne se dur bhejenge

isse kya hoga ki :- jab saare points classify hojainge & line bilkul hai then good points line ko dur bhejenge & agar jyada dur bhejdiya toh dusre wale class ke points phir is dur 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

inversely

jyada distance → kam push
kam distance → jyada push.

↓ ↓ This is logical also.

This is the change that Scikit 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.

Hamne jo dono cases mei bhi change karne hai ie. in cases mei $W_{new} = W_{old}$ nahi chahiye ie $\eta (y_i - \hat{y}_i) x_i$ ko zero nahi hona chahiye

classified points

Yaha change nahi ho rhe the in perceptron trick ie $W_{new} = W_{old}$

y_i	$\hat{y}_i$	
1	1	
0	0	
1	0	→ Yaha change ho rhe the (mis classified points)
0	1	

→ $y_i - \hat{y}_i \neq 0$ hona chahiye

$\hat{y}_i$ 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 (0) aise hi $\hat{y}_i$ nikalte tha.

pehle hum predicted placed & not placed ko 0,1 se represent kar rhe the ab value aur se rep. karna chahte hai.

Now :- $y_i \leftarrow 0, 1$ values leta hai

This is data Given (cannot change)

& if $\hat{y}_i$ bhi 0,1 value lega toh $y_i - \hat{y}_i = 0$ hoga hi

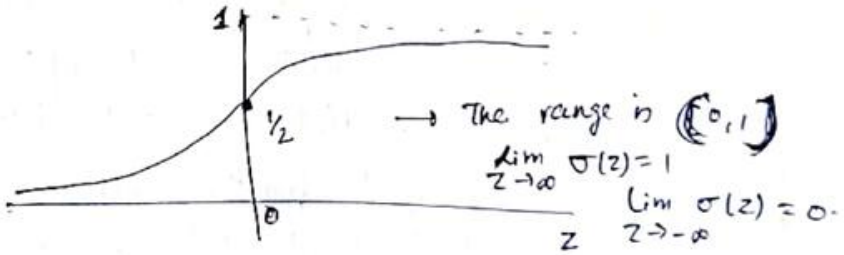
Hence target is ki $\hat{y}_i$ 0,1 ke alawa kuch aur values le. $\hat{y}_i$ ka calculation mechanism change karna hoga.

Hence → in the perception trick we were using step function which gave 0,1 to find $\hat{y}_i$ → Now we need to use something else function to find $\hat{y}_i$. instead of step func.

The function used is :- Sigmoid function :-

The Sigmoid fn.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Domain $(-\infty, \infty)$

Now to Calculate $y\_hat = \hat{y}_i$ $w_1 x_1 + w_2 x_2 + w_0 > 0 \rightarrow 1$ $< 0 \rightarrow 0$ ↓ step fn

Now we replace the step fn with Sigmoid fn

$\hat{y}_i$ = Sigmoid ($\underbrace{w_1 x_1 + w_2 x_2 + w_0}_{T}$) → $T > 0 \rightarrow \hat{y}_i > 0.5$

→ $T < 0 \rightarrow \hat{y}_i < 0.5$

So given input :- $\{7.5, 81\}$ (x_1, x_2) → placement hoga ya nahi?

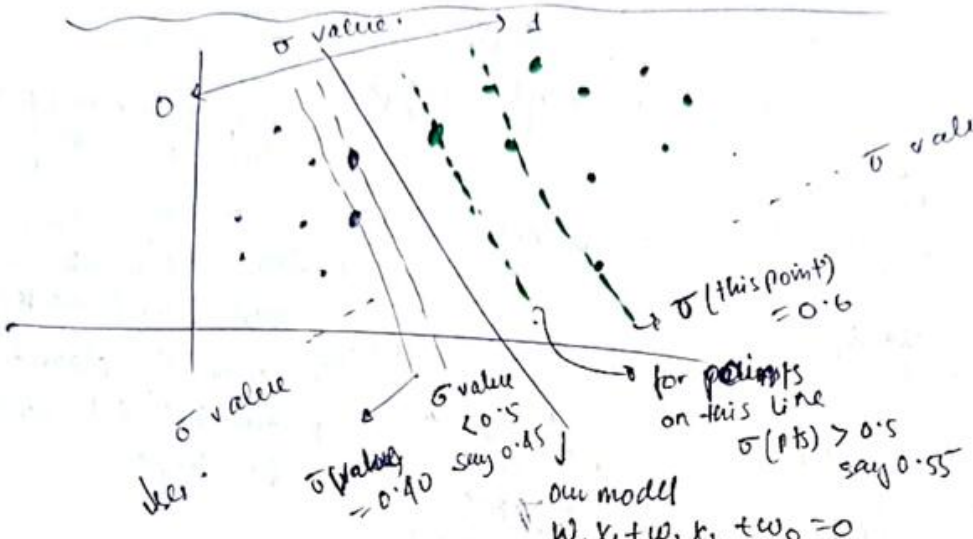
then $T = x_1 w_1 + x_2 w_2 + w_0$

$T = w_1(7.5) + w_2(81) + w_0$

> 0 → then $\sigma(T) > 0.5$ → placed

< 0 → then $\sigma(T) < 0.5$ → not placed.

Let's see how the sigmoid fn works.



So Now what the sigmoid fn represent is probability dist of placed.

on a line say σ value = 0.5 mean P(placed | pts. on that line) = 0.5

if σ value = 0.6 → P(pts on that line being placed) = 0.6

if σ value of a line = 0.4 → P(pts. on that line being placed) = 0.4.

∴ we can find P(Nahi hoga) = 1 − P(placed)

Hence ab har 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.

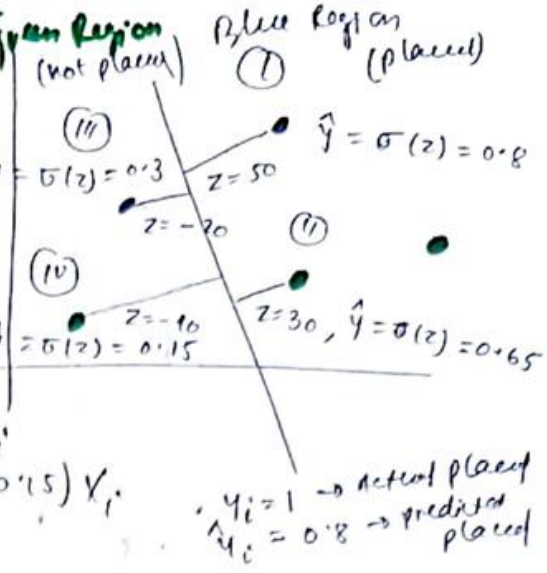
$$W_n = W_0 + \eta (y_i - \hat{y}_i) x_i$$

$$\hat{y}_i = \sigma(z), \quad z = \sum_{i=0}^{2} w_i x_i \quad \& \; x_0 = 1$$

Consider an Example :- $\hat{y}_i > 0.5$ mean predicted placed else predicted not placed.

y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
1	0.8	0.2 → $w_n = w_o + \eta(0.2) X_i$
0	0.65	−0.65 → $w_n = w_o + \eta(-0.65) X_i$
1	0.3	0.7 → $w_n = w_o + \eta(0.7) X_i$
0	0.15	−0.15 → $w_n = w_o + \eta(-0.15) X_i$

↓ not zero for any point (unlike step fxn)



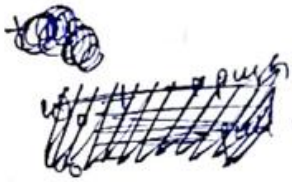
$y_i = 1$ → Actual placed
$\hat{y}_i = 0.8$ → predicted placed

1st case: correctly classified & far toh dhere ↑ push kiya (add kiya)

2nd case: incorrectly classified & dur hai toh strongly pull kiya
$y_i = 0$ (Actual not placed)
$\hat{y}_i \not< 0.5$ predicted placed.

3rd case: incorrectly classified
$y_i = 1$ → placed
$\hat{y}_i = 0.3$ (0.5 → predicted not placed)
and same hai toh dhere pull kiya

4th case: correctly classified
$y_i = 0$ (Actual not placed)
$\hat{y}_i = 0.15$ (predicted not placed)
& dur hai toh dhere ↑ push kiya.



Case I : $w_n = w_o + \eta \times 0.2 \times X_i$ = add kar rhe hai toh niche bhej rhe hai line → hence (I) ne push kiya line

Case II : $w_n = w_o + \eta \times (-0.65) \times X_i$ = subtract → upar bhej rhe hai matlab pull kiya line ko (II) ne

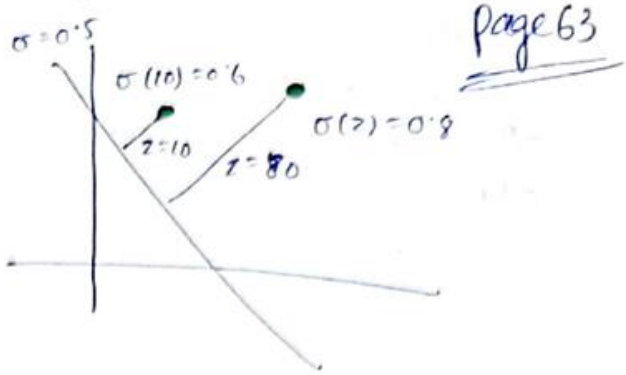
Case III $w_n = w_o + \eta \times (0.3) X_i$ → add → niche bhej : matlab (III) pull kar rha hai line ko apne taraf

Case IV. $w_n = w_o - \eta \times 0.15 \times X_i$ → subtract → line ko upar bhejne matlab line ko push kar rha hai (IV)

∴ Misclassified pt → pull karrahe hai
& correctly classified pts → push kar rahe hai line ko.

what effect

Let's see the distance of pts. from Line have on the magnitude of push & pull:



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 samne 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 correct point jo dur thei wo kam push kar rahe hai

Case III & Case IV: same thing (reverse order) can be seen

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 pata 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 then phir 1000 random pts select honge & on that basis a final Line will be selected but the random pts selected nao 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 soln but it doesn't guarantee that the soln obtained is the best soln since next time we run it, it may give better soln, but again we have no guarantee it is the best one.

So using the ML way:- finding the Loss fn or Error fn

(then Minimize it & find coefficient.

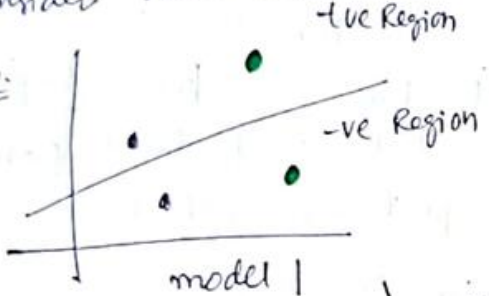
Minimize mein class form ya gradient descent use kar sakte hai.

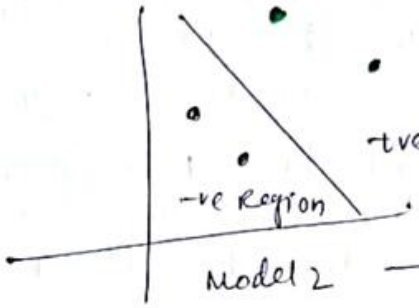
Finding the Loss fn

Maximum Likelihood:-

Consider two models:-

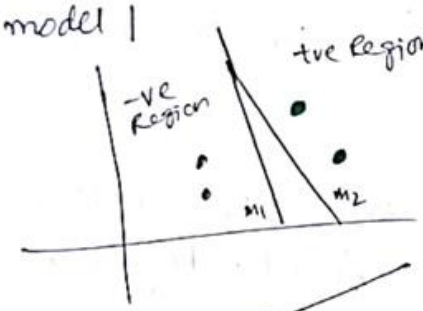
1st one:-





→ of course model 2 is better since data is classified atleast.

Next On:-



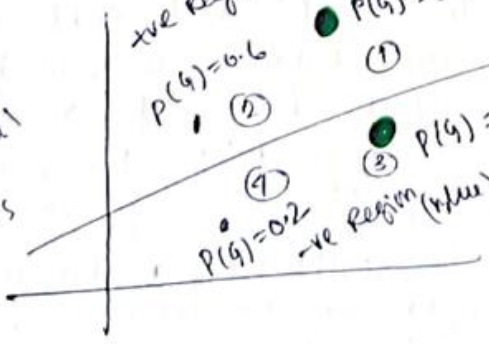
→ Now it asked which model is better M1 or M2 → Difficult to say

Here Loss fn will help & tell which one is better.

& using it we can fetch the best model.

to derive/find Loss fn we use concept called Maximum(green) Likelihood (probability concept).

Consider the models in On1.



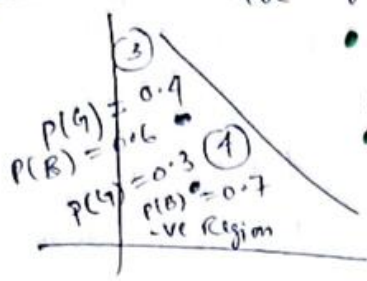
→ To find P(B) for all point

P(B) = 1 − P(G)

↓

Prob(blue) = Prob(not placed)

Acc to model 2.



$\hat{y} = \sigma(z)$ = Prob (of that point to be placed)

$z = \sum w_i x_i$

Nao:: Acc. to maximum likelihood says:
for a model multiply all the prob obtained of the model with the highest prob is better.

But:: AAp sirf wo prob lo jo ek point ka jis color ka wo point hai.

for model 1:: Total prob = $0.7 \times 0.6 \times 0.6 \times 0.3 = 0.176$ → maximum likelihood for model 2
$= P(1) \times P(2) \times P(3) \times P(4)$

for model 1: Total prob = $0.7 \times 0.4 \times 0.4 \times 0.8 = 0.089$
$= P(1) \times P(2) \times P(3) \times P(4)$
→ maximum likelihood for model 1.

∴ Maximum likelihood of model 2 > Max Likelihood of model 1

∴ Model 2 is better than model 1.

Hence humme ek aisa model/line chahiye jiska maximum likelihood max aaye.

Calculation of

The only problem is → product of small Nos.

upar bas 4 prob ko multiply karne pe 0.089 aya. in real world there would be 10000 or more data & hence the product will be 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. → ∴ the sum will be -ve → so to avoid this

for model 1 : $-\log(\text{max likelihood}) = -(\log(0.7) + \log(0.9) + \log(0.9) + \log(0.8))$

for model 2 : $-\log(\text{max likelihood}) = -(\log(0.7) + \log(0.6) + \log(0.6) + \log(0.3))$

↓ This is called cross entropy.

Max likelihood ka log ka -ve. cross entropy

Nao ∵ we multiplied by -ve

∴ the model with minimum cross entropy will be the best

we were maximizing max likelihood & log is ↑ty fn so log (max likelihood) ko bhi maximize karna hoga but − log (max likelihood) = cross entropy ko minimise karna hoga for best model selection.

∵ formula of cross entropy → (note humne woo prob lena hota hai jis color ka woo ball hai).

for placed pts / Green pts $y_i = 1$

& for blue pts / not placed pt $y_i = 0$

$P(\text{placed}) = P(\text{Green}) = \hat{y}_i$

∴ 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.

↓ ∴ formula should consider both.

∴ for any pt the cross entropy value is :-

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

∴ for Green pts. Cross entropy $= -\log(\hat{y}_i)$

& for Blue pt. Cross entropy $= -\log(1-\hat{y}_i)$

∴ Loss fn := Error $= \sum_{i=1}^{n} \left[-y_i \log(\hat{y}_i) - (1-y_i)\left[\log(1-\hat{y}_i)\right] \right]$

∴ to find average Error $= \frac{-1}{n} \sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$

↳ This is called Log loss Error or Binary cross Entropy function.

And Now we need to minimize this fⁿ:.

ie we need to find w_0, w_1, w_2 such that:-

$$L = -\frac{1}{n} \sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)$$ is minimized.

↓ This doesnot have any closed form sol^n (ie direct formula for w_0, w_1 & w_2) & hence we'll need to use Gradient Descent to find the sol^n.

↓ This is what happens in the Scikit Learn Logistic Regression.

Derivative of Sigmoid function:-

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

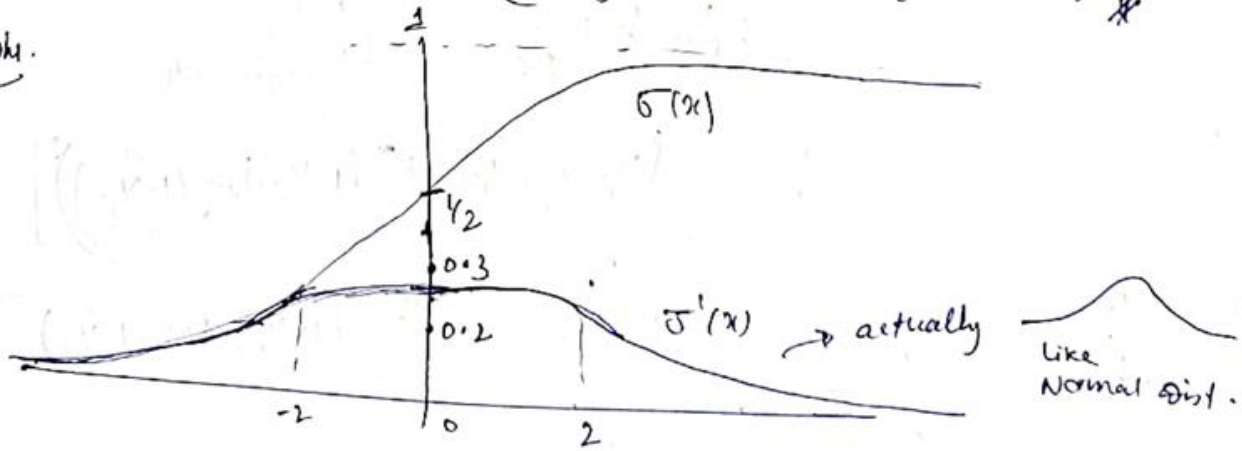
$$\sigma(x) = \frac{1}{1+e^{-x}} \rightarrow \sigma'(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{-\left(0+e^{-x}(-1)\right)}{(1+e^{-x})^2}$$

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}}{(1+e^{-x})}$$

$$\sigma'(x) = \sigma(x)\left[\frac{e^{-x}}{1+e^{-x}}\right] = \sigma(x)\left(1 - \frac{1}{1+e^{-x}}\right)$$

$$\sigma'(x) = \sigma(x) - (\sigma(x))^2 = \sigma(x)\left(1-\sigma(x)\right)$$ ✱

Graph.



Optimizing the Loss-fn using gradient descent

Consider a data set with m rows & n input cols

1	2	3	...	n	output(Y)
x_{11}	x_{12}	x_{13}	...	x_{1n}	y_1
x_{21}	x_{22}	x_{23}	...	x_{2n}	y_2
⋮					
x_{m1}	x_{m2}	x_{m3}	...	x_{mn}	y_m

$\hat{y}$ ↓ σ

$\hat{y}_1 = \sigma(w_0 + w_1 x_{11} + \cdots + w_n x_{1n})$

$\hat{y}_2 = \sigma(w_0 + w_1 x_{21} + \cdots + w_n x_{2n})$

⋮

$\hat{y}_m = \sigma(w_0 + w_1 x_{m1} + \cdots + w_n x_{mn})$

So in Logistic Regression we need to find the coeff- $w_1, w_2, \ldots, w_n$ & intercept- w_0 ↓ Total $(n+1)$ coeff

$\hat{y}_i$ = Sigmoid of $w_0 + w_1 x_{i1} + \cdots + w_n x_{in}$

$= \sigma(w_0 + w_1 x_{i1} + \cdots + w_n x_{in})$

So:- $\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} = \begin{bmatrix} \sigma(w_0 + w_1 x_{11} + \cdots + w_n x_{1n}) \\ \sigma(w_0 + w_1 x_{21} + \cdots + w_n x_{2n}) \\ \vdots \\ \sigma(w_0 + w_1 x_{m1} + \cdots + w_n x_{mn}) \end{bmatrix}$

$\hat{Y}$ ← matrix of predicted values

This sigmoid operation is on all terms ∴ we take it out

$$\hat{Y} = \sigma\left(\begin{bmatrix} w_0 + w_1 x_{11} + \cdots + w_n x_{1n} \\ \vdots \\ w_0 + w_1 x_{m1} + \cdots + w_n x_{mn} \end{bmatrix}\right)$$

$$\hat{Y} = \sigma\left(\begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ 1 & x_{21} & \cdots & x_{2n} \\ \vdots & & & \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}\right) = \sigma(XW)$$

↓ input data with 1st col of all 1's. ↓ coefficients

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

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

Taking this out:-

$$\sum_{i=1}^{m} y_i \log(\hat{y}_i) = y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + \cdots + y_m \log \hat{y}_m$$

$$\sum_{i=1}^{m} y_i \log(\hat{y}_i) = [y_1\ y_2\ \cdots\ y_m] \cdot \begin{bmatrix} \log \hat{y}_1 \\ \log \hat{y}_2 \\ \vdots \\ \log \hat{y}_m \end{bmatrix}$$

$$= [y_1, y_2 \cdots y_m] \cdot \log \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} = y \cdot \log(\hat{y})$$

(Actually transpose hai ka problem hai start se handle karo).

$$= y \cdot \log(\sigma(xw)) \rightsquigarrow \text{matrix form}$$

Silly. $\sum_{i=1}^{m} (1-y_i) \log(1-\hat{y}_i) = (I-y) \log(\sigma(I-xw))$ Log fn in matrix form.

Thus. Loss fn $= L = -\frac{1}{m}\left[y \log(\hat{y}) + (I-y) \log(I-\hat{y}) \right]$ where $\hat{y} = \sigma(xw)$

This doesn't have any closed form eqn ∴ to minimise it: we have to use gradient descent

we initialize $W = [\ldots]$ with random entries

& we decide epochs & then we update the W with formula:- $W_{new} = W_{old} - \eta \frac{\Delta L}{\Delta W}$ → gradient of L cost w.r.t. W.

(η = learning rate)

Finding $\frac{\Delta L}{\Delta W}$:- $\left[\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \ldots \frac{\partial L}{\partial w_n}\right]$:

$$L = -\frac{1}{m}\left[y \log(\hat{y}) + (I-y) \log(I-\hat{y}) \right]$$

$$\frac{dL}{dW} = -\frac{1}{m}$$

$$\frac{d}{dw}(y \log \hat{y}) = y \frac{d}{dw} \log \hat{y} = \frac{y}{\hat{y}} \frac{d}{dw} \hat{y} = \frac{y}{\hat{y}} \frac{d}{dw} \sigma(xw)$$

$$= \frac{y}{\hat{y}} \sigma(xw)[1 - \sigma(xw)] \frac{d}{dw} xw$$

$$= \frac{y}{\hat{y}} \hat{y}(I-\hat{y}) x \quad \because \sigma(xw) = \hat{y}$$

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

$\because \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$ where $\sigma(x) = \frac{1}{1+e^{-x}}$

Silly.

$$\frac{d}{dw}(I-y) \log(I-\hat{y}) = (I-y) \frac{d}{dw} \log(1-\hat{y}) = \frac{I-y}{I-\hat{y}} \left[-\hat{y}(I-\hat{y})\right] x$$

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

Thus $\frac{dL}{dw} = -\frac{1}{m}\left[y(1-\hat{y})x - \hat{y}(1-y)x \right]$

$= -\frac{1}{m}\left[y(1-\hat{y}) - \hat{y}(1-y) \right] x$

$= -\frac{1}{m}\left[y - y\hat{y} - \hat{y} + \hat{y}y \right] x$

$= -\frac{1}{m}\left[y - \hat{y} \right] x$ → Do the derivative of loss fn without the matrix form to find gradient.

Hence... $W_{new} = W_{old} + \frac{\eta}{m}(y-\hat{y})x$.

$W = \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix}_{(n+1)\times 1}$; $X = \begin{bmatrix} 1 & x_{11} & x_{12} \cdots x_{1n} \\ \vdots & & \\ 1 & x_{m1} & x_{m2} \cdots x_{mn} \end{bmatrix}_{(m,\, n+1)}$; $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}_{m\times 1}$

$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_m \end{bmatrix}_{m\times 1}$

Jupyter Notebook.

Problem

We learnt Regression metrics in Regression, to find which model performs better on a Dataset.

Similarly classification problems have classification metrics which guide us ki kon si classification model kitna accha perform kar raha hai.

There are lot of classification metrics :-

confusion matrix, precision, Accuracy metric, F1-score, ROC, AUC etc.

Accuracy

1000 students data → 800 train data, 200 test data

we use two classification Algorithm :-

Consider: cgpa | IQ | placement

placed or not :-

Actual label	Log pred.	Dec. Tree pred
1	1	1
0	0	0
0	0	0
0	0	0
1	1	1
0	0	0
0	0	0
0	1	1
1	1	1

Logistic Reg → Train → Then Test

Decision Tree → Train → Then 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 galti ki ya nahi & same for Decision Tree & then find kisne kam galti ki & that model is better.

$$\text{Accuracy score} = \frac{\#\text{q. correct pred}}{\text{Total pred}}$$

$\therefore$ acc. score for Log. Reg $= \frac{8}{10} = 0.8 = 80\%$

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, Versinica, Veriscal

0 1 2

Actual Label	Log. Reg. pred	Dec. Tree pred
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	1

→ in this case

$$\text{accuracy score} = \frac{\#\text{q. correct pred}}{\text{Total pred}}$$

$\therefore$ acc. score of log reg $= \frac{10}{10} = 100\%$.

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 bhi model ko deploy kar sakte hai & 99% pe bhi nahi kar sakte.

Means: food order karega ya nahi wale model ka 80% accuracy hai toh bhi deploy kar sakte hai since agar wrong predict kiya toh kuch khas loss nahi hoga

par :- heart disease hai ya nahi wale model ko 99% pe

bhi deploy nahi kar sakte :- galti kiya toh patient marega → galti ka stake bahot high hai.

The problem with accuracy score:

→ It gives a number matlab say acc. score = 0.9 matlab 10% galti par rha hai model par acc. score ye nahi bata rha ki galti ka type kya hai.

say binary classification hai :-

Galti types

Actual	predicted
1	0

&

Actual	predicted
0	1

Type 2 Error (False Negative) ← Placement hua that but model ne not placed predict kiya

Placement nahi hua that but model ne placed predict kiya. → Type 1 Error. (False positive).

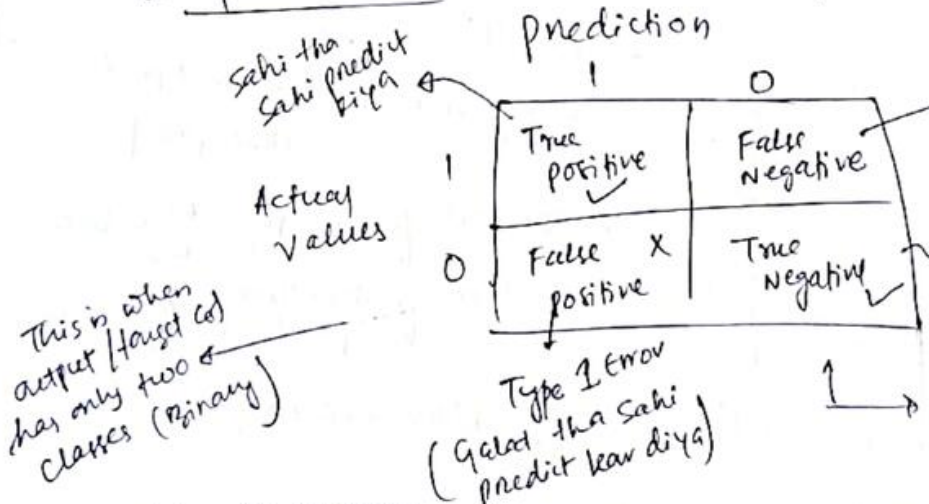
Acc score ye nahi batayega ki. konsa Error kitne baar hua in that 10% galti.

↓ To handle this:- we use

Confusion Matrix.

first see jupyter Notebook how to Calculate Confusion Matrix using Scikit Learn.

Confusion Matrix: (scikit Learn Nomenclature)



↳ issse hum accuracy score bhi nikal sakte hai

Ex: Heart disease.

Actual \ Pred.	1	0
1	26	6
0	0	29

→ 26 logo ko heart disease tha & model ne bhi hai predict kiya

→ 29 logo ko nahi tha & model ne bhi nahi hai predict kiya

→ 6 log ko heart disease tha & model ne nahi hai predict kiya.

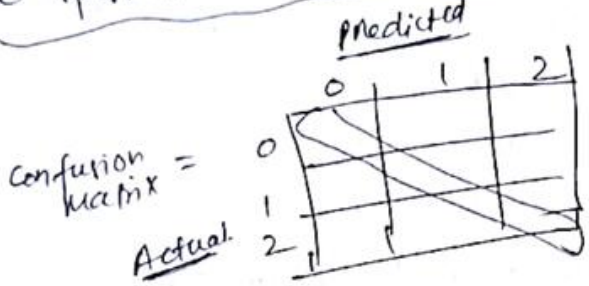
accuracy score = $\frac{\#\text{ of correct pred}}{\text{total pred}}$

$= \frac{\text{True pos + True Neg}}{\text{True pos + False pos + True Neg + False Neg.}}$

* False positive → is called Type 1 Error
& False Negative → is called Type 2 Error.

Confusion Matrix for multiclass classification problem

↓ output mei more than 2 categories hai



Sily the Accuracy score = $\frac{\#\text{ of Correct pred}}{\text{total pred.}}$

are the Right/correct Prediction.

in the MNIST Data set → there a 10 classes (0-9)
& Hence the confusion Matrix will be 10 x 10 order.

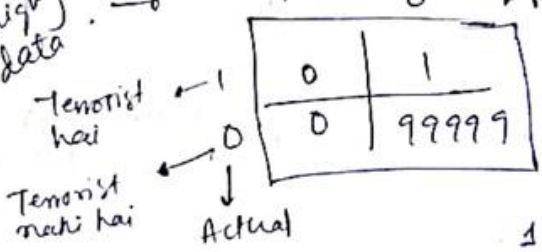
When accuracy is misleading?

↳ Imbalanced Dataset → ie. the in categories are not equal/almost equal.

Ex: On an Airport passenger hai ya Terrorist. This will be highly imbalanced data.

↓ in such case accuracy gives bad result.

Total 100000 passenger



→ 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:. (Email spam classifier)

spam: 1 ; not spam: 0

predicted

Actual	Sent to spam	Not sent to spam
Spam	100	170 (False neg.)
not spam	30 (False pot)	700

Model 1

predicted

Actual	Sent to spam	Not sent to spam
Spam	100	190 (False neg.)
Not spam	10 (False pot)	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
& 190 spam are not sent to spam.

Here: not spam messages are important (as they may contain something imp)

∴ if more non spam are sent to spam → it is not good.

Hence model with less False pot mistake is good.

Hence model 2 is good comparatively.

↳ isi ko precision bola jata hai.

↳ It is what Proportion of predicted positives is truly positive :-

∴ precision = True positive / True pot + False pot = TP / TP + FP

Higher the precision better the model.

in model 1 precision = $\frac{100}{130} = \frac{10}{13}$

in model 2 precision = $\frac{100}{110} = \frac{10}{11}$ → This is greater, hence in this case model 2 is better than model 1.

To understand Recall:.

Cancer Detection:

	Detected cancer	Not detected	
Has Cancer	1000	200	→ False Neg
NO Cancel	800	8000	

↓ model1 false pot

	Act: cancer	Not act	
Has cancer	1000	500	→ False Neg.
NO cancer	500	8000	

↓ model2. False pot

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 ∴ model 1 is Better. ↓ This is called Recall.

Defn: Recall :- What proportional of Actual positive is correctly classified (is predicted positive) ?

∴ for model 1 :- Recall = $\frac{1000}{1200} = \frac{10}{12}$

for model 2 :- Recall = $\frac{1000}{1500} = \frac{10}{15}$

formula: Recall = $\frac{\text{True positive}}{\text{True positive + False Negatives.}} = \frac{TP}{TP + FN}$

Lesser the Recall → Better the Model.

Hence: if Type 1 Error is Dangerous → then High precision model is prefered/better

& if Type 2 Error is Dangerous → 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 → cat ko dog bol diya → Which is dangerous we don't know
Type 2 Error → dog ko cat bol diya

↓ in this case both precision & Recall has to be handled — which is difficult as there is a tradeoff b/w them (they are kind of inv. proportional).

↓ In this case we use F1_Score metric.

↓

It is a combination of precision & Recall (actually a Harmonic Mean of both).

$$F1\text{-}score = \frac{2PR}{P+R} = \frac{2 \text{ Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1_Score → [0, 1]
worst ✓ ↓ best

A higher F1_score is generally better.

↓

It is still able to give true model performance when dataset is imbalanced.

Jupyter NoteBook.

Harmonic mean ek aisa mean hai jo hamesha lower value ke side inclined rehta hai

↓

Ex: P = 0, R = 100

$AV = \frac{P+R}{2} = 50$

F1_score = 0

↓

a, b a > b

Harmonic mean = $\frac{1/a + 1/b}{2}$

$\frac{1}{b} > \frac{1}{a}$ ∴ Harmonic mean shifts towards $\frac{1}{b}$

Precision, Recall & F1-score for multi class classification.

Consider Image classifier

Confusion matrix:

Actual \ Predicted	Dog	Cat	Rabbit	Actual Total
Dog	25	5	10	40 → Actual no. of dogs
Cat	0	30	4	34 → Actual no. of cats
Rabbit	4	10	20	34 → Actual no. of Rabbits
Predicted Total	29	45	34	

29 ↓ Predicted no. of dogs
45 ↓ Predicted no. of cats
34 ↓ Predicted no. of Rabbits.

Calculation for precision = Actual for / total pred for

→ precision for dog class = # of Actual dogs / total # of predicted dogs = 25/34 = 0.88

(1st col)

2nd col → " " Cat class = # of Actual Cats / total # of predicted cats = 30/45 = 0.66

3rd col → " " Rabbit class = # of Actual Rabbits / total # of predicted Rabbits = 20/34 = 0.58

Combined precision of the model:

- Macro precision (when classes are balanced) = Avg. of all the precision (dog, cat precision) / 3 classes = (0.88 + 0.66 + 0.58)/3 = 0.70
- Weighted precision (when classes/categories are imbalanced): Sum of (class weight × class precision) = (0.88 × 40/100) + (0.66 × 30/100) + (0.58 × 34/100); class weight = Actual for / total

Calculation for Recall = predicted for / total Actual for = 0.71

for row1 → Recall of dog class = 25/40 = 0.62

for row2 → Recall of cat class = 30/34 = 0.88

for row3 → Recall of Rabbit class = 20/34 = 0.58

Combined Recall of the model

- Macro Recall (when class/categories are balanced) = Avg. of all Recalls = (0.62 + 0.88 + 0.58)/3
- Weighted Recall (when): Sum of (class weight × class Recall) = (0.62 × 27/100) + (0.88 × 45/100) + (0.58 × 34/100); predicted for / total

Similarly Calculation of F1-score

F1-score of dog class = $\frac{2 P_D R_D}{P_D + R_D}$

" " cat class = $\frac{2 P_C R_C}{P_C + R_C}$

" " rabbit class = $\frac{2 P_R R_R}{P_R + R_C}$

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 :. (binary classification)

What if there are more than two :-

↓ Then we use :-

Softmax Regression or Multiclass/Multinomial Logistic Regression

Consider Dataset :-

Cgpa	IQ	placement
7.1	7.1	0
8.5	85	1
9.5	97	2
⋮	⋮	⋮

The classes here :- Placed 1, Not placed 2, Opt out 3

we need to a Logistic Reg. classifier that predicts output given the necessary input.

Logistic Regression (for binary output) is a special case of Softmax Regression.

What is Softmax function :-

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$; K = # of classes/categories in the output col.

placed.

$\sigma(z)_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$; $\sigma(z)_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$; $\sigma(z)_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$

↓ prob of placed class

↓ not placed → prob of not placed class

↓ prob of opt out class.

Note :- $0 < \sigma < 1$

$\sum_{i=1}^{K} \sigma(z)_i = 1$ → K = no. of class/categories in output.

Softmax for two classes for output is Sigmoid fun.

Training intuition of softmax Reg. (Just an example. Ise Exactly scikit Learn mei nahi hota hai)

cgpa	IQ	placement
:	:	0
:	:	1
:	:	2

{ Y = 0 / N = 1 } → 0 → 1

→ one hot encoding of output col

class1 not placed, class2 placed, class3 opt out.

cgpa	IQ	0	1	2
:	:	1	0	0
:	:	0	1	0
:	:	0	0	1

→ Now we have 3 cols.

→ As there are three output cols & they are Binary classified.

Now we split it into three (No. of categories) Datasets.

D_1 class 1

cgpa	IQ	notplaced = 0
:	:	:

D_2

cgpa	IQ	placed = 1 : class 2
:	:	:

D_3

cgpa	IQ	optout : 3 : class 3
:	:	:

Now we train this on three Logistic Reg model.

Model 1 → This will return 3 coeff (∵ 2 input Data) $w_1^{(0)}, w_2^{(0)}, w_0^{(0)}$

Model 2 → 3 coeff $w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$

Model 3 → 3 coeff $w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$

Actually scikit Learn mei training thode diffient hota hai. (Loss fun & Gradient descent use ho rha hai).

Now training is done :- Let's focus on predictions :-

M1

$w_1^{(0)}, w_2^{(0)}, w_0^{(0)}$

↓ given a query point this model predicts placement hua ya nahi

'placement hua' : pot class

0 : nahi :- Baaki do cases opt out ya placed.

M2

$w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$

↓ given a query point it predicts

1 : placement hua (pot class)

0 : baaki dono class not placement hua ya opt out

M3

$w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$

↓ given a query point it predicts :-

1 : opt out (pot class)

0 :- Baaki dono class placed ya not placed.

Given a student :- cgpa = 7, IQ = 70 → To predict (placed, not placed yo opt out

not placed
M1 :-

$w_1^{(0)}, w_2^{(0)}, w_0^{(0)}$

$z_1 = 7w_1^{(0)} + 70w_2^{(0)} + w_0^{(0)}$

70 → Pot side → means not placed

<0 → -ve side → means either placed or opt out

Now we ↓ apply the

$$\sigma(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

↓ Prob of Not placed σ(not placed) = 0.35 (say)

placed
M2.

$w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$

$z_2 = 7w_1^{(1)} + 70w_2^{(1)} + w_0^{(1)}$

→ Sly in these two cases also.

Softmax fn :- ↓

$$\sigma(z_2) = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

↓ σ(placed) ↓ prob of placed = say 0.90

opted out
M3.

$w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$

$z_3 = 7w_1^{(2)} + 70w_2^{(2)} + w_0^{(2)}$.

↓

$$\sigma(z_3) = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

σ(opt out) = prob opt out = 0.25 (say)

Hence going by this logic prob of placed is highest
∴ the student will get placed.

↳ # 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

↓ so we use diff approach

we change the Loss fn of the Logistic Regression & using that changed loss fn → using a single model we can train the above three dataset & get all the coefficients for each model.
(9 coefficients)

↓

This is implemented in scikit Learn

And for this Reason this is not implemented in Scikit Learn

Say we choose degree 2.

Loss function:-

Logistic Regression loss fn (with m rows/students) $= -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$

Softmax Reg / Multinomial Logistic Regression loss fn =

$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{K} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right)$$

- m = # of rows
- K = # of categories in output
- for rows → how many rows we have
- # of classes ka summation hai.

Ex:

x_1	x_2	y
x_{11}	x_{12}	1
x_{21}	x_{22}	2
x_{31}	x_{32}	3

→ one hot encoding.

x_1	x_2	$y_{K=1}$	$y_{K=2}$	$y_{K=3}$
x_{11}	x_{12}	1	0	0
x_{21}	x_{22}	0	1	0
x_{31}	x_{32}	0	0	1

Loss fn $= -\frac{1}{3}\Big[y_1^{(1)} \log\left(\hat{y}_1^{(1)}\right) + y_2^{(1)} \log\left(\hat{y}_2^{(1)}\right) + y_3^{(1)} \log \hat{y}_3^{(1)}$ (model 1 for class 1, $i=1$, $K=1,2,3$)

$+ y_1^{(2)} \log\left(\hat{y}_1^{(2)}\right) + y_2^{(2)} \log\left(\hat{y}_2^{(2)}\right) + y_3^{(2)} \left(\log \hat{y}_3^{(2)}\right)$ (model 2 for class 2, $i=2$, $K=1,2,3$)

$+ y_1^{(3)} \log\left(\hat{y}_1^{(3)}\right) + y_2^{(3)} \log\left(\hat{y}_2^{(3)}\right) + y_3^{(3)} \left(\log \hat{y}_3^{(3)}\right)\Big]$ (model 3 class 3, $i=3$, $K=1,2,3$)

($m=3$)

for the above Data :-

$$\text{Loss fn} = -\frac{1}{3}\Big[y_1^{(1)} \log\left(\hat{y}_1^{(1)}\right) + 0 + 0 + 0 + y_2^{(2)} \log\left(\hat{y}_2^{(2)}\right) + 0 + 0 + 0 + y_3^{(3)} \log\left(\hat{y}_3^{(3)}\right)\Big]$$

$$= -\frac{1}{3}\Big[y_1^{(1)} \log\left(\hat{y}_1^{(1)}\right) + y_2^{(2)} \log\left(\hat{y}_2^{(2)}\right) + y_3^{(3)} \log\left(\hat{y}_3^{(3)}\right)\Big]$$

Now to find: softmax for $\hat{y}_1^{(1)}$, $\hat{y}_2^{(2)}$, $\hat{y}_3^{(3)}$:-

$$\hat{y}_1^{(1)} = \sigma\left(w_1^{(1)} x_{11} + w_2^{(1)} x_{12} + w_0^{(1)}\right)$$ → where: $w_1^{(1)}, w_2^{(1)}, w_0^{(1)}$ are weights / coeff wrt col $y_{K=1}$

$$\hat{y}_2^{(2)} = \sigma\left(w_1^{(2)} x_{21} + w_2^{(2)} x_{22} + w_0^{(2)}\right)$$ → $w_1^{(2)}, w_2^{(2)}, w_0^{(2)}$ are coeff wrt col $y_{K=2}$

$$\hat{y}_3^{(3)} = \sigma\left(w_1^{(3)} x_{31} + w_2^{(3)} x_{32} + w_0^{(3)}\right)$$ → $w_1^{(3)}, w_2^{(3)}, w_0^{(3)}$ are coeff wrt col $y_{K=3}$.

So the coeffs are :- $\begin{bmatrix} w_1^{(1)} & w_2^{(1)} & w_0^{(1)} \\ w_1^{(2)} & w_2^{(2)} & w_0^{(2)} \\ w_1^{(3)} & w_2^{(3)} & w_0^{(3)} \end{bmatrix}$ → as in the loss fun

Hence to find the gradient of loss fun wrt them in gradient descent we need to calculate :

$$\frac{\partial L}{\partial w_1^{(1)}}, \frac{\partial L}{\partial w_2^{(1)}}, \frac{\partial L}{\partial w_0^{(1)}}, \frac{\partial L}{\partial w_1^{(2)}}, \frac{\partial L}{\partial w_2^{(2)}}, \frac{\partial L}{\partial w_0^{(2)}}, \frac{\partial L}{\partial w_1^{(3)}}, \frac{\partial L}{\partial w_2^{(3)}}, \frac{\partial L}{\partial w_0^{(3)}}.$$

∴ In gradient descent → we initialize with random values of the above matrix & then in epochs we update for each of 9 coefficients.

$$w_1^{(1)} = w_1^{(1)} - \eta \frac{\partial L}{\partial w_1^{(1)}}$$

$$w_2^{(1)} = w_2^{(1)} - \eta \frac{\partial L}{\partial w_2^{(1)}}$$

⋮

and so on.

jupyter Notebook.

Polynomial features in Logistic Regression

Non-Linear Logistic Regression

we know Logistic Regression works when the data is Linearly seperable.

But if the data is non Linearly seperable 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.

x_1	x_2	y
⋮	⋮	⋮

→

x_1^0	x_1^1	x_1^2	x_2^0	x_2^1	x_2^2	y

& Now we use Logistic Regression

Jupyter Notebook.

Hyperparameters in Logistic Regression

- Penalty: 'l1', 'l2', 'elasticnet', 'none' :→ for Regularization
  default 'l2'
- tol (default = $1e^{-4}$) → tolerance → it is the stopping criteria for gradient descent.
- C (default = 1.0) → it is the inverse of regularization strength
  ie. $C = 1/\lambda$ → smaller the value of C stronger the Regularization.
- fit_intercept (default = True) → w_0 wala value chahiye ya nahi.
- class_weight: (default = None); 'balanced' is used if the data has unbalanced classes in output (like 95% one class & only 5% another class)
- solver: {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default = 'lbfgs'.
  ↳ we made our own Logistic Reg. class using gradient descent sirf 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
  ↳ used when solver == 'sag', 'saga' or 'liblinear' to shuffle data.
- max_iter: int; default = 100
  ↳ # 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 → for every class a Logistic Reg model is made & coeff are found like we discussed
  multinomial → Loss fn is minimized & coeff are found.

- warm_start :- default : False :- ← ek baar Curve se training hoga.
  When set to True, jis point pe Training stop kar the ho, use load first training start karega then jis point pe stop kiya tha wahi se start hoga.

- n_jobs : int, default : None.
  No. of cpu cores used when parallelizing over classes if multi_class = 'ovr'
  ↳ means using all processors → multi processing.

- l1_ratio :- ($0 \leq l1\_ratio \leq 1$) → only used if penalty = 'elasticnet'.
  l1_ratio = 0 → means penalty = 'l2'
  l1_ratio = 1 → mean penalty = 'l1'.

DECISION TREES

Consider the data:

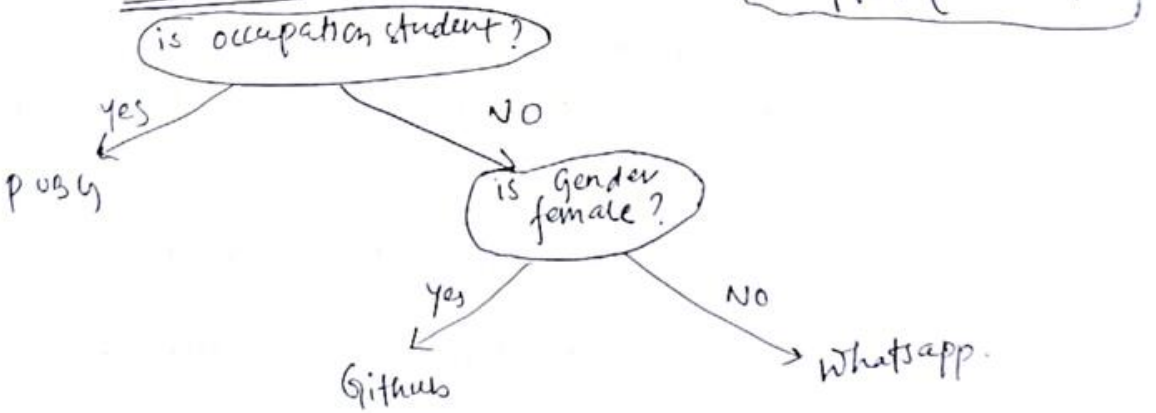
Gender	occupation	Suggestion
F	student	PUBG
F	programmer	Github
M	"	whatsapp
F	"	Github
M	student	pubg
M	"	"

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).

```
if occupation == 'student'
    print (PUBG)
else if gender == 'female'
    print (Github)
Else
    print (whatsapp)
```

The tree :-

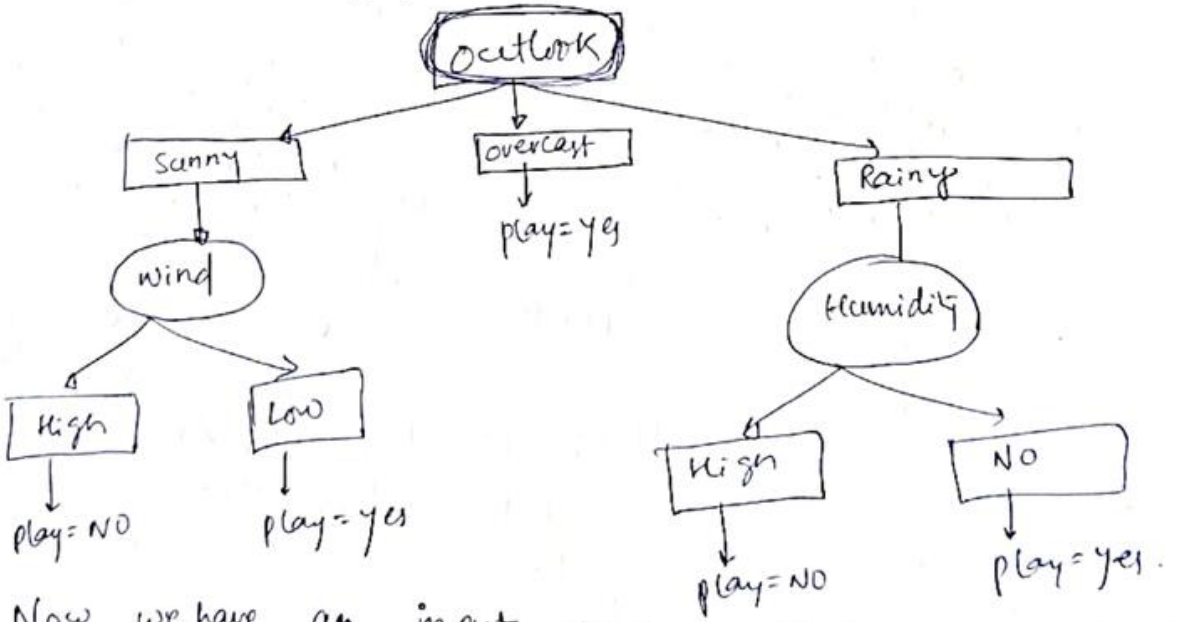


Consider the Data:-

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]
outlook Temp Humidity wind

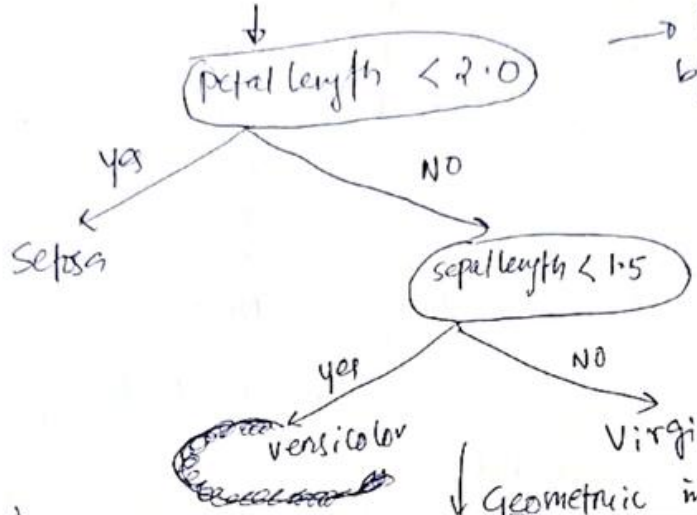
we go to Rainy & then check Humidity instead of Temp & hence predict play = No.

Here, we only checked outlook & Humidity even though there were other conds, since our decision tree is made that way.

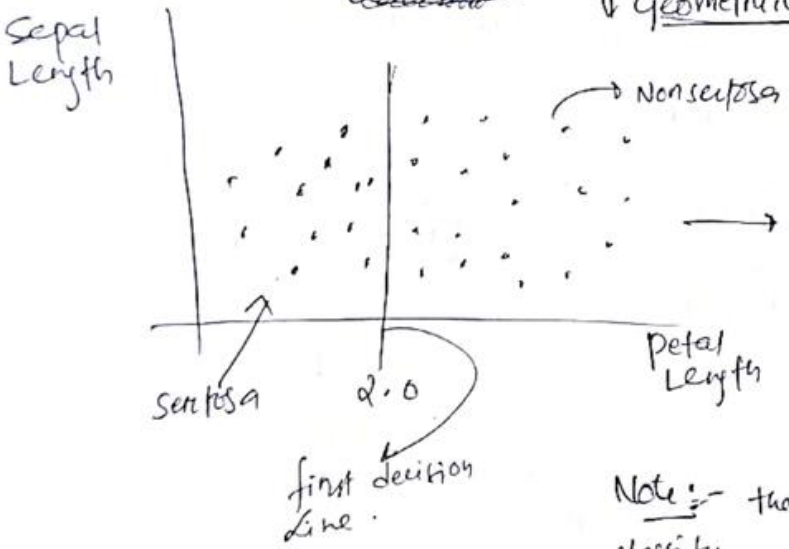
What if we have Numerical Data?

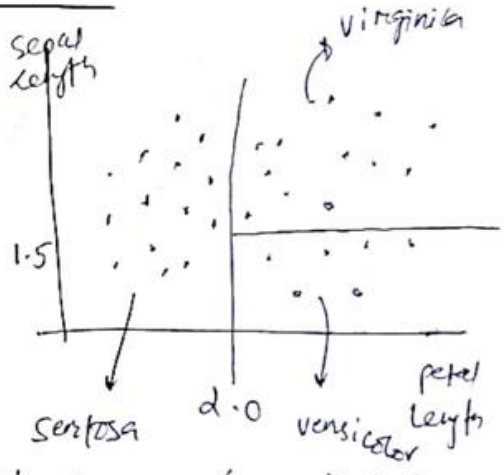
petal-length	sepal length	Type
1.34	0.34	Setosa
3.45	1.45	Versicolor
1.69	0.98	Setosa
2.56	1.79	Virginica
3.00	1.13	Versicolor
1.3	0.68	Setosa.

→ 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.

Note:- The lines we make are parallel to coordinate axes.

↓ In higher Dimensions the Lines will be planes & hyperplanes.

underfitting.

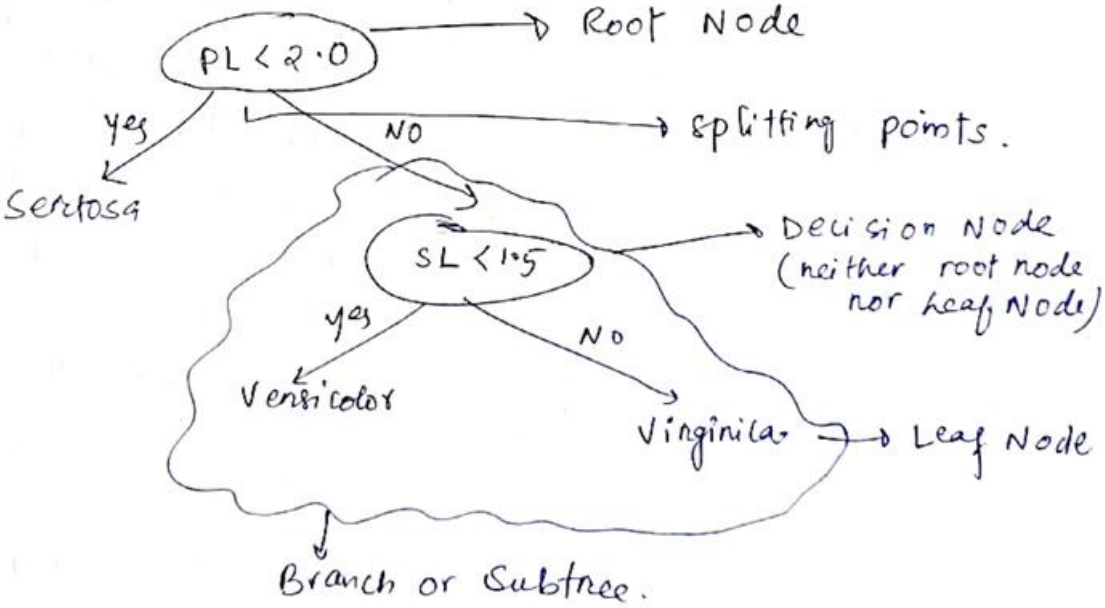
Pseudo Code :- for decision Tree with

- Begin your training dataset, which should have some feature variables and classification or regression output.
- Determine the "best feature/col" in the dataset to split the data on (we will see how we define best feature)
- Split the data into subsets that contains the correct values for this best feature. This splitting basically defines a node on a tree i.e. each node, is a splitting point based on a certain feature from our data
- Now we chose the next best feature
- Recursively Generate new tree nodes by using the subset of data created from step 3.

Conclusion:

- programatically speaking, Decision trees are nothing but a giant structure of nested if else condn.
- mathematically speaking, Decision trees uses hyperplanes which run parallely to any one of the axes to cut your coordinate system into hyper cuboids.

Termindogy.



Some Questions:-

- How to decide which column should be considered as root node? (ie. which col/feature is the best feature).
- How to select subsequent decision nodes? (Next best features/cols)
- How to decide the splitting criteria in case of Numerical columns?

Advantages

- Intuitive & easy to understand
- Minimal data preparation is Reqd.

The cost of using the tree for inference is logarithmic in the number of data points used to train the tree :. Time complexity is $O(\log(\#\text{ of data pts in } X\_train))$

Disadvantages

- There is a great chance of overfitting
- prone to errors for imbalanced dataset

* 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 problems as well, hence named CART.

* There is a Game called Akinator :- where we think of a character & the Akinator will ask us questions :- Is he asian? Is he Indian? Is he an actor? does he 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)

underfitting :-

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? → It is related to uncertainity

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 :-

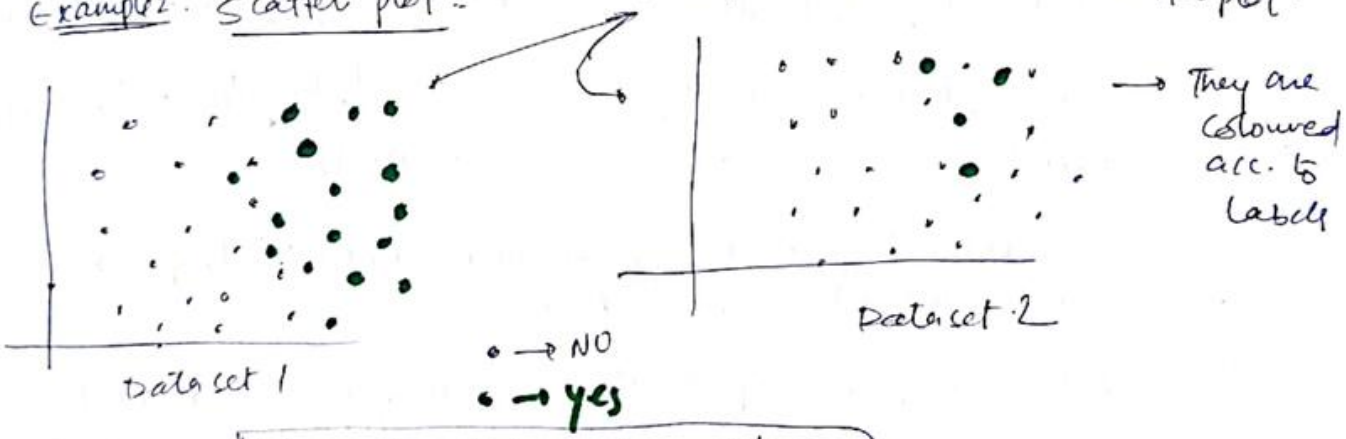
Water has three states :-

molecules are tightly packed — the molecules are loosely packed

ice, water, vapour → ice is solid, water is liquid, vapour is gas → the molecules are most loosely packed.

Hence gas (vapour) has the highest Entropy of the three.

Example: Scatter plot :-



More knowledge Less Entropy.

∴ The dataset jispe 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. i.e 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.

$P_Y = 2/5$, $P_N = 3/5$

$$\therefore E(D_1) = -P_Y\log_2(P_Y) - P_n\log_2(P_n)$$

$$E(D_1) = -2/5\log_2(2/5) - 3/5\log_2(3/5)$$

$$E(D_1) = 0.97$$

D2

Salary	Age	Purchase
34000	31	No
15000	25	NO
69000	57	Yes
25000	21	NO
32000	28	NO

It also has two classes yes & No

$P_Y = 1/5$; $P_N = 4/5$

$$E(D_2) = -P_Y\log_2(P_Y) - P_n\log_2(P_n)$$

$$= -1/5\log_2(1/5) - 4/5\log_2(4/5)$$

$$= 0.72$$

> It has higher entropy so more uncertainity & hence less certainity than D2.

D3. all entries in output col = No (Then we will be most certain & hence it should have least entropy)

$$E(D_3) = -P_{yes}\log_2(P_{yes}) - P_N\log_2(P_N)$$

$$= -0\log_2(0) - 1\log_2(1)$$

$$= 0$$

* Calculating Entropy of a 3 class problem :-

Salary	Age	Purchase
20000	21	yes
10000	45	NO
60000	27	yes
15000	31	NO
30000	30	maybe
12000	18	NO
40000	40	maybe
20000	20	maybe

→ classes : yes, NO, Maybe

$E(\text{dataset}) = -P_y \log_2(P_y) - P_N \log_2(P_N) - P_m \log_2(P_m)$

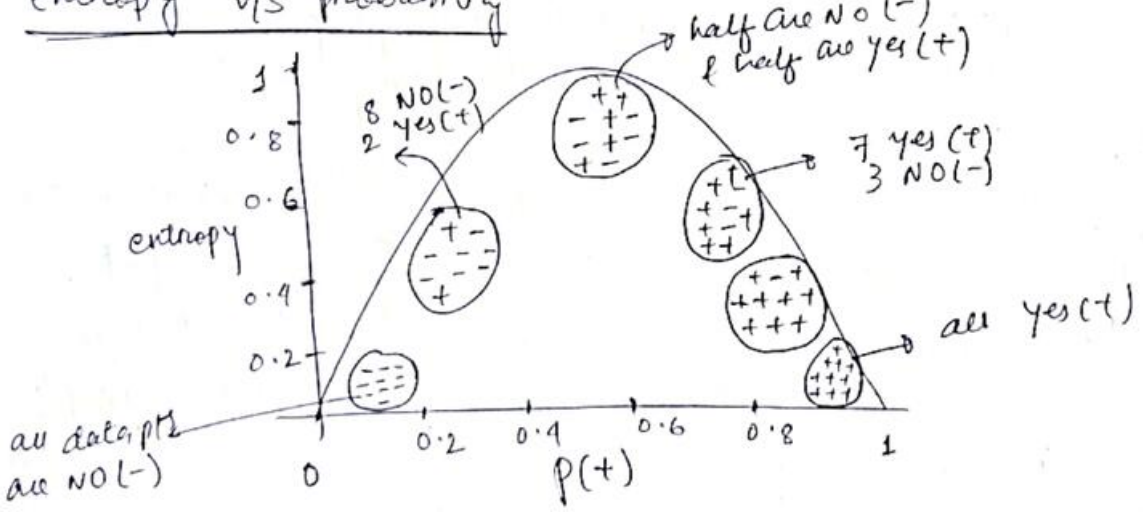
$P_y = 2/8, \quad P_n = 3/8, \quad P_m = 3/8$

$\therefore E(\text{Dataset}) = 1.56$

Observations:

- more the uncertainity 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 :: we only want to know ki ki's dataset mei Entropy/uncertainity jyada hai.

Entropy V/S probability



Hence Entropy is highest if $P(\text{class}) = 0.5$ for 2 class

Or. if all the class has equal probabilities then the entropy is the highest (which is obvious also → :: then we are least certain about a given datapoint that it is of a particular class.

Entropy for Continuous Variables :-

↳ output col is Numerical

Dataset 1.

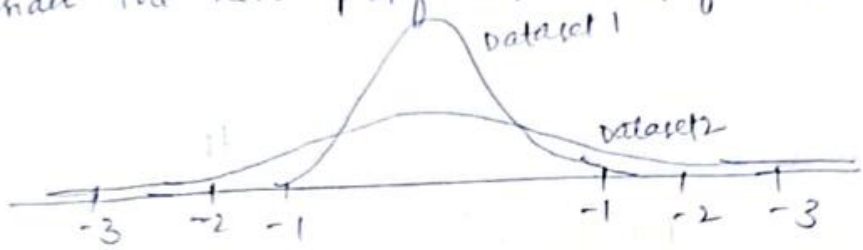
Area	Builtin	price
1200	1999	3.5
1800	2011	5.6
1400	2000	7.3
⋮	⋮	⋮

Dataset 2.

Area	Builtin	price
2200	1989	4.6
880	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.

Dataset 1 ke jyadatar points $[-1, 1]$ ke hai & hence we have more knowledge about Dataset 1 → kam area mei jyada points

Dataset 2 ke jyadatar points $[-3, 3]$ ke hai & hence we have less knowledge about Dataset 2 ∴ Datapoint spread hai

Hence:- Dataset 1 has lesser Entropy compared to Dataset 2.

Hence for cont's variable :- plot the kde (histogram → smooth)

higher the peak of kde → Less the Entropy &

smaller the peak of kde → higher the Entropy.

Information Gain:

Information Gain is a metric used to train Decision Trees. Specifically, this metric measures the quality of a split.

The information gain is based on the decrease in entropy after a data-set is split on an attribute / col / feature

Constructing a decision tree is all about finding the attribute / col / feature that returns the highest information Gain.

formula for Information Gain for a particular col:

Info Gain = E (parent) − {weighted Average} * E (Children)

E (parent) → Complete dataset before split.

{weighted Average} → Larger split the weight average of Entropy of each the Children

underfitting :-

Consider the Dataset with cols: outlook, Temp, Humidity, wind, play.
Let's calculate the info gain for the outlook col

Step1: Entropy of parent

$$E(P) = -P_Y \log_2(P_Y) - P_N \log_2(P_N) = 9/14 \log_2(9/14) - 5/14 \log_2(5/14)$$
$$= 0.94$$

Step2:- We'll divide the dataset based on the outlook col :- (Group by outlook col)

outlook

Sunny

Day	outlook	Humidity	wind	play
1	Sunny	High	weak	NO
2	"	High	Strong	NO
8	"	High	weak	NO
9	"	Normal	weak	Yes
11	"	Normal	Strong	Yes

Overcast

D	O	H	W	P
3	Overcast	High	weak	Yes
7	"	Normal	Strong	Yes
12	"	High	Strong	Yes
13	"	Normal	weak	Yes

Rain

D	O	H	W	P
4	Rain	High	weak	Yes
5	"	Normal	"	Yes
6	"	"	Strong	NO
10	"	"	weak	Yes
14	"	High	Strong	NO

Calculate the Entropy of all these children Datasets.

E(Sunny) = 0.97
$P_Y = 2/5$; $P_N = 3/5$

E(overcast) = 0
$P_Y = 1$; $P_N = 0$

E(Rain) = 0.97
$P_Y = 3/5$, $P_N = 2/5$

Step3: Calculated weighted Entropy of children

weighted entropy = Prob(Sunny) x E(Sunny) + P(overcast) x E(overcast) + P(Rain) x E(Rain)

$$= 5/14 \times 0.97 + 4/14 \times 0 + 5/14 \times 0.97$$

W.E.(Children) = 0.69

Note:- Overcast is a leaf node as it's Entropy is zero(0) → this data will have no more splitting.

Step4:- Calculate Information Gain:-

Info Gain for outlook Column = E(parent) − WE(children) = 0.97 − 0.69 = 0.28

So the info gain (or decrease in Entropy / impurity) when you split this data on the basis of outlook condition / col is 0.28

Step5:- Calculate the Info Gain for all the cols in the similar manner & whichever column has the highest information Gain (max decrease in entropy) the algorithm will select that col to split the data.

Step6:- Find info gain Recursively.

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

In sklearn DecisionTree Classifier there is a parameter called 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 wala two Datasets :-

Lesser the GI more pure data is more knowledge we have - more confidently we can predict & lower the Entropy

for Dataset 1 :-

$$GI = 1 - (P_Y^2 + P_N^2) = 1 - (4/5 + 3/5) = 0.48$$

for Dataset 2 :-

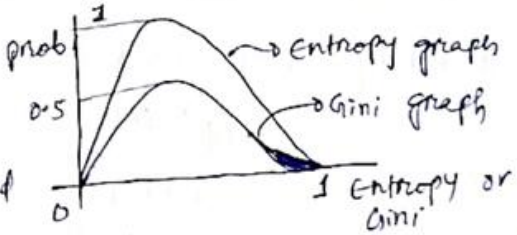
$$GI = 1 - (P_Y^2 + P_N^2) = 1 - (1/25 + 16/25) = 0.32$$

Hence Dataset 2 is more pure than Dataset 1.

Note :- jo kaam hum Info Gain ke liye Entropy ke through karte the wohi same kaam hum Gini Impurity ke through bhi kar sakte hai. The formula will be Info Gain = GI(parent) − weighted GI(children)

if :- $P_Y = 1$, $P_N = 0$ → $E(D) = 0$, $GI(D) = 0$ & if $P_Y = 0.5$, $P_N = 0.5$ → $E(D) = 1$, $GI(D) = 0.5$

∴ 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 Entropy case compared to Gini. (Hence always do hyperparameter tuning for both Gini & Entropy to get the best result).

HANDLING Numerical input col.

→ Here if we Groupby User Rating data['user_rating'].nunique() = 12 → all unique values.

Sl No.	UserRating	Downloaded
1	3.5	Y
2	4.6	N
3	2.2	Y
4	1.6	N
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

Now splitting on 12 data we will get lot of (12) subtrees & hence calculation will be difficult

Solution :-

Step 1 :- Sort the data on the basis of the numerical col. (Here user_rating) → 1.6, 1.9, 2.2, ...

Step 2 :- Split the entire data on the basis of every value of user_rating

rating ≤ 1.6 or rating > 1.6 again.

4	1.6	Y

(one data | rest all data)

rating ≤ 1.9 or rating > 1.9 & soon for all the rows after sorting.

4	1.6	Y
12	1.9	Y

(two data | rest all data)

underfitting :-

So we have :- for every particular value of user rating we split the data set.

step3. step1.

$D \to f > V_1 \to D_1 \to E(D_1) = E_1$, $D_2 \to E(D_2) = E_2$ → weighted entropy $WE1$ → infogain = $IG1$ = E(Parent) − $WE1$

$D \to f > V_2 \to D_1 \to E(D_1) = E_1$, $D_2 \to E(D_2) = E_2$ → weighted entropy $WE2$ → $IG2$ = E(Parent) − $WE2$

$D \to f > V_{12} \to D_1 \to E(D_1) = E_1$, $D_2 \to E(D_2) = E_2$ → weighted entropy WE_{12} → $IG12$ = E(Parent) − $WE12$.

Where: D → dataset
f is feature/col user-rating
& V_i's are the values of the various rows of User-Rating.

After this :-

step5 : max {IG1, IG2, . . . , IG12}

say IG3 is the max then $f > V_2$ will be the splitting criteria for col user-rating.

step6 : 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 donot do this on the test dataset or on deployment level when a new query point is given. ∴ The train time complexity is still higher but the test time complexity is log(n) (Tree based).

Hence :- Decision Tree is more apt for categorical Datas ; if the Data has many Numerical cols then we mostly avoid Decision Tree.

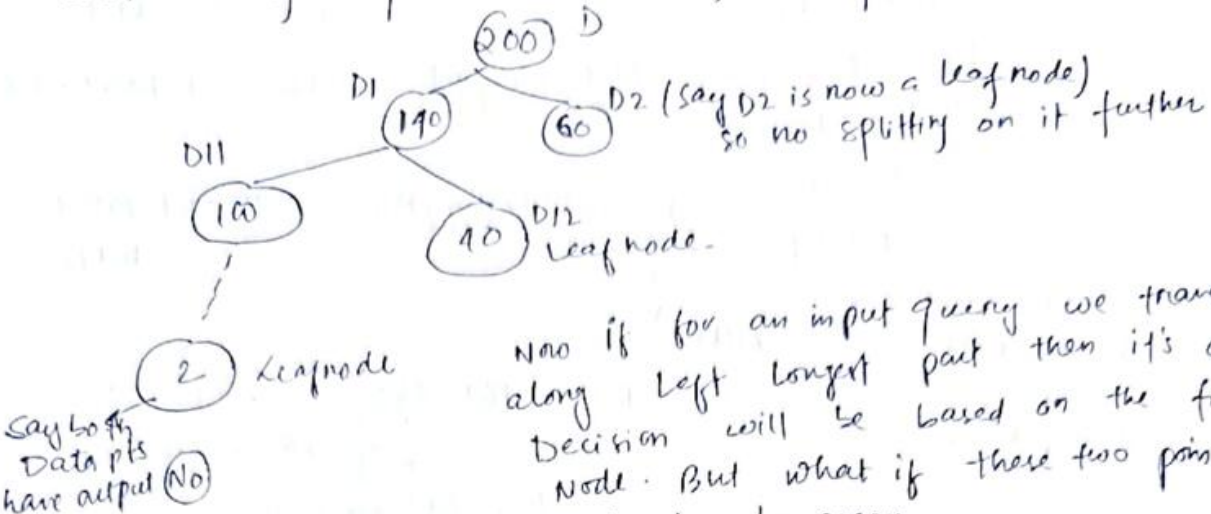
HYPERPARAMETERS (Overfitting & underfitting)

DT has a tendency to overfit (training data pe acche results deti hai par test data pe nahi).

- Depth of Tree → is the main thing jiske wajah se overfitting ya underfitting hota hai.

Overfitting → performs well on Training data
performs bad on Test data

Let's say we 200 rows in a data set & on the basis of splitting criteria of a particular column, we split it



Now if for an input query we travel only along Left longest part then it's output decision will be based on the final Leaf Node. But what if these two points are Noisy / outliers / Erroneous

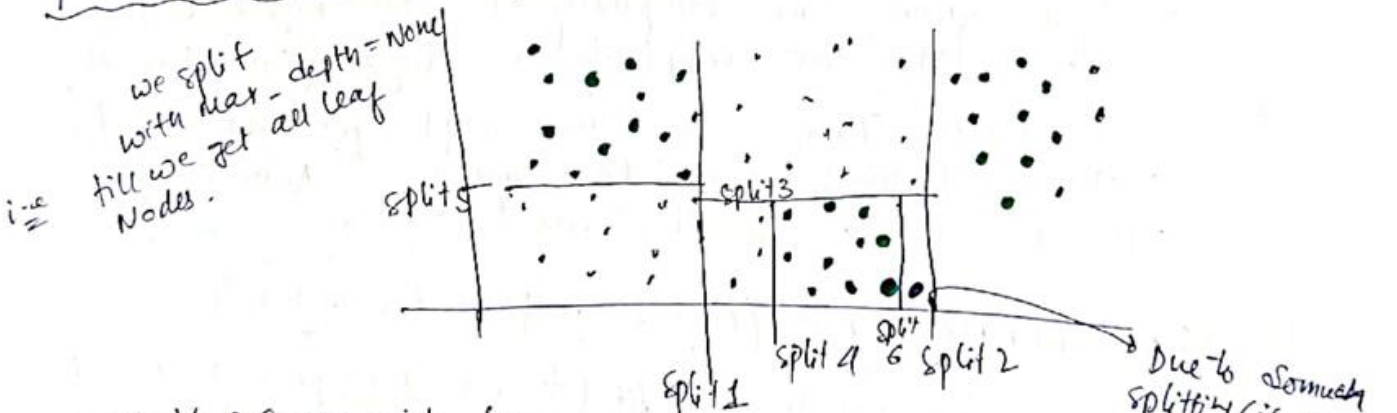
Hence:- all those if else to reach outliers → Then the system / Algo will fail.

↳ 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 Leafnodes & in such cases there is a huge chance of overfitting.

Geometric Intuition of Overfitting:-



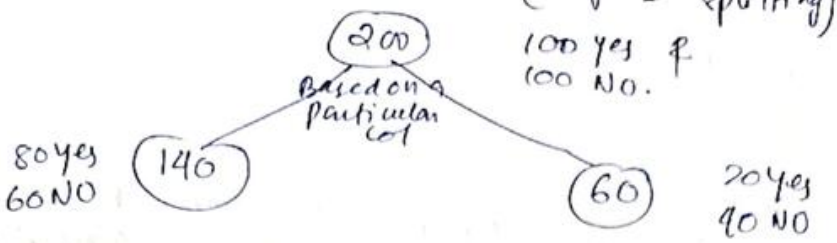
No if a query point falls in that box then even though it has high chance of being green (Yes) the Algo will label it to be Blue (No)

↳ this is case of overfitting.

Regression Problem)

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 as No.)

So we donot want both underfitting (low value of max_depth) & overfitting (high or None value for max_depth)

Jupyter NoteBook.

for More hyperparameter Tuning :- see again the video Decision Trees — Hyperparameters from 12:00 minutes.

imp hyperparameters are :- Criterion (gini, entropy)

splitter

Max_depth (by default = None)

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.)

splitter :- best (default), random

for numerical input col → we saw how we choose the splitting criteria → the one with best information gain

but sometimes we can keep it based on a col Randomly → random to split the Data

random reduces overfitting

∵ randomly select karte hai toh training data pe utna accha fit nahi kar pata Algorithm.

min samples split :- = 10 (say)

↳ It means suppose we are splitting Dataset now D1 will be further splitted iff min 10 rows are available in D1 & say on D2 & if there is further splitting then aage ke child Datasets ka splitting tbhi tak hi hoga if they have atleast 10 (rows/dataset) hai unme otherwise us child dataset ka splitting ruk jaiga.

D1 D2

↳ This is also other way of trimming our Decision Tree.

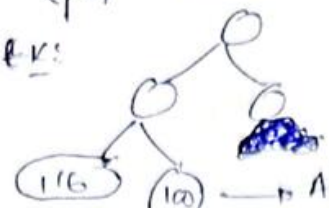
Higher its value → Higher chance of underfitting

Lower its value → Higher chance of overfitting (Think).

- **Min Sample Leaf** :- It is the same as above ki again min sample leaf = 4 ka matlab row atleast nahi hue in the leaf node. Toh usko split karne hi nahi doge.

Ex:

Say minsample leaf = 100



→ Algo will stop here :. again isko 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 → underfit
Smaller value → overfit.

- **Max features**:- we can choose ki hum Decision Tree ko kitne col dena chahte hai to work at every node.
  - used in case of high dimensional data & there is overfitting

  Say max features = 3 → then it will split based on the 3 cols. at every node.
  Random → (not the best one)
  :. if value = 1 then at every node a random col is selected & based on it split is done.
  ↳ This is done to introduce Randomness → hence reduce overfitting.

- **Max_leaf_nodes**:- How many Leaf Nodes we want in the final Decision Tree. → Higher value → overfitting
  Lower value → underfitting

- **Min Impurity Decrease = x (say)**
  at every node by splitting we decrease Error / impurity / entropy
  ↳ It means ki agar split karne pe based on a col agar min x impurity ghatega tabh hi split hoga warna nahi hoga.
  Higher value → underfitting
  Lower value → overfitting.

Decision Tree is imp because it is used in

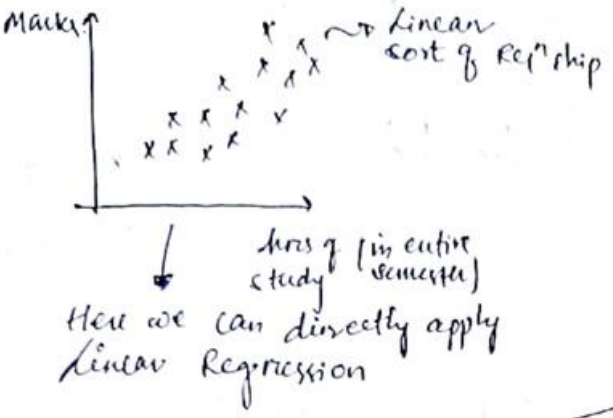
- Random forest
- Bagging
- Gradient Boosting

} so we need to know clearly ki Decision Tree ke konse hyperparameters ko kaise tune kare to get desired Result.

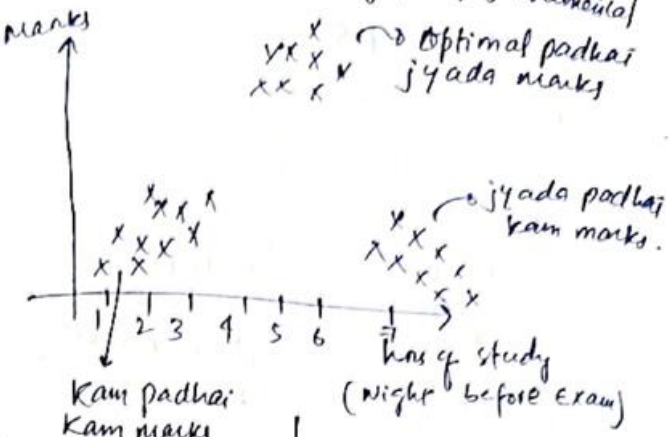
Regression Trees (Decision Trees on Regression problem)

Target col is Numerical

Where is it applicable?

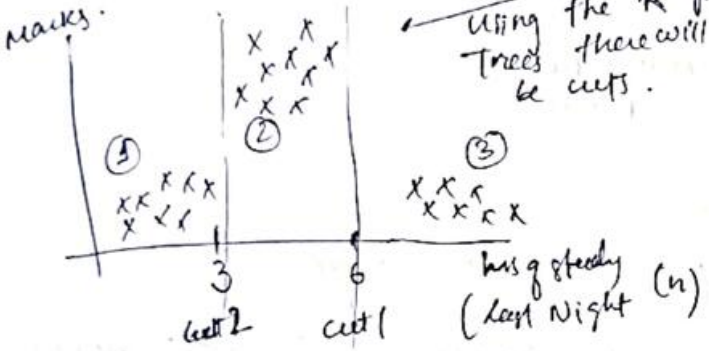


Here we can directly apply Linear Regression



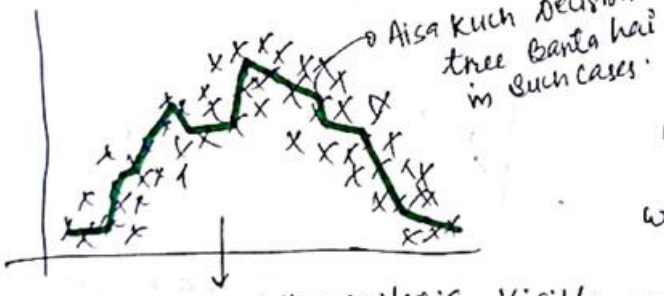
Here Linear Regression will not help.

Hence when we use Regression Trees pattern there is a non linear in the Data.

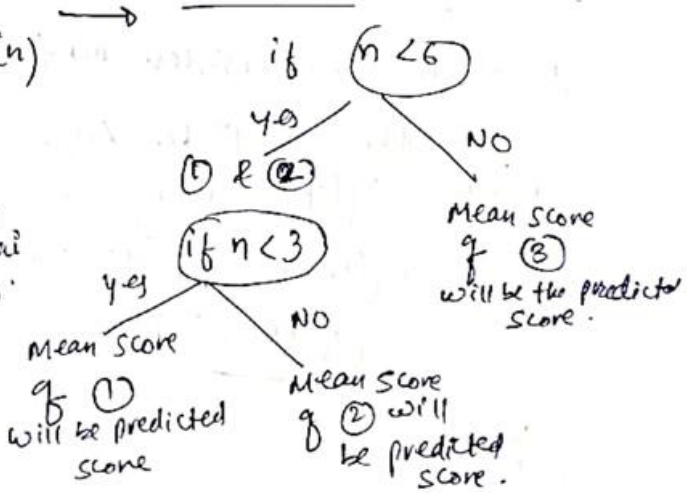


Using the Regression Trees there will be cuts.

But what if data Aisa hota

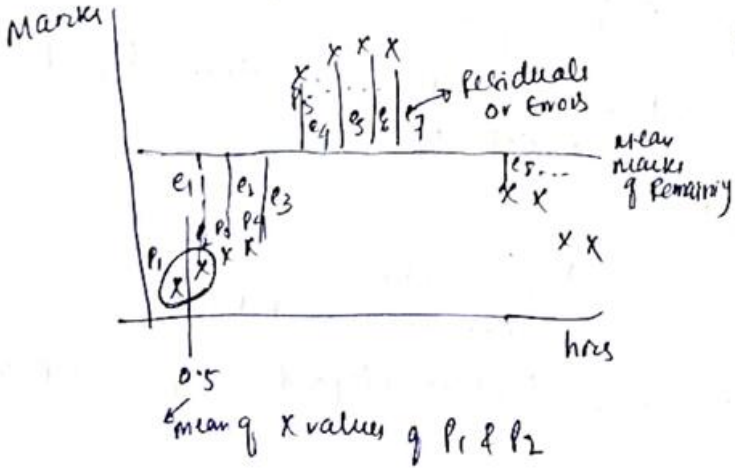


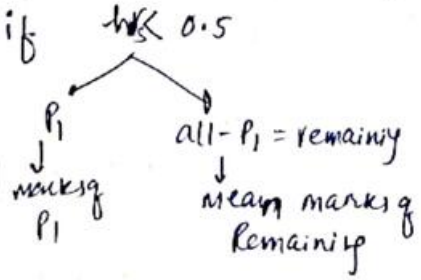
Code will be:-



Then splitting criteria visible nahi hai → Regression Tree handles this.

So how Regression Tree will work (Step 1:- select (if two)





Calculate :- $SSE_1 = e_1^2 + \cdots + e_n^2$

Sum of squared Errors

Now select another two

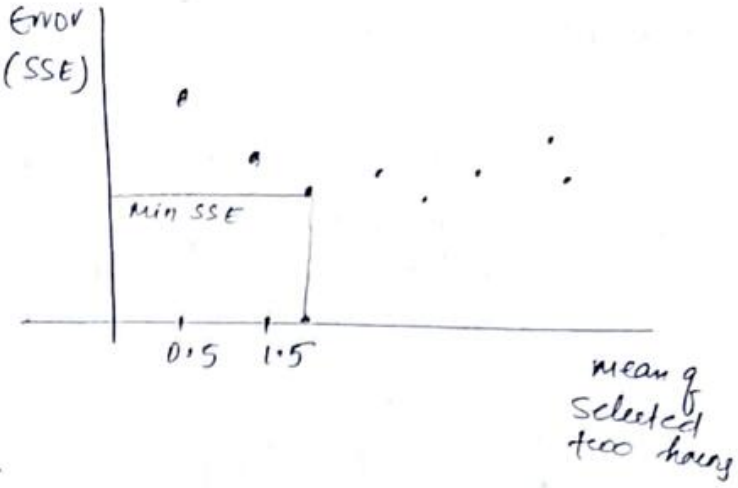
P_2 & P_3 → mean of their X-values = 1.5 (say)

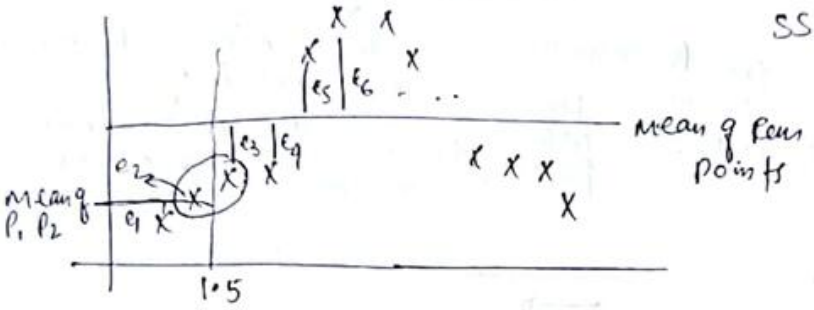
& proceed similarly.

Here:

hrs < 1.5

- P_1, P_2 → mean marks of P_1, P_2
- all − P_1, P_2 = rem → mean marks of Remaining.





$$SSE_2 = e_1^2 + e_2^2 + e_3^2 + \dots + e_6^2$$

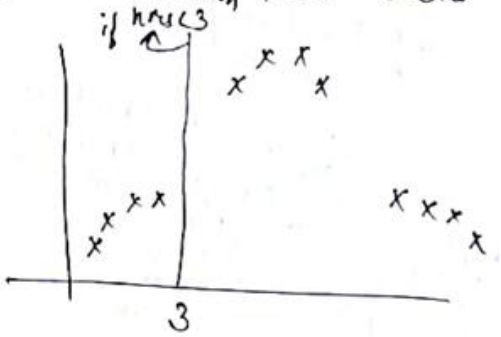
& so on we consider next two points & proceed.

∴ jis bhi cond^n ke liye SSE minimum hai wohi hamara pehla split banjaega.

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) → we only need to split in clusters.

for this if a child has ≤ x (threshold) # of elements we won't further split.

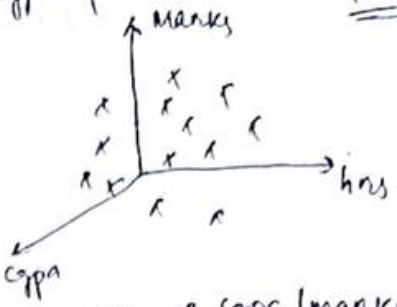
Say in above case we set it 4 & hence the left part of the subtree won't be splitted further.

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

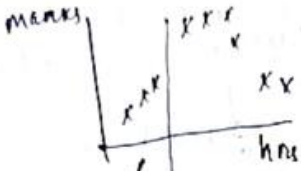
* what if we have input cols hours/cgpa & output col marks?



↳ 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

↓

Jske corresp we have min sum of squared Error = SSE_{hr}

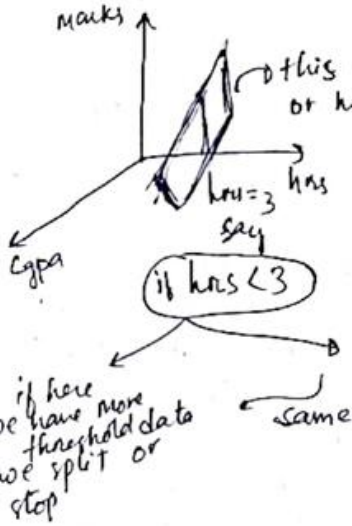
find the first splitting point using the above method.

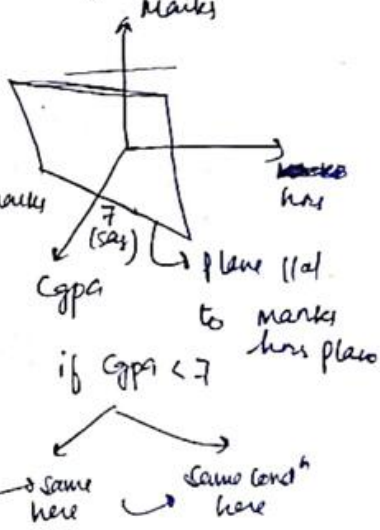
↓

Jske this corresp. we have min sum of squared Error = SSE_{cgpa}.

Now there are two cases:- $SSE_{hr} > SSE_{cgpa}$ & $SSE_{hr} < SSE_{cgpa}$

↓ in this case





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 :- → Product review ⇒ humme reliability milta hai ki humme product kharidne chahiye ya nahi in online Marketing

Bohot logone kharida but bas ek ne 4.5 rating diya (we'll not trust much)

But agar bohot logo ne 4.5 rating diya toh we'll trust it & buy it.

Similarly IMDB Rating for movies.

→ Democracy (jyadatar log agar ek party ko support kar rhe hai toh woh sahi hi hogi).

→ Experiment done → Animal on stage → its weight is asked by people - Everyone guessed it (no one was accurate) but the mean of everyone's guess was almost the same.

↓

Somehow crowd ko answer pata hota hai.

↓

Isi cheez pe based hai Ensemble Learning

↓

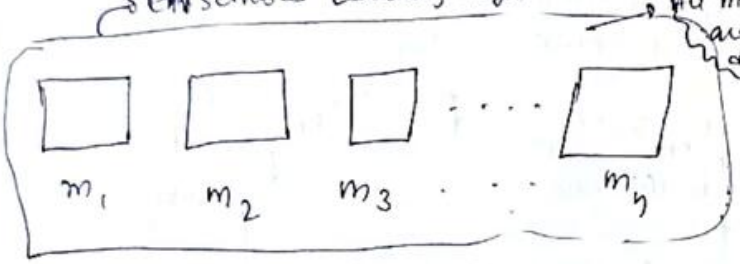
i.e. Crowd ko single ⇒ jyada pata hota hai

↓

Crowd should be mixed (not all of same type like software Engineers)

Prediction stage in Ensemble Learning

Say we have an Ensemble Learning Algorithm (Collection of smaller ML Algos / base models)



Say :- m_1 → Linear Reg (all different)
m_2 → SVM
m_3 → Decision Tree & so on.

Or :- m_1 → Linear Reg but during the training they were given different data & ∴ they were trained differently & hence variety is created.
m_2 → Linear Reg
m_3 → Linear Reg
∴ & 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 ie $m_1, m_2, m_3, \ldots, m_n$

Prediction: Y N N Y Y → Ab Democracy wala logic lagega (majority count) & wohi actual prediction

Ex: Soft :- (multi class classification problem) page 85

for a regression input query :- cgpa/IQ & to predict (placement in lakhs) page 83

the input is again given to all the models :-

$m_1, m_2, m_3, \ldots, m_n$

8.1 3.2 9.1 ... 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 = SVM
m_2 = Logistic Reg
m_3 = 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 Binabar value diya jaa rha hai

Bagging

↓ stands for Bootstrap Aggregation

↓ Random forest

Here the models are the same but different training data is given to each

say we have m_1, m_2, m_3 say SVM.

Now in Data we have 100 rows

↓ now we create variety of Data, by Bootstraping

D_1, D_2, D_3

Say each of them will have 500 datapoints then by Bootstraping we randomly choose 500 datapoints from Data & make it D_1 again out of 500 (1000 rows) random Datapoints → D_2 & so D_3 & Now for training the Ensemble model :- $D_1 \to m_1$; $D_2 \to m_2$, $D_3 \to m_3$

Note :- Bahot high prob hai ki D_1, D_2 & D_3 are different.

Now a new query point is given to all m_1, m_2 & m_3 & majority count or mean is predicted finally.

Random forest is special case of Bagging where the base models are Decision Trees

Boosting

- AdaBoost (adaptive Boosting)
- Gradient Boosting
- XG Boost (Extreme Gradient Boosting)

in Boosting again the models are same say

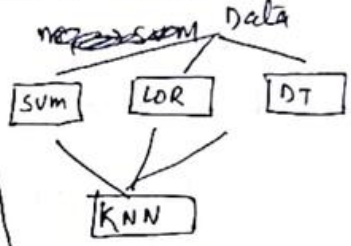
$m_1 = m_2 = m_3 =$ Linear Reg.

Here :- the Data is given to m_1 & m_1 keeps track of kaha kaha usne galtiya ki & yahi Data with the galtiya m_2 ko Explain karke pass kar deta hai & yeh m_2 galatiyo par sahi results dene lagta hai par iski khudki galtiya bhi hai & wo m_3 ko batata jata hai

This is Boosting → hence finally the model makes lesser mistake.

Stacking

is ~~it~~ 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



SVM, LOR (log. Reg) & DT will be trained on the Data & KNN will be trained on

SVM	LOR	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 konsa 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 prediction 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 | use Ensemble Learning
we'll get Low bias + Low variance.

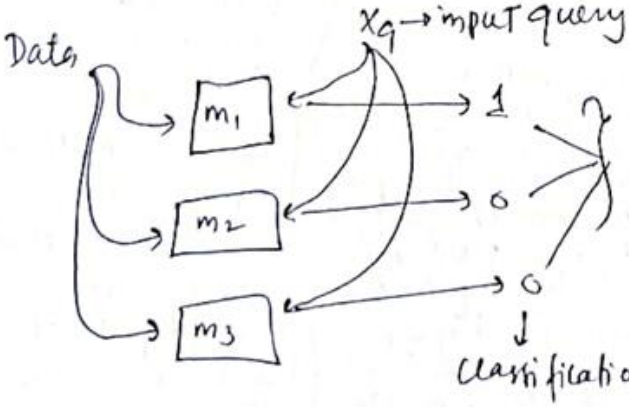
— Robustness — Data mei bohot jyada change bhi kardoge toh bhi performance jyada change/fluctuate nahi hota.

Disadvantage is :-
we are training multiple models & hence the computational complexity increases.

* When to use Ensemble Learning?
Ans :- Always.

VOTING ENSEMBLE

- core idea / Intuition
- Voting classifier (for classification problems)
- Voting Regressor. (for Regression problems).



Agar Regression problem hota toh jeeno ka Mean return karta as final prediction.

Ek:. Soft :- (multi class classification problem) ⇒

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.

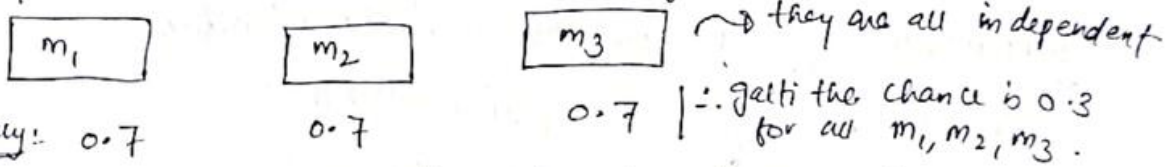
then how do they combine in voting Ensemble to give an overall accuracy that is better than all the individual accuracy (say 0.88) ↓ This we need to Discuss

Before that Let's Discuss the Assumptions behind voting Ensemble:-

- All the models are independent (kind of different)
- Saare individual models ka minimum accuracy should be > 51% nahi toh overall accuracy . min $\{m_1, m_2, m_3\}$ se bhi kharab hoga.

Why voting works is :- 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.

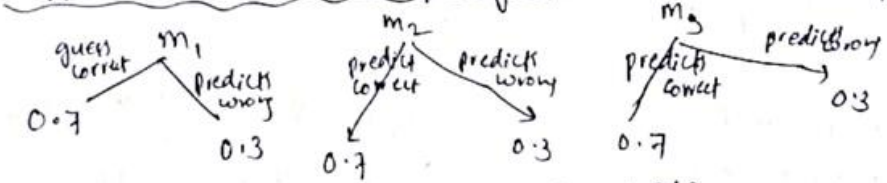
These are called base model. ↓ mathematically proving this (probability)

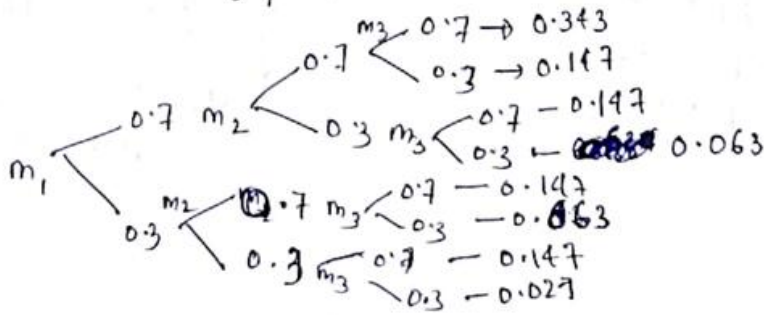


Now ab agar inka voting classifier banaye then majority jo bolega wohi final prediction hoga.

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 atleast two yes (majority count).

→ Hence the prob of that is:-

$YYY + YYN + YNY + NYY$

$= 0.343 + 0.147 + 0.147 + 0.147$

$= 0.784$

↳ Hence the prob to predict yes by the voting classifier is 0.784

↓ 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

we have discussed the core idea of voting classifier using majority count before.

→ see this video again from 3 minutes to understand & visualise again.

hyperparameters:- sklearn.ensemble.VotingClassifier

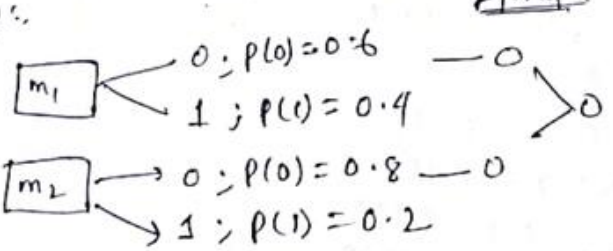
→ estimators : list of tuples of models ('str', model) type.

→ voting : {'hard', 'soft'} ; default :- 'hard'

if hard → uses predicted class labels for majority rule voting

if soft → predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

Example:-



Hard:

Soft

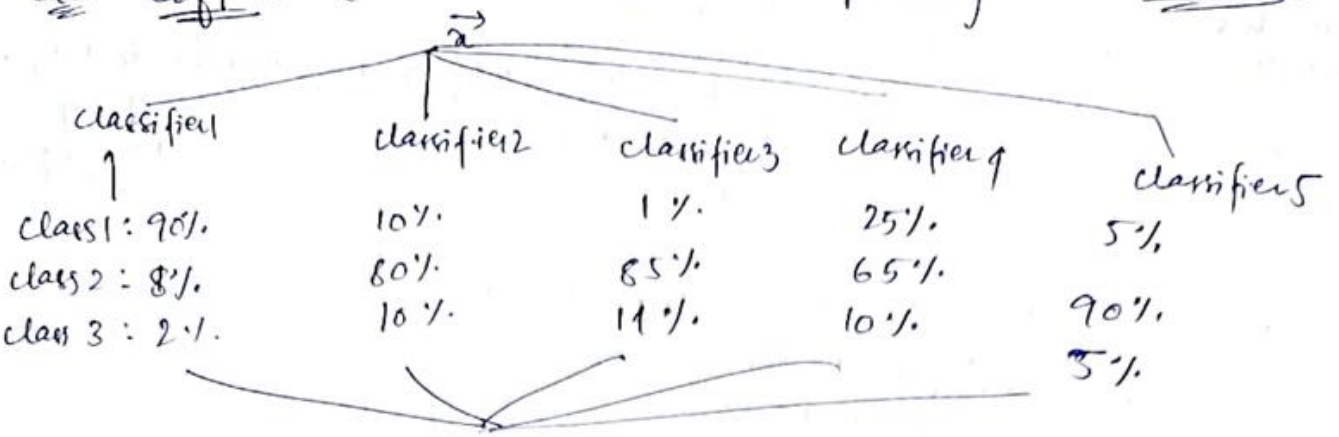
avg prob of 0 = $\frac{0.6+0.8}{2} = 0.7$

avg prob of 1 = $\frac{0.4+0.2}{2} = 0.3$

↓

∴ output = 0 ∵ av prob of 0 > av prob of 1.

Ex:- Soft :- (multi class classification problem)



Average :

class 1 : 26.2% → final prediction. class 2.
class 2 : 65.6%.
class 3 : 8.2%.

- 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, Gaussian Naive Bayes & Random Forest with soft voting gives better Result than using the same models with hard voting.

in soft voting the decision boundary ka smoothening hota hai & hence sometimes better result de deta hai.

Jupyter Notebook.

Voting Regressor (For Regression problem)

the hyperparameters are ← estimators list & weights.

↳ we already know how it works. the final output is the mean of the each individual output.

Jupyter Notebook.

Bagging Ensemble

in this all the Base model are same Algorithms.

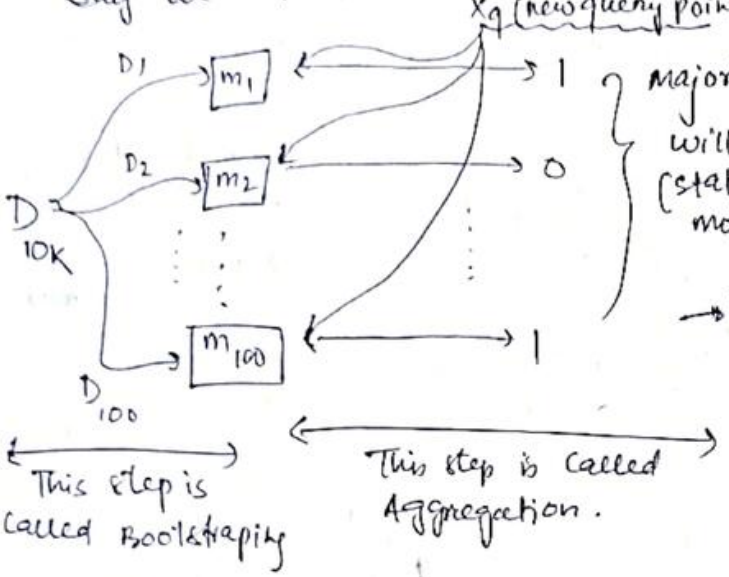
- Core idea (intuition, why use Bagging, when to use, Types of Bagging.
- Bagging classifier
- Bagging Regressor

Bagging → Bootstraping + Aggregation.

Bootstraping → Given population mei se samples draw karna 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 :- Har model alag Data pe Train hoga. (Say 1000 rows pe)



↓

This is done by Bootstraping :- randomly selecting 1000 rows from D10K (sometimes it can be with replacement & sometime without replacement).

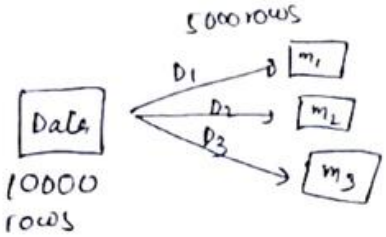
* Why use Bagging ? Why it works?

→ It is based on Bias variance tradeoff (we need Low Bias & Low variance) ! But they are inversely prop.

→ 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 with low neighbour value)

Now Bagging will reduce variance &



Now if we say change 500 rows (completely diff from original) in the original data & again Train then these 500 rows will be distributed Randomly. Again each Ek 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 karo toh bhi model ke output/performance mai jyada changes nahi aate ie. model consistent Result deta hai)

& Low bias (accurate) toh woo bhi hi hence overall Resultant model Low Bias + Low Variance ban jata hai.

Hence we use Bagging.

→ When to use Bagging?

Always, whenever you are dealing with low Bias + High variance model.

↳ implementation of Bagging is using Decision Trees as base models → Random Forest

for Classification Problem. Jupyter NoteBook

Bagging classifier (General Base models are:- Decision Trees, KNN & sometimes SVM)

- Base estimator/model (default is Decision Trees)
- # of estimators/models
- max_samples → How many rows do we want to sample for each model
- Bootstrap_samples = True → Means sampling with replacement ie D_i can have repeated rows.
- max_features → kitne features/cols ko use karna chahte hai.
- Bootstrap_features → Kitne features use karna chahte hai.

These are hyperparameters that we can tune.

isko = False means sampling without replacement means 'PASTING'. D_i's will not have repeated rows.

RANDOM SUBSPACES

↳ max_samples = all rows
Bootstrap_samples = False
max_features = not all cols (say 2 if there are 5 cols)
Bootstrap features = True.
↓ sampling on columns
This is done when we have large no. of rows like on MNIST dataset.

RANDOM PATCHES (Both Row & Col Sampling)

Say max_samples = 50 (50 rows for each model)
Bootstrap samples = True (with replacement)
max_features = 1 (say there were 3 cols)
Bootstrap_features = True.

Jupyter Notebook.

Bagging Regressor (for a Regression problem)

↳ in the Aggregation step → the final output is the mean of all the individual outputs & the Bootstrap step is the same as Bagging classifier.

Jupyter NoteBook

RANDOM FOREST (VV IMP)

↳ it is a bagging technique (bootstrap aggregation) ↳ random sampling

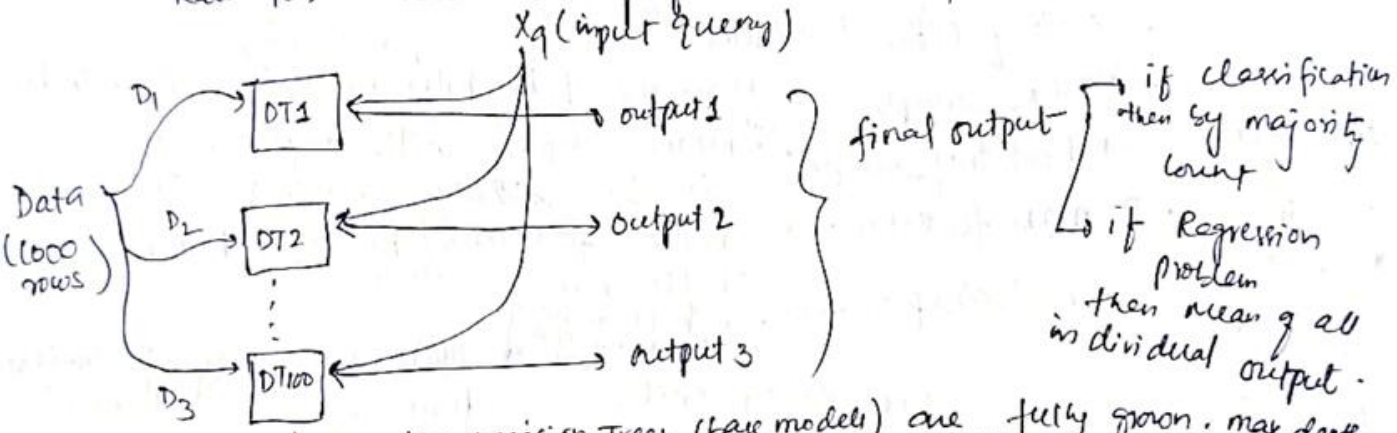
group of Decision Trees ↑

→ 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 jaroorat nahi hai. i.e. Hyperparameters bohot jyada change nahi bhi karo tab bhi iska performance kafi accha rehta hai.



↳ All the Decision Trees (base models) are fully grown; max_depth = None i.e. Low Bias + High variance.

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

Bootstraping

→ 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 & all 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. Page 89.

and so on.

Jupyter Notebook.

Bias-Variance & Random Forest :-

Good result on training data ↑ Low Bias & Low Variance. Good result on Test data ↑

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 trade off)

most ML Algos are :-

Low Bias + High variance
- Fully grown Decision tree
- SVM
- KNN (when #q 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 ie even if we change some data points in original data by some noisy data set then (unrelated data / outlier) due to bootstraping 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 also). But in Random forest the Base models is Decision Tree.
There is a seperate class for Random forest in sklearn & different class for Bagging.

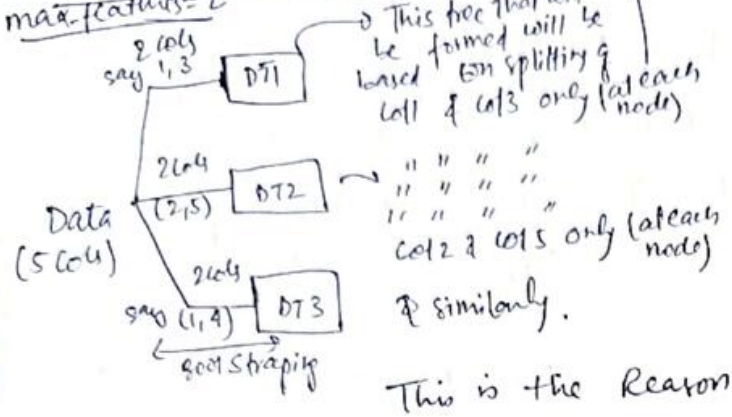
Question is:- Bagging with Base models — Decision Tree → Is it a Random forest → Ans:- No.

The difference is in feature Sampling (.max_features parameter)

Bagging

Column/feature Sampling

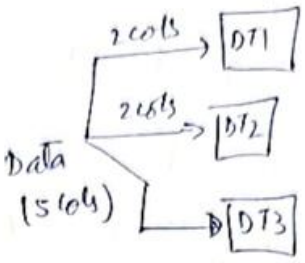
max_features = 2



Har node pe ya col1 split hoga ya col3

Random forest

max_features = 2



Jab ye tree ban rha hoga tab har node pe sampling hoga & based on that sampled col splitting hoga.

Har node pe Randomly 2 cols sampling hoga & phir splitting hoga.

Here there is more randomness.

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 RandomForest :-

- number of estimators :- decides the No. of Decision Trees to be used to make the random forest. default = 100
- max_features : int, float, sqrt, log2, None, auto (default). :- # 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 we choose max_feat = 2 then at every node creation two cols will be randomly selected and then from that two a col will be selected for splitting.
- Bootstrap :- if we want to select the rows/cols randomly or not - (default : True), if false the whole dataset is used for each Decision Tree.
- max samples :- # of rows we want to give to each Decision Tree - (if bootstrap is True)
  Generally 50% to 75%.
  - None (default) then use X.shape[0] samples
  - if int then jo diya hai utna use karo
  - if float (0,1) then 0.7 means 70% of rows use karo.

already discussed earlier

- criterion :- gini (default) or entropy
- max_depth : int, None (default) → fully grown
- min_samples_split → itne samples agar hai node mei tohi further split hoga.
- min_samples_leaf → split karne ke baad agar leaf mei atleast itne samples honge tab hi split hoga.
- max_leaf_nodes → kitne leaf nodes we want to keep in each Decision Tree.
- min_impurity_decrease :- agar node ko split karne pe atleast itna impurity kam hoga tab hi split hoga.
- n_jobs :- (int, None (default)) (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 bootstraping samples (rows) used when building trees (if bootstrap = True) and the sampling of the features to consider (cols) when looking for the best split at each node (if max_features < n_features ↳ total no. of cols)
- warm start (default = False)
  used when stages mei model ko Train karte hai (when set to True, reuse the soln 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 Regressor :- 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 Bootstraping (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.

the performance.

and this out of Bag samples can be used for Testing the 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 Note Book.

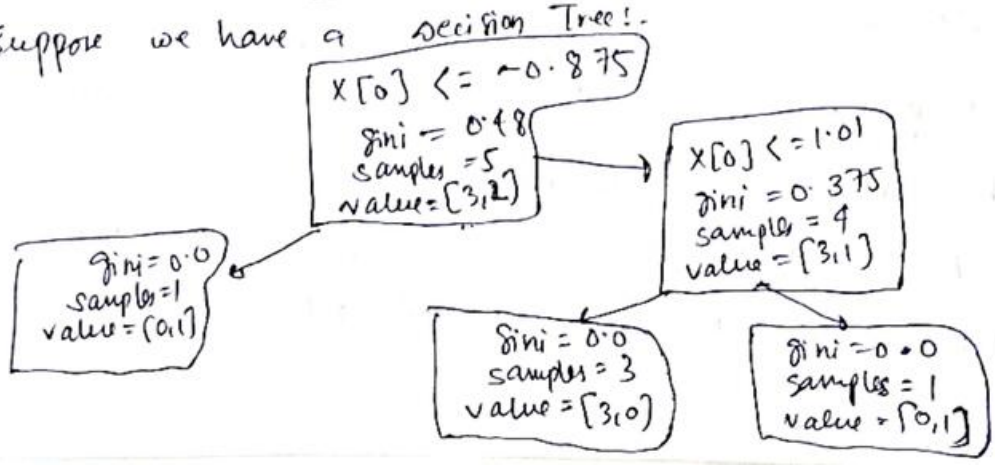
How is feature importance Calculated.

+ to understand how it is Calculated in Random Forest we need to how how it is Calculated in Decision Tree.

In Documentation of DecisionTree classifier :- there is an attribute :- feature_importances_
+ Returns the feature importances
+ The importance of feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as Gini importance.

Jupyter Note Book.

Suppose we have a decision Tree :-



Now:- Kisi bhi col/feature ka importance calculate karna hai using Decision Trees :-

$$n_i = \frac{N_t}{N} \left[\text{impurity} - \left(\frac{N_{t,R}}{N_t} \times \text{right\_impurity} \right) - \left(\frac{N_{t,L}}{N_t} \times \text{left\_impurity} \right) \right]$$

- N_t = # of rows in node n_i
- impurity = (gini value) of that node
- $N_{t,R}$ = # of rows in right node of n_i
- N = Total # of rows present in data
- N_t = # of rows in node n_i
- right_impurity = gini value of right node of n_i
- left_impurity = gini value of left node of n_i
- Dono terms have similar meanings

Le node ka importance

$$f_{I_k} = \frac{\sum_{i \in \text{node split on feature } k} n_i}{\sum_{i \in \text{all nodes}} n_i}$$

- f_{I_k} = feature imp of k^{th} col
- $i \in$ all nodes on which splitting occured.

In the diagram we have two cols $X[0]$ & $X[1]$.

Now to find feature importance of $X[0]$, we need to find all the nodes in the diagram/Tree jispe splitting $X[0]$ col ke wajah se hui hai. → $X[0] <= 1.01$ → only 1 node.

& iss node the importance find karenge pehle

$X[0] <= 1.01$
gini = 0.375
samples = 4
value = [3,1]

Hence:- then feature importance of $X[0]$ = $\frac{x}{x+y}$

- x → sum of all node importance of those node jispe splitting $X[0]$ ke wajah se hui hai
- y → sum of all the node importance jispe splitting hui hai irrespective of kis col ke wajah se.

* node importance of $X[1] <= -0.375$

$$= \frac{5}{5}\left[0.48 - \frac{4}{5} \times 0.375 - \frac{1}{5} \times 0\right] = 0.18$$

* node importance of $X[0] <= 1.01$

$$= \frac{4}{5}\left[0.375 - \frac{1}{5} \times 0 - \frac{3}{5} \times 0\right] = \frac{4}{5} \times 0.375 = 0.30$$

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

Suppose there are 10 Decision Trees in a Random Forest:-
then har decision tree col1 ka feature importance nikalega (considering ke isse kitni pe split hue ho). Then Actual/final feature importance of col1
= 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:-

- Weak Learners:- Aise ML Algo jiska accuracy 50% just thora sa jyada hota hai.
  (opposite:- strong Learners:- jinka accuracy bohot accha hota hai 95%, 99% etc.)

  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.

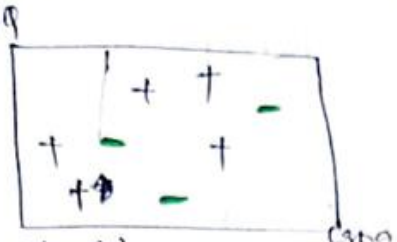
- +1 and -1 :- Generally in Binary classification problem the classes are 1 & 0, here in adaboost the classes are 1 (for +ve points) & -1 (for -ve points).
  ↳ pts in +ve Region. ↳ pts in -ve Region

The Big Idea of AdaBoost:

It is a stagewise additive method.

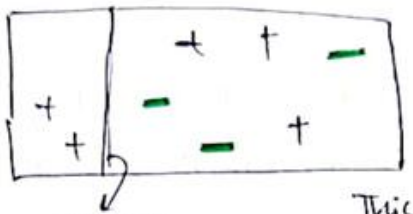
Cgpa | Iq | placed

- Not placed
+ placed



Say we have 3 Base models (decision stumps)

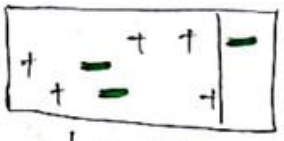
Step 1: - first Decision Stump will be done (splitting) :- & the splitting will be done based on that column jisse entropy zaada kam ho



for representing the model
(output -1 or +1)

This splitting made some mistakes & when passing this data to the next decision stump, we will do upsampling ie we will ask the 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.

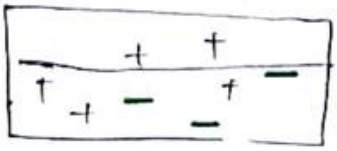
Step 2 - 2nd Decision Stump:- focussing more on first Decision Stump mistakes makes a split such that the entropy is lowest possible

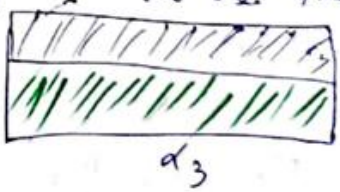


is ke performance ka bain pe iska weightage α_2 calculate hoga.

↳ we see isne Decision Stump 1 ke mistakes toh thik kar diye but iske ab khudke mistakes hai → (-ve wale in true side)

Step 3: 3rd Decision Stump (the -ve in true regions) are upsampled & are focused on more here & again split is performed to have the highest entropy possible





iska weightage α_3 hai

Suppose isne pehle wale ki galti thik kardi but ab khudki galti kar diya.

Now we have three Base models (on the right, diagram) & their weights (α_i). Now we add the three models so Adaboost model

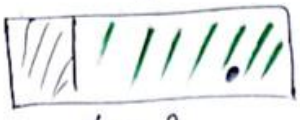
$$A = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

↳ if pos $A = +1$
↳ if -ve $A = -1$

Now let's do a prediction :- Cgpa = 8.5, IQ = 81

— Not placed
— placed

1st model.



$\alpha_1 = 2$ → −1 (not placed)

2nd model.

$\alpha_2 = 10$ → +1 (placed)

3rd model.

$\alpha_3 = 1$ (say) → (-1) (not placed)

∴ final prediction :-

$$H = \text{sign}(\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3)$$

$$= \text{sign}[2(-1) + 10(1) + 1(-1)] = \text{sign}(7) = +1$$

∴ the student will be placed.

final decision boundary ke liye teeno models ke decision boundary ko ek ke upar ek rakhdo. we get:-

Adaboost step by step

n = 5 = # rows

X_1	X_2	Y
3	7	1
2	9	0
1	4	1
9	8	0
3	7	0

→ initially hum har apne row ko ek weight assign karte hai (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!-

weight
0.2
0.2
0.2
0.2
0.2

Stage 1 :-

1) we train a Decision stump (m_1) (split with max entropy ke or max information gain) or max gini impurity ke.

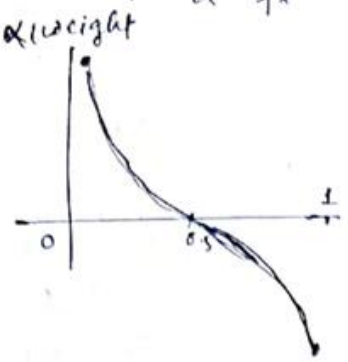
Now iss m_1 ka performance dekhenge on the data set & we find Y_pred.

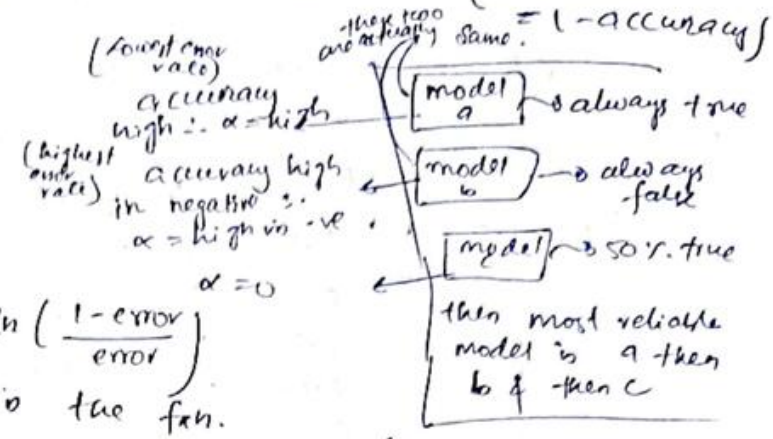
X_1	X_2	Y	Y_pred	weight (initial)
3	7	1	1	0.2
2	9	0	0 (X)	0.2
1	4	1	0 (X)	0.2
9	8	0	0	0.2
3	7	0	0	0.2

→ wrong prediction.

Now we find α_1 = weight of m_1 on final prediction.
↳ it depends upon error rate (m_1 ne data pe kitni galti ki)
error rate jitna jyada, model utna kharaab & α_1 (generally error rate = 1 − accuracy)
utna hi kam. ∴ $\alpha_1 \propto \frac{1}{\text{error rate}}$

∴ we want a fun with graph





$$\alpha = \frac{1}{2} \ln\left(\frac{1-\text{error}}{\text{error}}\right)$$
↳ this is the fun.

To calculate error of a Decision Stump:-
from the Table find all the rows that are misclassified by m_1 (different y_{pred} than actual y) & then un rows ka initial weight ka sum:- So for m_1; error = 0.2 + 0.2 = 0.4.

$$\therefore \alpha_1 = \frac{1}{2} \ln\left(\frac{1-0.4}{0.4}\right) = \frac{1}{2} \ln\left(\frac{0.6}{0.4}\right) = 0.20$$

$$\therefore \boxed{\alpha_1 = 0.20}$$ ↳ itna weight hoga m_1 ka in final prediction.

Ab m_1 ke galtiyon ko upsampling karke (importance badha kar) Stage 2 (m_2) ko bhejne hai:-

To increase the importance of mistakes of m_1:- Galtiya (misclassified) wali rows ka weights ko increase (Boost) kar denge & classified wale points ka weight kam 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: wt = $0.2 \times e^{0.2} = 0.24$
for class. pts wt = $0.2 \times e^{-0.2} = 0.16$

new table-

X_1	X_2	y	y_pred	weight	updated weights	normalized	divide by sum = 0.96
3	7	1	1	0.2	0.16	0.166	
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

n = 5

X_1	X_2	y	new_wt	range	cumulative sum upper value - lower value
3	7	1	0.166	0 – 0.166	
2	9	0	0.25	0.166 – 0.416	0.166 + 0.25
1	4	1	0.25	0.416 – 0.666	0.416 + 0.25
9	8	0	0.166	0.666 – 0.832	0.666 + 0.166
3	9	0	0.166	0.832 – 1.0	0.832 + 0.166

now we make a dataset with only these rows.

upsampling

5 random numbers generated b/w 0 & 1.

Say 0.13 → Row1
0.43 → Row3
0.62 → Row3
0.50 → Row3
0.8 → Row4

Since wt of Row2 & Row3 are more therefore the possibility of a number falling in that row range is higher since that range is bigger.

∴ Dataset for stage 2.

Row1
Row3
Row3
Row3
Row4

→ this will go to the second Decision stump & everything will be repeated.

Jupyter Notebook.

Adaboost Hyperparameters:

- base_estimator (default:- Decision Tree with max_depth = 1) → this gives the best result.
  → otherwise we can use Logistic Regression, SVM we want to use
- n_estimators = # of base models (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 uses the contributions of each classifier. There is a trade off b/w learning rate & n_estimators parameters.
- algorithm (samme, samme.R) → default.

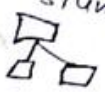
Jupyter Notebook

Bagging V/S Boosting

— Type of model used :-
Target is to achieve Low Bias + Low variance in Both

in Bagging: the base models have Low Bias + High Variance → like fully grown Decision Trees. (or KNN with low neighbours)

in Boosting: the base models are High Bias + Low variance (like: shallow Decision Tree :- Tree whose depth is small). ie. Decision Stump



— Sequential V/S parallel Learning
in Bagging the Dataset is randomly split into subDatasets & given to different base models. } Parallel Learning (saare models saath mei train hote hai)

in Boosting: Data → model1 then to model2 ... and so on. Sequential Learning (Model ek ke baad ek train ho rhe hai).

— Weightage of base learners
in Bagging the base models/learners have weightage = 1 (by default) (sabki ka vote same hai)

in Boosting: har model ka apna weightage (α_i) hota hai. jiska weight jyada uska contribution jyada in final output.

K-MEANS CLUSTERING

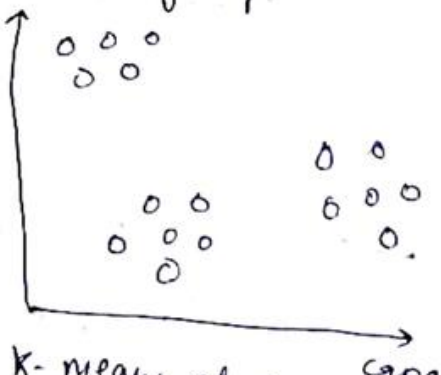
Clustering is an unsupervised ML technique where we put similar things in a group.

Internal working

Data → IR

2D mei dikh rahe hai toh hum dekhkar bhi cluster kar denge & bol payenge ki konsa point kis cluster mei jaiga.

But ye agar higher dim mei hota then hum clusters deken nahi payenge. & then we use K-means clustering.



Steps: 1. Decide n clusters
2. Initialize centroids
3. Assign Cluster
4. Move (y) centroids
5. Finish.

1./ Decide n clusters :- we'll have to tell the Algo how many clusters we want (wiend) → 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 chumme har point ko EK cluster assign karna hai on the basis of the 3 centroids chosen.

EK point ka teeno cluster centroid ⊕ Euclidean distance nikalo & jis centroid ⊕ distance sabse kam hai usse us centroid ke cluster mei rakh do.

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 cggpa & points in cluster 3 , mean of ⊕ of point in cluster 3)

↳ This need not be a point on the Dataset
↳ It will be at the centre of the cluster 3.

Silly. we will find centroids of cluster 1 & cluster 3.

* To check we should finish the Algo are not :-
compare the new centroids to the previous step centroids, if they are the same then no improvement in centroids → finish the Algo & if they are different then Algo is improving by moving the centroid → continue. step3 ⟶ step4 ⟶ Do this till finish cond^n is met

step3 ↓ Assign clusters again wrt these new centroids.

step4 ↓ move centroids again using the above formula

↳ centroids same rehna ka matlab hai ki points in that cluster are not changing, no new points are added or no points are getting removed ie. cluster is now fixed.

↳ The only metric we used in this Algo is Euclidean distance which works in higher dimension also & ∴ K-means clustering is such a powerful Algo.

* How do we guess the no. of clusters (K) at the beginning of the K-means clustering Algo?

(ELbow method)

WCSS (inertia) = within clusters sum of squared distance

In this we plot a graph called Elbow graph/curve

To find WCSS for a cluster → ek cluster lo aur uska centroid ka Euclidean distance us cluster ke har points se Nikalo then sab ka square karke, sum karlo

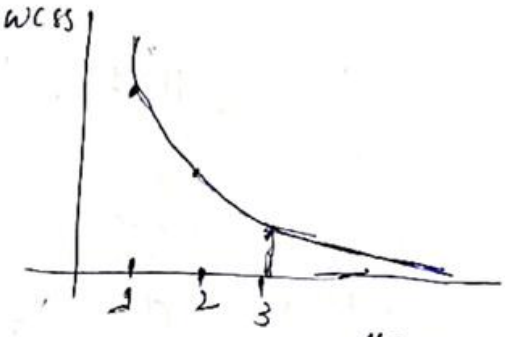
if there are multiple clusters then WCSS of big cluster = Sum of individual WCSS.

* to form the Elbow curve:

pehle maanlo ki pura dataset ek hi cluster hai:- uska WCSS nikalo & plot karlo $(1, WCSS_1)$

Now consider two clusters in the Data set,
Total WCSS = sum of individual WCSS
plot $(2, \text{Total WCSS}_2)$

Next:- consider 3 clusters in the Dataset,
Total WCSS = sum of individual WCSS
plot $(3, \text{Total WCSS}_3)$
and so on do it till 10, 15, or 20 (at most in General)



It is obvious that:- $WCSS_1 > WCSS_2 > WCSS_3 > \dots > WCSS_n$.

(Think :: the new centroids are getting closer to the points as the clusters are increased & hence WCSS does)

if all the points are centroids i.e. #of clusters = #of points in Dataset then individual WCSS = 0 ∴ Total WCSS = 0

Now in the graph we need to find the elbow point:-
The point in the graph where the decrease seems to stablizes. (steepness/slope less, stablizes at that point).

& that elbow point is the #of clusters.

Jupyter Notebook.

on salary then m_1.

GRADIENT BOOSTING

→ Sequential stage wise Addition matlab data ek ke baad ek base models pe train hoga.
& har next base model ko apne pichle wale model ka galtiyon ko pata hoga & wo un galtiyo ko nahi karega.

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

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 /predicts

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

step 2. Now we need a loss fn that tells ki pichle model ne galti kitni ki & woh galti hum next model ko batainge.

res1
-1.8
-0.8
3.2
1.2
-1.8

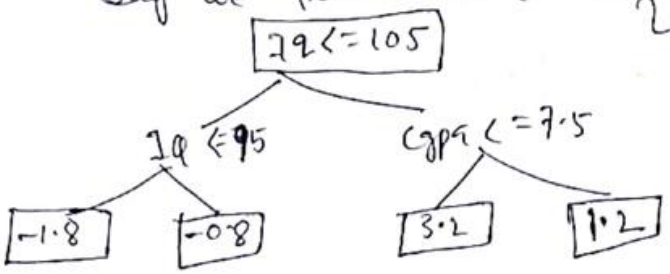
Here Loss fn = pseudo residual = actual - predicted
↑ new col of pseudo residual of m_1
& Now is galti ko humme somehow model 2 tako bhejna hai

Our model 2 generally is Decision Tree (Gives best Results).

Note:- To model 2 we give: IQ | cgpa | res1 (mistakes of m1) output i.e. by model 2 we want to predict the res1 & jitna accurately

m2 res1 Pred. karega hum utna accurately hum actual output (salary) predict kar payenge.

Say we trained m2 (Decision Tree) & we get:-
→ using Decision Tree Regressor



Now we predict on the data fitting m_2 →

IQ<105

IQ<95 | cgpa<7.5

-1.8 | -0.8 | 3.2 | 1.2

use this

IQ	cgpa	resl	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

↳ overfitting (since pure training data pe exactly same predict kar diya).

Now we have 3 base models but say agar 2 hi hote then Actual prediction of salary kaise hota :-

Actual pred = m1 pred + m2 pred (which is obvious also)

↳ ye pred. kaafi accha hone chahiye

of m2 on salary

Ex:- for input query:

IQ	cgpa
90	8

m1 pred = 4.8 (mean)

m2 pred = -1.8 (mistake of m1)

∴ Actual pred = 4.8 - 1.8 = 3

↳ yaha overfitting ho toh rahi hai (since exact prediction ho rha hai hence naye data pe accha prediction nahi hoga)

So to avoid this we use Learning rate :-

of m2 on salary

90 Actual pred = m_1 pred + lr * m2 pred

generally lr = 0.1 ↳ ye lene se hamara step chota chota hai towards the right direction (actual Ans)

Now for input :-

IQ	cgpa
90	8

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 (by m1)	resl (salary - pred)	pred2 (by m2 on resl)
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

= res2

= salary - (pred1 + lr * pred2)

mistakes of pred of m_2 on salary

res2
-1.62
-0.72
2.88
1.08
-1.62

↳ clearly we see that values of resl > res2

hence the mistakes are less hence m_2 did lesser mistakes on salary than m_1.

Till here :- Actual pred of m2 on salary = pred1 + lr * pred2 (lr = 0.1)

Ideally we want the residual = 0 & as the no of base models ↑es the residual ↓es.
So we want the residual to be 0 or almost zero.

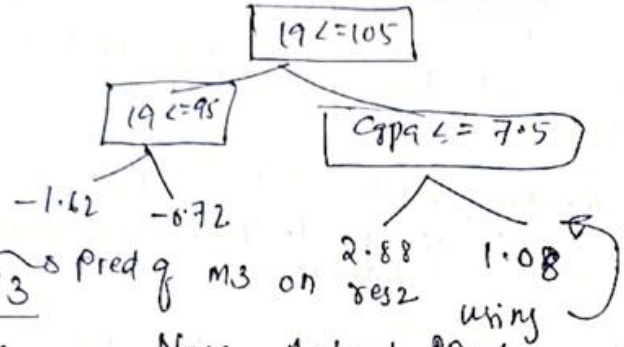
m3 will again be Decision tree

Now the data will be

IQ	cgpa	res2 (output col)
:	:	:

i.e m3 will predict the mistakes of m2. & more accurately it predicts the mistakes of m2 the better it will predict the Actual output (salary).

model 3 (Decision Tree)



Pred 3 ← Pred of m3 on res2 using

Pred 3
-1.62
-0.72
2.88
1.08
-1.62

Now Actual Pred of m3 on Salary

= pred1 + lr x pred2 + lr x pred3 (lr = 0.1 same for all base models.)

Say input col :- IQ | cgpa
90 | 8

then :- Actual Pred = 4.8 x 0.1 * (-1.8) + 0.1 x (-1.62)
= 4.8 - 0.18 - 0.162
= 4.458 ← Again Actual Avg = 3

So we need more base models

But this is how Gradient Boosting works.

Jupyter Note Book.

AdaBoost V/S GradientBoost

1) Max_Leaf nodes :- in Adaboost we use Decision Stumps ie Decision tree with max depth = 1 ∴ max_leaf nodes = 2 (fixed)

in Gradient Boost max_leaf nodes ranges from 8 to 32
↳ for small data set ↳ for big data set.

2) Learning Rate :- In Adaboost :- Every model (base) gets a weight & they can be different & tells ki konsa model jyada imp hai.
In Gradient Boost we use learning rate & it is same for Each Base model.

math for Gradient Boosting classification

Concept of Additive modelling

What ML models do? $\frac{x}{\text{input}} \mid \frac{y}{\text{output}}$ — then ML models find/represent $y = f(x)$.

If Data is: $x_1 \mid x_2 \mid x_3 \mid y$ ⇝ then $y = f(x_1, x_2, x_3)$

if the relⁿ b/w input & output are simpler then the relⁿ is found easily (linear, polynomial) but if the relⁿship is complex we use additive modelling.

So in additive modelling we find the bigger fⁿ y or $F(x)$ as sum of smaller fⁿˢ: $y = F(x) = f_0(x) + f_1(x) + \cdots + f_n(x)$.

in Ensemble Learning these fⁿˢ are given by base models.

Additive modelling is a statistical technique for modelling complex relⁿships between variables by breaking them down into a sum of simpler relⁿships. The idea is to add up the simpler effects of the predictors to model the response, rather than attempting to model the response as a complicated fⁿ of the predictors.

Ex: modeling the relⁿship b/w temp, rainfall & crop yield. Instead of trying to model the relⁿship b/w these variables as a single complex eqⁿ, an additive model would break it down into three seperate, simple relⁿships, the relⁿship b/w temp & crop yield, rainfall & crop yield & temp & rainfall. These simple relⁿships can be added together to give a final model of the relⁿship 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 $L(y, F(x)) = L(y, \hat{y})$

x_i is input data (it can have multiple col) & y_i is output data $\{(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)\}$ and number of iteration = M

$\hat{y}$ → model ka predicted output

the loss fⁿ that we will use here is Least square just ½ ki multiply kardenge for mathematical convinience

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

in our ex in jupyter Notebook.

X_i : R&D spend, Admin's, Marketing spend

y_i : Profit & we have only x_1, x_2, x_3 & y_1, y_2, y_3

Step1:- Initialize $f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$.

we have to find $y = F(x)$

& Since this is a Boosting Algo :- $\therefore F(x) = f_0(x) + f_1(x) + \cdots + f_n(x)$

every fⁿ is a Decision tree.

Now to find $f_0(x)$.

use $\frac{1}{2} \sum_{i=1}^{3} (y_i - \hat{y}_i)^2 = L(y_i, \hat{y}_i)$

& diff wrt γ & make it $= 0$ 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.

KNN. (K Neanest Neighbours)

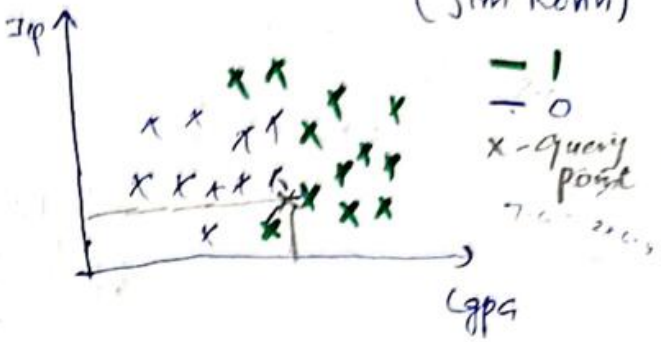
* you are the average of the five people you spend the most time with. (Jim Rohn)

Consider 100 students Dataset

Cgpa	Iq	placement
8	80	1
7	70	0
⋮	⋮	⋮

plot the input

input — output



The logic of KNN is ki aap apne neighbours ki tarah hote ho.

We first check / choose the value of K (say = 3)

Now say we have a query point (see diagram, pencil)

Now we find the distance of this query point from all the points in the training dataset. (generally euclidean) (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

$K = \sqrt{n}$ (n = # of observations = # of rows in the training data set)

Generally, we avoid even value for K since majority voting mei dikkat hoge.

Experimentation (Better)

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 classification in: SVM, Log. reg, Dec. Tree, Neural Networks.
↳ for 1D, 2D or 3D Data
: for higher dim we cannot imagine the Data.

Consider 2D Data

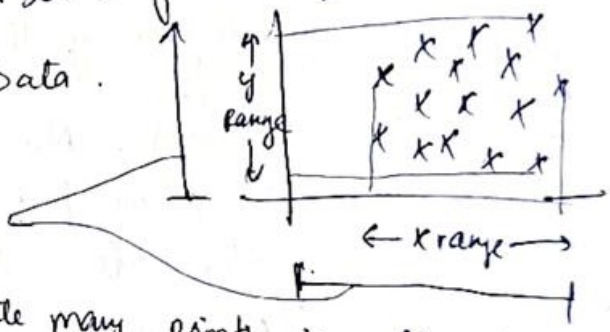
Cgpa	IQ	placement

↳ Binary classification problem (output has two classes)

K=5

then it is in the red decision Region hence uska placement Red wala ke taraf hi hoga.

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 (trained)
⑤ send the meshgrid points generated to the knn model
if knn predicts 0 → blue ↑ encode karo us meshgrid
if knn predicts 1 → Red ↓ point ko
" .
& 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 surfaces.

Overfitting & underfitting.

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
↓
Minor changes ko bhi capture kar rha hai model. (High variance → alag alag dataset pe alag alag prediction karna.)

K = n = # of rows. (high value)

↳ Here the output will be that class which is dominating i.e. jis class mei sabse jyada elements hai wohi dominate karega & Decision Region bhi bas ek usi class ka banega.
Hence output of any query point is the majority class

This is a case for underfitting.
The model is unable to understand the patterns of the Dataset.

Limitations of KNN

→ KNN is a lazy learning technique (Aise ML algo jisme saara kaam prediction mei hi hota hai (distance calculation). ↓ Training me toh bas data plot hota hai.)

(1) We would not use KNN on large datasets like # of rows/observations = in lakhs & # of features/cols = 100 (say) bcz query point ka har point se distance nikalna hai & that will take a lot of time. & hence the prediction will be very slow.
↳ (5 lakhs distance calculate → sort them → use majority count)

(2) High dimensional data (# of cols = say 500) then curse of dimensionality aajata 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 outlier ke nearest neighbours ko dekhkar hi predict kar dega.

4) Non-homogeneous feature scales
↳ To 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
eg 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%) wala 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 cgpa ka jyada contribution tha ya IQ ka.
↳ cannot tell ki konsa feature jyada imp hai prediction ke liye
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.

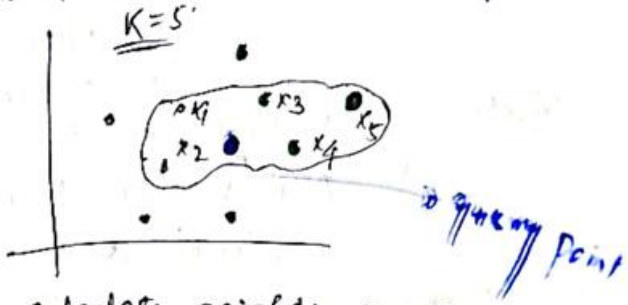
KNN Hyperparameters.

- n-neighbours → k value (default = 5)
- weights :- (default = 'uniform')
  ↓ all points in each neighbourhood are weighted equally.
  Other values:
  → 'distance' : weight points by the inverse of their distance. In this case closer neighbors of a query point will have a greater influence than neighbours which are further away.
  → callable : a user-defined fn which accepts an array of distances & returns an array of same shape containing the weights.
- algorithm → used to compute the nearest neighbours.
  - Ball Tree
  - kd-tree
  - brute (default)
  - auto → decides the most appropriate algo based on values passed to fit method.

- leaf_size → default = 30
  - ↳ used only when we are implementing the Ball Tree or Kd Tree algorithm
- p (integer, default value = 2) (Minkowski)
  - ↳ ~~the distance metric to use for the tree.~~
    - When p = 1 → manhattan_distance (l1)
    - When p = 2 → euclidean distance (l2)
- metric :- (default : minkowski)
  - ↳ with p = 2 it is equivalent to standard Euclidean metric
- n_jobs → multiprocessing, default = 1; if we want to use all the cores of the CPU then use value -1.

Weighted KNN

↳ 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.



5 nearest Point	Label	Distance from query point
x_1	B	0.2
x_2	B	0.5
x_3	G	0.7
x_4	G	1.2
x_5	G	1.5

weight

$1/0.2 = 5$

$1/0.5 = 2$

1.4

0.8

0.6

Now we calculate weight: Based on weighing fn

Simple weighing fn is the inverse fn

$$w_i = \frac{1}{d_i}$$

↳ Rule is Distance less, weight less.

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 kare toh Green labelled kate ye query point (see the above Diag.)

Naive Bayes

Conditional prob.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad \text{provided } P(B) \neq 0$$

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 ie ek ke hone se dusre ko farak nahi padta.

Ex:- Rolling a die twice, pehli baar 3 aya toh dusri baar 5 aane ki kya prob hai Ans is 1/6 irrespective of pehli baar kya aya tha

In case of indep Events

$$P(A \cap B) = P(A) \times P(B)$$

& we know $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) \times P(B)}{P(B)} = P(A)$

$\therefore P(A|B) = P(A)$

Sily $P(B|A) = P(B)$

Mutually Exclusive Events:-

A, B are called mutually exclusive events if

$$P(A \cap B) = 0$$

Ex: Ek coin uchala → Head & Tail dono saath mei aaye — not possible.

Also. $P(A|B) = \frac{P(A \cap B)}{P(B)} = 0$

Sily. $P(B|A) = 0$

Bayes Theorem

A, B are two events then

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}, \quad \text{given } P(B) \neq 0$$

$P(A|B)$: Posterior; $P(B|A)$: likelihood; $P(A)$: prior; $P(B)$: Evidence.

Intuition of Naive Bayes

Consider the Dataset of CSK.

toss	venue	outlook	Result
won	Mumbai	overcast	won
lost	Chennai	Sunny	won
won	Kolkata	sunny	won
won	Chennai	sunny	won
lost	Mumbai	sunny	lost
won	Chennai	overcast	Lost
won	Kolkata	overcast	Lost
won	Mumbai	sunny	won.

← input → ← output →

→ given input query we have to predict chennai super kings will loose or win the match.

→ Binary classification problem

Suppose input query is :-

{ Lost, Mumbai, Sunny } → then Naive Bayes will find out P(win) given the above (query) cond^n's & P(loss) given the above (query) cond^n's.

Hence we need to find P(W/A) & P(L/A)

& jiski bhi prob jyada hoi wohi Answer hai Naive Bayes ka.

A = {Loss ∩ mumbai ∩ Sunny}

We know :- $P(A/B) = \frac{P(B/A) \times P(A)}{P(B)}$

so $P(w/A) = \frac{P(A/w) \times P(w)}{P(A)} = \frac{\times 5/8}{1/8}$

A/w.

won mum over	won
Lost Chen Sun	won
won kol Sunn	won
won chen Sunn	won
won mum sunny	won

∴ P(A/w) = 0

Now :- Naive Bayes is using a simple Assumption on the Bayes theorem

& The condition is :- $P(A/w) = P(Loss, mumbai, Sunny / w)$

$= P(Loss/w)\, P(mumbai/w)\, P(Sunny/w)$ (why we see Later)

$= \frac{1}{5} \times \frac{2}{5} \times \frac{4}{5}$

$\therefore P(w/A) = \frac{(1/5 \times 2/5 \times 4/5) \times 5/8}{1/8} = 0.2$

Sly. $P(L/A) = \frac{P(A/L) \times P(L)}{P(A)} = \frac{P(Loss, mum, Sunny/L) \times P(L)}{P(A)}$

using Naive Bayes $= \frac{P(Loss/L) \times P(mum/L) \times P(Sunny/L) \times 3/8}{1/8}$

$= \frac{1/3 \times 1/3 \times 1/3 \times 3/8}{1/8} = 0.3$

Hence $0.3 > 0.2$ ∴ CSK will lost the match)

Mathematics behind Naive Bayes.

$X = \{x_1, x_2, \cdots, x_n\}$ → input cols / input query.

$C_K = \{C_1, C_2, \cdots, C_K\}$ → classes / categories in output col. → multiclass classification (K classes)

we need to find :-

$P(C_i/X)$ $\forall i = 1, 2, \cdots, K$ and jo sabse jyada hoga wohi hum answer mein lenge.

$$P(C_i|X) = \frac{P(X|C_i) \times P(C_i)}{P(X)}$$ (using Bayes theorem)

Note that $\forall i = 1, \cdots K$; the deno for all $P(C_i/X)$ is same $P(X)$ hence it will not help in comparison so we only find the numerator.

so we only need to find $P(X|C_i) \times P(C_i)$ $\forall i = 1, 2 \cdots, K$

we know $P(A|B) = \frac{P(A \cap B)}{P(B)} \Rightarrow P(A \cap B) = P(A|B) \times P(B)$

$\therefore P(X|C_i) \times P(C_i) = P(X \cap C_i) = P(X, C_i)$

$= P(x_1 \cap x_2 \cap \cdots \cap x_n, C_i) = P(\overbrace{x_1}^{A}, \overbrace{x_2, \cdots, x_n, C_i}^{B})$ → $P(A \cap B) = P(A|B) \times P(B)$

$\forall i = 1, 2, \cdots K$

$= P(x_1 | x_2, x_3, \cdots, x_n, C_i) \times P(x_2, \cdots x_n, C_i)$

$= a_1 \times P(\overbrace{x_2}^{A}, \overbrace{\cdots x_n, C_i}^{B})$ a_1 $= P(A \cap B) = P(A|B) \times P(B)$

$= a_1 \times P(x_2 | x_3, \cdots, x_n, C_i) \times P(x_3, x_4, \cdots, x_n, C_i)$

$= a_1 \times a_2 \times P(x_3, x_4, \cdots, x_n, C_i)$ → This breaking is called Chain Rule for cond. prob.

$\vdots$

$= a_1 a_2 \cdots a_{n-1} \times P(x_n / C_i) \times P(C_i)$

$= P(x_1 / x_2, x_3, \cdots x_n, C_i) \times P(x_2 | x_2, x_3, \cdots x_n, C_i) \times \cdots \times$

$P(x_{n-1}, x_n | C_i) \times P(x_n | C_i) \times P(C_i)$

Now we take Naive Assumption

ie. x_j is independent of $x_{j+1}, x_{j+2}, \cdots, x_n$, it only depends upon C_i

And the Assumption is called Conditional Independence

Hence:- $P(X_1 / X_2, \cdots, X_n, C_i) = P(X_1 / C_i)$

Sly. $P(X_2 \mid X_3, \cdots X_n, C_i) = P(X_2 / C_i)$

$\vdots$

$P(X_{n-1} \mid X_n, C_i) = P(X_{n-1} \mid C_i)$

∴ Numq $P(C_i \mid X) = P(X_1/C_i)\, P(X_2/C_i) \cdots P(X_n/C_i) \times P(C_i)$

for $i = 1, \cdots, K$. $= P(C_i) \times \prod_{j=1}^{n} P(X_j / C_i)$

∴ Actual formula is

for $i = 1, 2, \cdots K$

$$P(C_i \mid X) \propto P(C_i) \prod_{j=1}^{n} P(X_j / C_i)$$

Now: jis i ke value ke liye $P(C_i/X)$ will be Largest X usse C_i class mei jaiga.

So $\hat{y} = \underset{i=1,2,\cdots K}{\arg\max}\ P(C_i) \prod_{j=1}^{n} P(X_j / C_i)$ → Maximum a Postirori rule. (MAP)

($\hat{y}$: predicted output)

Jupyter Notebook.

Numerical Data (input) 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 $P(M \mid H=185, W=170) = P(M) \times P(H=185/M) \times P(W=170/M)$

& $P(F \mid H=185, W=170) = P(F) \times P(H=185/F) \times P(W=170/F)$.

using ✓ previous method

So we take Assumption that Height is Gaussian Distributed Random variable.

$P(H=185/M) = 0$ Sly. $P(H=185/F) = 0$

So to find :- $P(H=185/M)$ & $P(W=170/M)$

172	150	M
180	170	M
165	190	M
190	200	M

Step 1 - find μ, σ of this → Assume this is Normally dist

incase of Normal dist we know pdf (prob) $= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ → we will use this for $P(H=185/M)$ by putting $x = 185$

Sly. for weight find μ, σ for 150, 170, 190, 200.

& Sly 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 konsa 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 → Removing html tags, Removing words like and, if, or jinka analysis mei koi fayda nahi hai
2) Algo coill 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.

Bag of words

Review column mei jitne bhi unique words hai unn sabka ek ek col bana dete hai.

Let's say we have only 4 unique words :-

	"Great"	"awesome"	"poor"	"pathetic"	Label/sentiment
Review1	3	1	0	0	1 (pot)
Review2	0	1	2	1	0 (-ve)

Now we pick up Review1 and see in dono mei kitne kitne words aaya hai & kitna nahi in Review1, agar Review1 mei Great 3 baar aya, awesome 1 baar, poor → 0 & pathetic 0 baar then we fill the cols above for row Review1 as above.

& so on we do that for each Review. (50000 Reviews aaya)

↓

Bag of words can be done using the class [CountVectorizer] in the [Sklearn Library]

Suppose the final Dataset is:-

awesome f1	great f2	boring f3	Label
3	1	0	pot (1)
0	0	2	-ve (0)
3	3	1	pot (1)

So what Naive Bayes do:-

$P(\text{label} = \text{pot} \mid f_1=0, f_2=1, f_3=1)$

& $P(\text{label} = -\text{ve} \mid f_1=0, f_2=1, f_3=1)$

↓ we know the formula for Calculating these.

Input Query:

"The movie was (great), but slightly (boring) → [0, 1, 1]

↓

How to predict the sentiment

This is nothing frequency count of the words "awesome", "great", "boring" in the input Review string.

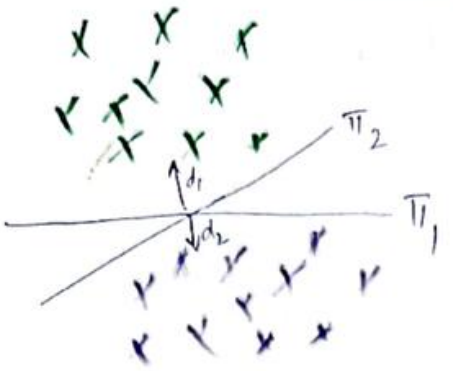
& jo bada hoga vohi Ans hoga.

Jupyter Notebook. (Do it on Kaggle, Jupyter Notebook mei slow hojaiga.

ge 106.

Support Vector Machines

→ It improves the idea of Logistic Regression



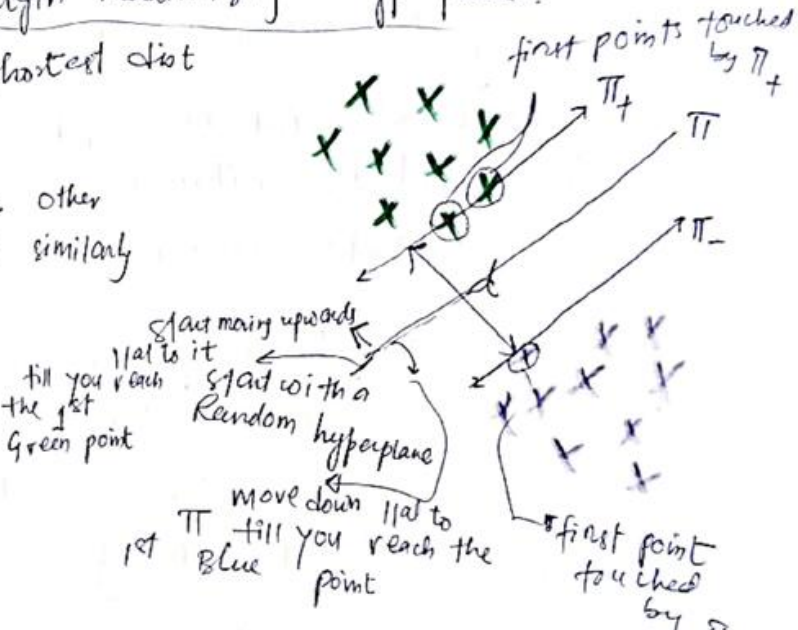
→ Both the planes / Lines π_1 & π_2 are able to classify the Dataset with 100% accuracy. But as a classifier which one is better? SVM chooses the better hyperplane for us.
Ans is: π_1

SVM chooses that Line jo Data ko classify kare & saath hi saath d_1 & d_2 ko maximise kare (for binary classification problem)
∴ The hyperplane is called Margin Maximizing hyperplane.

How to select the Margin Maximizing hyperplane:

Now we find Margin = Shortest dist b/w π_+ & π_- = d_1

Then again we start with some other random hyperplane & proceed similarly & find d_2, then $d_3 \cdots d_n$
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.

$\Pi: w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_0 = 0 \rightsquigarrow w \cdot x + w_0 = 0$

$w = [w_1 \cdots w_n]$

$x = [x_1 \cdots x_n]$

$w^T x + w_0 = 0$

for 2-D. $w_1 x_1 + w_2 x_2 + w_0 = 0 \rightsquigarrow$ line

for 3-D $w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 = 0 \rightsquigarrow$ plane.

$w^T = [w_1 \cdots w_n]$

$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

if $w_0 = 0$ then hyperplane Passes through origin.

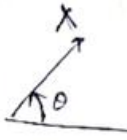
∴ Assuming ki hamara hyperplane origin se pass kare, the eqn is

$\therefore$

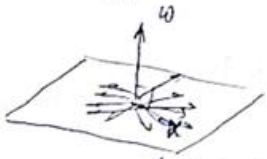
$w^T x = 0$

we know $w^T x = w \cdot x = \|w\| \|x\| \cos\theta = 0$

when $\theta = 90°$.



w is a vector which is $\perp^r$ to the hyperplane : Π.



↳ 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 $\vec{w}$

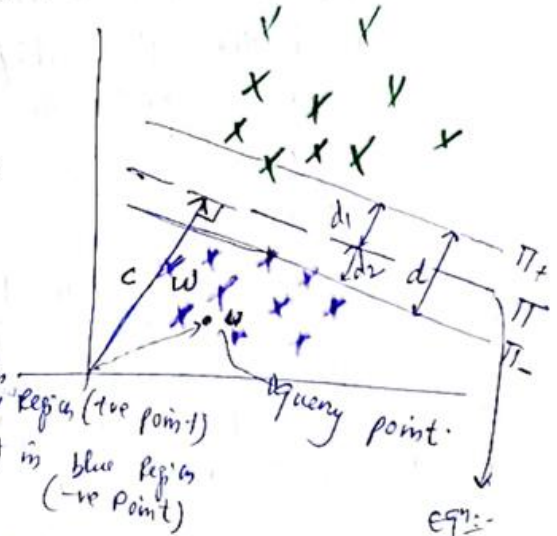
$\vec{u} \cdot \vec{w}$ & we check

$\vec{u} \cdot \vec{w} \geq c \rightarrow$ yes query point in green Region (+ve point)

$\rightarrow$ NO query point in blue Region (−ve Point)

$\vec{w} \cdot \vec{u} - c \geq 0$

$\boxed{\vec{w} \cdot \vec{u} + w_0 \geq 0}$ → Decision Rule.



In General : Eqn of Π : $w_1 x_1 + \cdots + w_n x_n + w_0 = 0$; $w^T x + w_0 = 0$

$\vec{u} = (u_1 \cdots u_n)$ is query point then we check

$w_1 u_1 + \cdots + w_n u_n + w_0 \geq 0$ or ≤ 0 (simple).

is $w^T u + w_0 \geq 0$ or ≤ 0.

so for any $\vec{x_i}$ (query point)

$$\hat{y} = \begin{cases} +1 & \text{if } \vec{w}\cdot\vec{x_i} + w_0 \geq 0 \\ -1 & \text{if } \vec{w}\cdot\vec{x_i} + w_0 \leq 0 \end{cases}$$

Now we need to find the eqn of hyperplane $w^T x + w_0 = 0$ ↓ To find w & w_0.

maximizing d = shortest dist b/w Π_+ & Π_-

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_- : $w^T x + w_0 = -1$

($1, -1$ ko ya $5, -5$ ko ya kuch bhi lo, magnitude same hi rahega & hyperplane same hi aayega).

Π_+ & Π_- mein ye same rahega to Π hai. Parallel

Now Note: LHS of Π_+ & LHS of Π_- ko

- >1 factor se multiply karenge to d shrink (kam hoga)
- <1 factor se multiply karenge to d ↑ se hoga → see in Desmos using 2D (Line) graph

Note: There is some constraints on the Data set:

Π_+ is the ||el shift of Π such that it touches the first green point so no green pts 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 ($\vec{x}$) → $\vec{w}\cdot\vec{x} + w_0 \geq 1$

& for blue points ($\vec{x}$) → $\vec{w}\cdot\vec{x} + w_0 \leq -1$ (equality holds for support vectors)

↓ Now we find d maximising it & selecting the corresp. hyperplane.

Converting the two constraints to a single constraint:-

Let $y_i \to +1$ for pot (green points)

$\to -1$ for -ve (red pts)

∴ The constraint becomes: $\boxed{y_i(\vec{w}\cdot\vec{x_i} + w_0) \geq 1}$ → equality will hold for support vectors

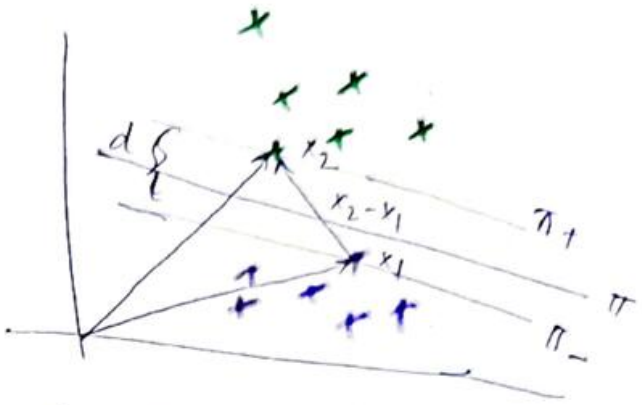
So given the constraint $y_i(\vec{\omega}\cdot\vec{x_i}+\omega_0) \geq 1$ we need to find eqn of hyperplane $\vec{\omega}\cdot\vec{x}+\omega_0=0$ with max d.

and for support vectors :- $y_i(\vec{\omega}\cdot\vec{x}+\omega_0)=1$.

So projection of $\vec{x_2}-\vec{x_1}$ vector on a unit vector $\perp^r$ to π_- or π or π_+ will give us d

& unit vector $\perp^r$ to $\pi = \frac{\vec{\omega}}{\|\omega\|}$

$$\therefore d = (\vec{x_2}-\vec{x_1})\cdot\frac{\vec{\omega}}{\|\omega\|}$$



$x_2 \rightarrow$ support vector on $\pi+$

$x_1 \longrightarrow$ support vector on π_-.

$$d = \frac{\vec{x_2}\cdot\vec{\omega} - \vec{x_1}\cdot\vec{\omega}}{\|\omega\|}$$

So for x_2.

$y_i(\vec{\omega}\cdot\vec{x}+\omega_0)=1$

$1(\vec{\omega}\vec{x_2}+\omega_0)=1$

$\vec{\omega}\vec{x_2} = 1-\omega_0$

for x_1

$y_i(\vec{\omega}\vec{x}+\omega_0)=1$

$-1(\vec{\omega}\cdot\vec{x_1}+\omega_0)=1$

$-\omega_0-1 = \vec{\omega}\cdot\vec{x_1}$

$$d = \frac{(1-\omega_0)-(-\omega_0-1)}{\|\omega\|}$$

$$\boxed{d = \frac{2}{\|\omega\|}}$$

So we need to max d

So we need to minimize $\|\omega\|$

Note again originally.

$\pi_+ : \vec{\omega}\vec{x}+\omega_0 = \alpha$

$\pi_- \text{ & } \vec{\omega}\vec{x}+\omega_0 = -\alpha$ Ajay Nikola

then $d = \frac{2\times\alpha}{\|\omega\|}$ Nikola.

So $\underset{(\omega^*, b_0^*)}{\text{argmax}} \frac{2}{\|\omega\|}$ such that $y_i(\vec{\omega}\vec{x_i}+b) \geq 1$.

↓

This formula we derived when there was perfect seperation b/w the two categories of data. This will not work if the data is mixed up (has impurity)

This type of problem is called Hard Margin SVM.

Soft Margin SVM

What is the data is not perfectly split (ie they are mixed up) → not perfectly linearly seperable.

Then the problem cannot be solved using Hard Margin SVM.

→ Generally we will get data such that ek ya do points idher ka udhar & udhar ka idher honge. (outliers) so we generally use soft margin SVM.

In Hardmargin SVM we had

$$\underset{(w^*, w_0^*)}{\arg\max} \frac{2}{\|w\|} \quad s.t. \quad y_i(\vec{w}.\vec{x_i} + w_0) \geq 1.$$

Also we know $\max f(x) \iff \min \frac{1}{f(x)}$

So $\underset{(w^*, w_0^*)}{\arg\min} \frac{\|w\|}{2} \quad s.t. \quad y_i(\vec{w}.\vec{x_i} + w_0) \geq 1$

$$\Downarrow$$

$$y_i(w^T x_i + w_0) \geq 1$$

In Softmargin SVM:- we have:- (Regularization term (by comp. with Logistic Regression))

$$\underset{(w^*, w_0^*)}{\arg\min} \left(\frac{\|w\|}{2} + C \sum_{i=1}^{n} \xi_i \right)$$

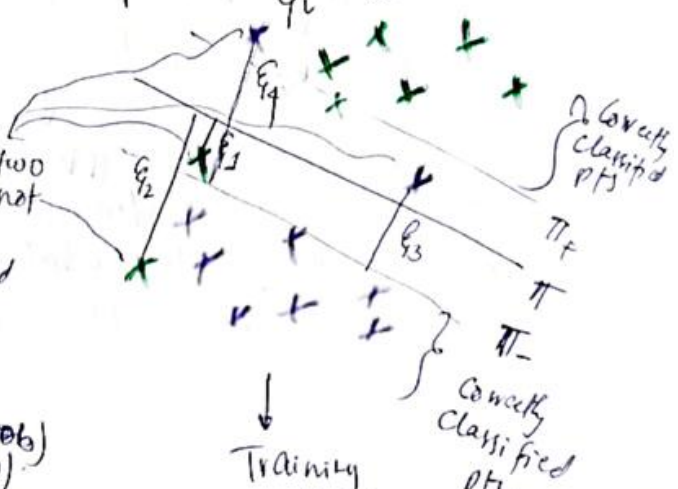
→ hinge loss

(n = # of rows/datapoints in the dataset).

ek pot (green) point galat classified hai toh uska ξ_i is uska $\perp^r$ dist for Π_+ hyperplane & ek -ve (Blue) point galat classified hai toh uska ξ_i is uska $\perp^r$ dist for Π_- hyperplane.

ξ_i is calculated for every x_i (row/datapoint) in the dataset.

if the point is correctly classified then for that point $\xi_i = 0$.



and minimising $\sum_{i=1}^{n} \xi_i$ means minimizing the Error (galat classified points)

So $\frac{\|w\|}{2}$ is called Margin Error → minimizing it reduces margin error.

& $\sum_{i=1}^{n} \xi_i$ is called classification Error.

& $C(\vec{\alpha})$ is a hyperparameter (tuning knob) ($\alpha = 1/\lambda$, λ from Logistic Regression)

if C is big → means we are not caring about margin Error, we only want ki algo koi bhi point ko misclassify na karne.

and if C is vv small → focus on margin Error & ignore classification Error matlab much misclassifications chalega.

Jupyter Notebook