Machine learning

it is a branch of AI that develops the algorithm by learning the hidden patterns of the data set used to make prediction on new data points

we can divide machine learning in 3 parts →

1. supervised learning
2. Unsupervised learning
3. Reinforcement learning

1) Supervised learning

1) supervised learning model is trained on labled data.

label data → input data + Output data.

2) The differentiation is done on the basic of the output data.

3) It has Two main categories → A) Regression (contineous data 22.5, 25.3, 26.5)

B) Classification (discrete data 1, 2, 3, yes, no)

A) Regression → 1) it deals with contineous target variable like (22.5, 25.6, 28.9, 30.1)

2) e.g. House price prediction based on area, size

B) Classification → 1) It deals with categorical/discreate target variable like (yes, No, 1, 2, 3) whole data

e.g. email spam or not
loan approved or not.

Note → 1) If targeted variable has contineous data then use regression algorithms.

2) If targeted variable contains discrete or categorical data then use classification Algorithm

Regression Algorithm	Classification Algorithm
1) linear Regression	1) logistic Regression
2) Polynomial Regression	2) Support Vector Machine
3) Decision Tree	3) decision tree
4) Random forest.	4) k-nearest neighbours
	5) Neive Bayes

2) Unsupervised Machine learning

1) Unsupervised learning model is trained on unlabled data set
2) Unlabled data set → only input data (No target Variable)
3) This Differentiation is done on the basic of the output data.
4) It has two main categories → 1) clustring
   2) Accociation

A) **Clustring** → In these process, there are different group are created
it is a process of grouping data points into cluster based on similarity.

B) **Association** → It is a technique that is used for descovring relationship between items of dataset.

Clustring Algorithm	Association Algorithm
1) k-mean clustring algorithm	1) frequent Pattern (FP) groth algorithm.
2) PCA - (Principal Component analysis)	
application	
1) Dimension Reduction	
2) Recomdation System	
3) image & video compression	

3) Reinforcement

It is based on feedback and Punishment

Algorithm →

1) Q learning
2) Deep-Q-Network
3) Policy Gradient Methods
4) Actor Critic Method
5) Proximal Policy Optimization
6) Trust Region Policy Optimization
7) Monte Carlo Tree Search

Application →

1) Gaming
   AI in bord games
2) Autonomous Vehicles
3) Robotics
4) Natural language Processing.

Supervised	Unsupervised
1) labled Data	1) Unlabled Data.
2) it is used for prediction (Predict Output)	2) it is used for finding Hidden pattern in data.
3) Input data is provided to the model along with output data.	3) only Input data is provided to the model.
4) It need supervision to train the model	4) It does not need any supervision to train the model.
5) Supervised is Categorized into A) Regression B) Classification	5) Unsupervised is classified into A) clustring B) Association
6) It takes direct feedback to cheak if it is predicting Correct output or not	6) It does not take any feedback
7) It includes various algorithms like linear, logistic, SVM, decision tree	7) It includes various algorithms like k-mean clustring, PCA

Regression

Different types of Regression are as →

1) ~~simple~~ linear Regression
2) ~~Multiple linear Regression~~
2) Polynomial linear Regression

(1) ~~simple~~ linear Regression

This is used to predict the value of a variable based on the value of another variable (same definition of Regression)

The variable we want to predict → Dependant

all other variables → Independant

This is based on a best fit line drawn on x, y plot.

Eq of line

$y = mx + c$

slope

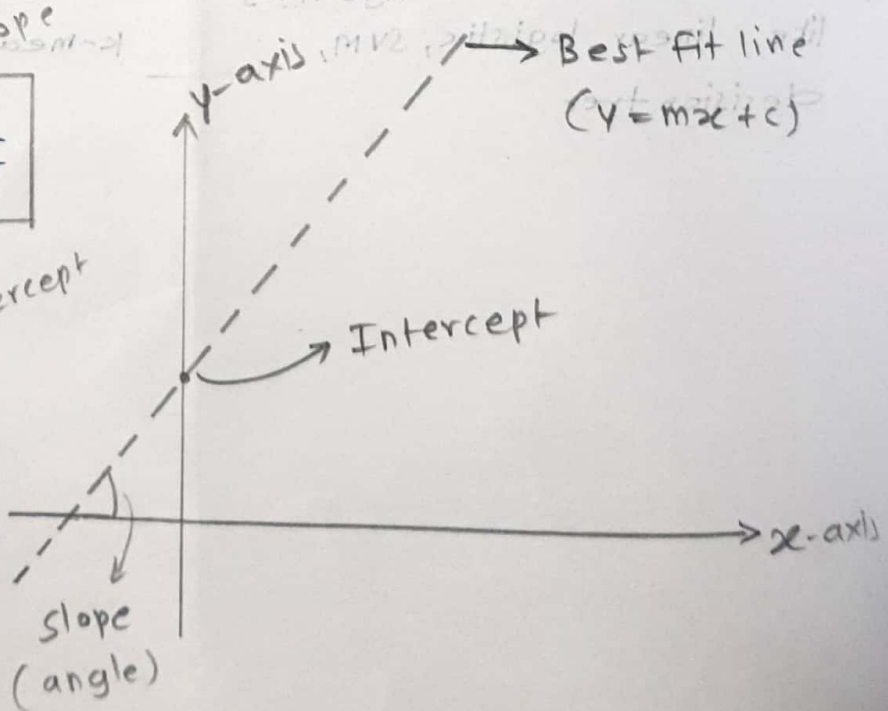
Intercept

m = slope

c = intercept

x = Independant

y = dependant.



Intercept ⟶ on y axis which point it touch that point is Intercept point.

slope ⟶ The angle created by that line with x-axis is slope.

$$\text{Slope}_1 = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2}$$

$$\text{Intercept} = \bar{y} - m\bar{x}$$

Relation in linear Regression is as

Independent ↑ then dependent ↑

Independent ↓ then dependent ↓

Two types in linear Regression

- simple linear Regression
- Multiple linear Regression

1) simple linear ⟶ ~~one~~ 1 Independent 1 dependent

2) Multiple linear ⟶ 1 Independent mutiple dependent.

we need to calculate the Best fit line with line eqⁿ

Best fit line ⟶ The line that has lease error which means the error betⁿ predicted value & actual value should be minimum.

Residual (error) ⟶ The difference betⁿ observed value & predicted value is called Residual.

$$\text{Residual} = Y_{pred} - Y_i$$

Y_{pred} = Predicted value

Y_i = actual value

Y_{pred} in simple linear Regression

$$Y_{pred} = \theta_0 + \theta_1 x$$

θ_0 = Intercept

θ_1 = Slope

x = Independant

Y_{pred} in Multiple linear Regression

$$Y_{pred} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n$$

Y_{pred} in Polynomial Regression

$$Y_{pred} = \theta_0 + \theta_1 x_1 + \theta_2 x^2 + \theta_3 x^3$$

Cost function

The function that we are trying to minimize it helps to find out the optimal values of θ_1 & θ_0

we are using three different cost function in linear Regression.

1) Mean Squared Error
2) R square
3) Root mean squared Error
4) Residual standard Error.

1) Mean Squared Error (MSE)

$$MSE = \frac{1}{N} \Sigma (y_{pred} - y_i)^2$$

2) Root mean squared Error (RMSE)

$$RMSE = \sqrt{MSE} = \sqrt{\frac{RSS}{n}} = \sqrt{\frac{\Sigma (y_i - y_{pred})^2}{n}}$$

It specifies the absolute fit of the model to the data that is How close the observed value points are to the predicted values.

5) Mean Absolute Error (MAE)

$$MAE = \frac{1}{N} \Sigma |y\_Pred - y1|$$

3) R squard (R^2)

1) It is the number that explain the amount of variation that is explained by the developed model.

2) always lies between 0 to 1

3) Higher the value of R^2 that means that better the model fits for data.

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\Sigma(Y_i - Y_{pred})^2}{\Sigma(Y_i - \bar{Y})^2}$$

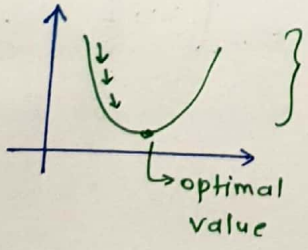
RSS = Residitial Sum Square
$= \Sigma(Y_i - Y_{pred})^2$

TSS = Total sum square
$= \Sigma(Y_i - \bar{Y})^2$

4) Residual Standard Error (RSE)

$$RSE = \sqrt{\frac{RSS}{n-2}} = \sqrt{\frac{\Sigma(Y_i - Y_{pred})^2}{n-2}}$$

learning rate → How fast algorithm reach to final state
It decides How fast the algo coverge to minimum value.



gradient Decent

In gradient decent
No of steps can be considered as learning rate

overshoot = when we have high learning rate

EDA

EDA ⟶ Exploratry Data Analysis.

The data we collect from different sources is not in good form so we need to clean that data and set all the parameters in one scale so our machine can understand that data.

1) Cheak All the data which is as per data type.
   e.g. Column holding Numbers but data type is object so like this situation need to be solve `df[col].astype(Data Type)`

2) cheak the Null values `df.isnull().sum()`

3) Cheak Duplicate values `df.duplicated().sum()`

4) find outliers and remove them (use boxplot)
   find the index of that outlier and then drop it from table
   end fence all are outlier Identify them from box plot and drop them

Count Plot ⟶ categorical Data
Dist plot ⟶ Numerical pata

5) Convert the categorical Data into Numerical Data with encoding.

6) Then cheak the distrubution of data point if there skewness is right or left then use skewness and convert Data into Normal Skew. (Pandas)

7) If Unit of data columns are different then use scaling to convert them in one scale (Sklearn)

* Encoding *

Convert All categorical Data in Numerical Data.

from sklearn.preprocesing
Pd.get_dumies
} library

1) label encoding
2) get dumies

1) label encoding → 1) from sklearn. preprocessing import LabelEncoder

2) le = LabelEncoder ()

3) le.fit_transform (col)

1) Convert Categorical Data into numerical data (Nominal Data)
2) Starts from 0 & gives the number upto the total categories.
3) red, grre, blue three categories so encoding will be like red = 0, green = 1, blue = 2
4) No new columns will be created.

2) get dummies → 1) Pd. get_dumies (col, dtype = int)

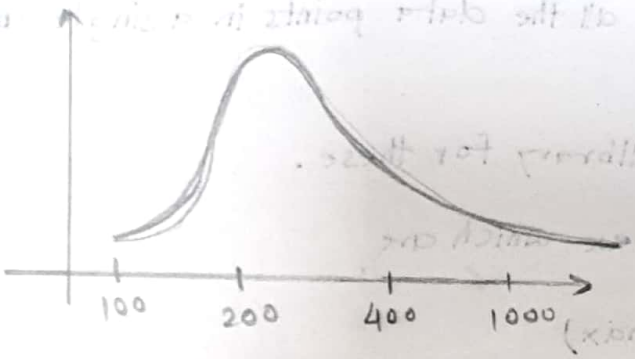
for every category there will be a new column which has only two values which are 0 & 1

Encoding is based on target variable

If target variable is categorical → perform EDA befor train test split.

If target variable is not categorical →

* Skewness *



In these data points have huge distance between them so the curve will be Not good or it could be left or Right skewed

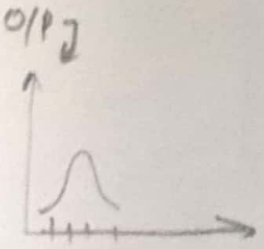
To solve this problem we perform skewness so the distance between these points are minimized and a normal curved could Be form.

Two types → 1) sqert → np.sqrt
2) log → np.log
} present in Numpy library.

1) sqrt → takes the square root of that valve and place it on x axis.

2) log → takes the log of that value and place it on x-axis

→ Done for every value present in the column



* Scaling *

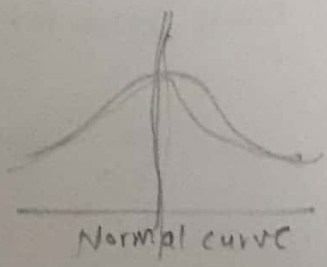
1) Machine don't understand units
2) Sometime 1M = 100 cm this is not ~~dif~~ identifyed by machine and it assume both as different value
3) So to solve these problem we perform scaling
4) In these we will convert all the data points in a single same unit
5) we are using sklearn library for these.
6) There are two types in these which are
   1) Normalization (min max)
   2) Standardization

N → from sklearn.preprocessing import MinMaxScalar

S → from sklearn.preprocessing import StandardScaler

1) Normalization → 1) adject the values of data and fit it within a range
   2) range is most the time [0,1]

$$\text{Normalization} = \frac{X - X_{min}}{X_{max} - X_{min}}$$



X = original value

Xmin = minimum value from dataset.

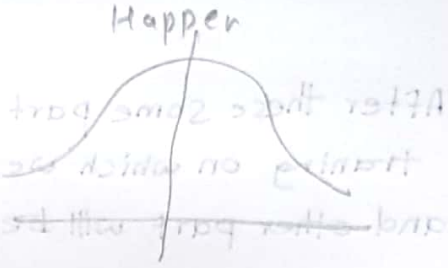
Xmax = maximum value from dataset.

mean = median = mode

2) Standardization →

std deviation = 1

mean = 0
std = 1
} Standard Normal Distrubution



diff { mean = 0
median = 0
mode = 0

Train Test Split

used in every model ← >500 data point

1) Now Data is clean or after geting clean data we need to seprate that Data into 2 parts which are independent & dependent.

2) After these some part of these Data will be used for traning on which we will build our model and other part will be used for testing.

3) if these is not done and all the data is used for traning then the accuracy for prediction could be 100% or it can face overfiting Because of that reason we will divide the data.

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.20,
                                                    random_state=42)
```

Cross Validation

500< data point

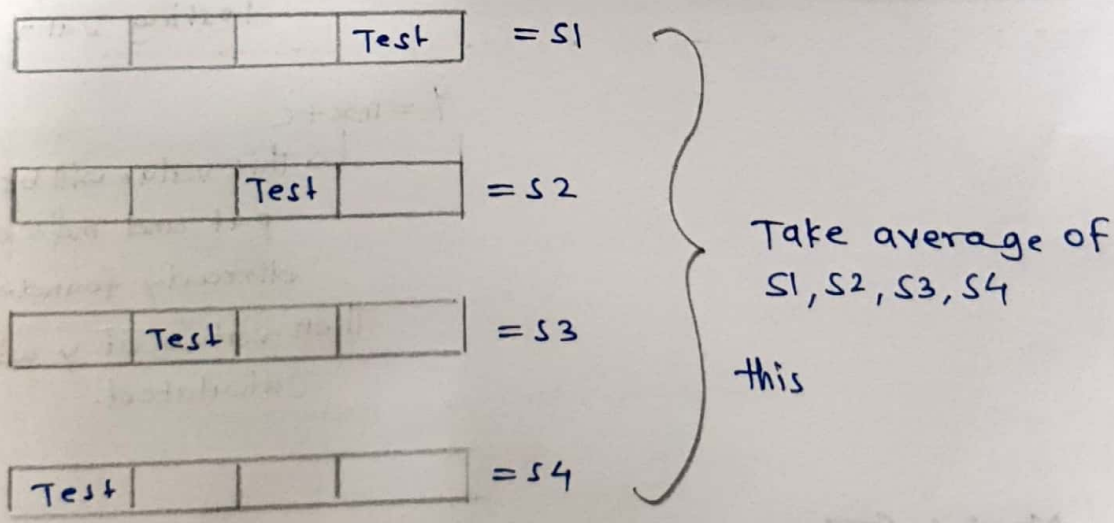
1) Some times data is not that huge or in small scale
2) So at that time we could not use train test split because some categories could be missed in this or some pattern will be undescroved
3) So at that time we will use k-fold validation

process →

default k = 3

1) first take value of k
   k = 4
2) Now the hole dataset will be divided into 4 equal parts
3) Now from these 4 parts → 3 used for traning
   1 used for testing
4) Step 3 will be repeated 4 time

Number of part = that much iteration.



Model Building

```
from sklearn.linear_model import LinearRegression
```

```
Model = LinearRegression()
```

→ $y = mx + c$

All value will be found with this

```
Model.fit(x_train, y_train)
```

→ 2 dimensional value needed.

→ only takes the traning data

To see the values:
- model.intercept_
- model.coef_

$y = mx + c$ → These 2 value will be found with .fit

```
y_pred = Model.Predict(X_test)
```

→ only provide testing Data

$y = mx + c$

→ This value will be put and m & c are already founded. Then value of y will be calculated.

~~Model.Score~~

~~Model.Score(X_test, Y_test)~~

```
Model.Score (X_test, Y_test)
Model.Score (X_train, Y_train)
```

→ The testing score

→ The traning score (Not that needed.)

find Error

MSE

```
from sklearn.metrics import mean_squard_error
mean_squard_Error (Y_test, Y_Pred)
```

→ The error is found beth actual value & predected value

actual value

predicted value.

RMSE

```
mSE = mean_squard_error (Y_test, Y_Pred)
np.sqrt (mSE)
```

R^2 score

```
from sklearn.metrics import r2_score
r2_score (Y_test, Y_Pred)
```

Rules Need to be follow while creating a ML Model

(for linear Regression)

1) There must be linear Relationship bet^n independant & dependant variable.
2) All variable are normally distrubuted.
3) There must be no outliers.
4) The mean of distrubution of error is zero
5) The variance of error is constant across all the level of independant variables. (Homocidity)
6) The distrubution of error is normal & all errors are independant.

Corelation →

lies between 0 to 1

1 is high

feature selection → find corelation bet^n independent if any independent variable has more corelation with other independent variable then remove

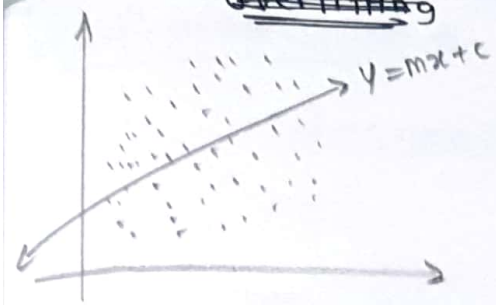
independent ← corelation → Independent

↳ Then remove any one lease imp.

dependent ← corelation → Independant

↳ good.

~~Overfitting~~



In this situation **Underfitting**

MSE is very high

R^2 score is very low

Because of that reason Underfitting is occured.

> Main Reason ⟶ low complexity
>
> traning score is low

Overfitting

~~Underfitting~~

In this situation

MSE is very low ⟶ travel to every point

R^2 score is very high

Because of that Reason Overfitting is occured

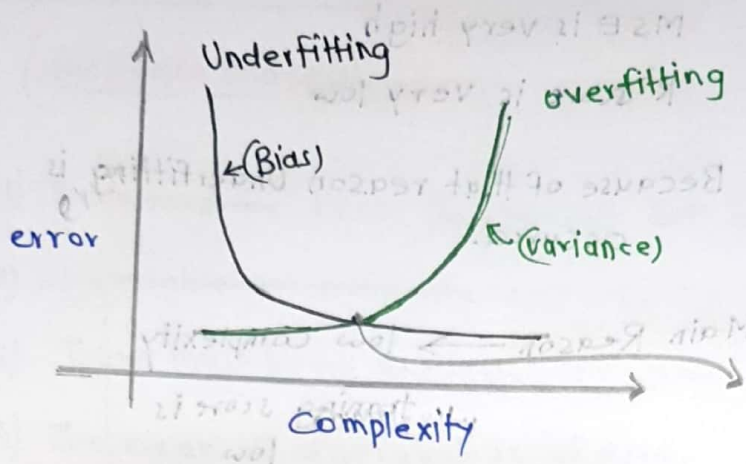
> Main Reason ⟶ High complexity
>
> test scror > traning score.

Complexity

$$\text{complexity} = \frac{\text{No of features}}{\text{No of independent col}}$$

- Complexity ↑ error ↓ (Underfitting)
- Complexity ↓ error ↑ (overfitting)

* Trade off (trade Variance) *



Need to find the features which has low complexity

As we increase complexity the error is decreased.

1) If we increase the complexity alot then the error will again start gradually increase.

2) The error in underfitting situation is called Bias As we increase the complexity the Bias will be decrease.

3) In overfitting situation the error start increase then that is known as variance

4) If we increase complexity then variance is also increased.

* Identify Underfiting *

1) Take data
2) Split it into traning & testing data
3) train ~~data~~ model on traning data
4) find traning score on traning data
5) if score is not good then underfiting

> 90 → good
=> 80 → avg
< 70 → bad

* Sol^n Underfitting *

Increase the Number of features.

* Identify Overfiting *

Same still step 4

5) find traning score for traning data
6) find testing score for testing data.
7) If traning score > testing score
8) If true then overfitting.

* Sol^n Overfiting *

Decrease Number of feature
Add some extra error
Two methods present for these.

1) L1 Regularization (Lasso)
2) L2 Regularization (Ridge)

we are addiing extrnal error in traning.

1) MSE + error then line shift toward the testing data & give us the best fit line

2) As we are adding very small amount of extrnal error in traning MSE which helps us to reduce the testing error.

external error + traning MSE = Regularization

1) L1 Regularization (Lasso)

$$\text{Lasso} = \underbrace{\frac{\Sigma(y - y_{pred})}{n}}_{\text{MSE}} + \underbrace{\lambda \Sigma |\omega|}_{\text{Error}}$$

ω = Coefficient

λ = hyperparameter

direct o the unwanted

It will minimize the coefficient of unwanted feature to o

2) L2 Regularization (Ridge)

$$\text{Ridge} = \underbrace{\frac{\Sigma(y - y_{pred})^2}{n}}_{\text{MSE}} + \underbrace{\lambda \Sigma \omega^2}_{\text{Error}}$$

direct toward's to the o

Ridge, Lasso ⟶ add extra error
remove unwanted features.

λ ⟶ decide by developer
it can not take negative

छोटी value होतो → Ridge $\lambda (\omega^2)$

मोठी value होतो → Lasso $\lambda (|\omega|)$

Ridge → approch near to zero λ = low

Lasso → Make direct 0 λ = high

always take testing data for Ridge & Lasso

Delet Unwanted feature + Add some error = overfiting Handling

Mostly use the Lasso → Because it directly make column value 0 so directly deleted.

Bad model → accuracy below or near 80%. then it is ok ok but not that good.

accuracy below 70%. then bad model.

Commands

Convert Categorical into Numerical	1) label Encoding	from sklearn.preprocessing import LabelEncoder le = LabelEncoder() le.fit_transform(col)
	2) get dummies	import pandas as pd pd.get_dumies(____, dtype = int)
Skewness	1) sqrt	b = np.sqrt(Numpy array) b = pd.dataframe(b) → convert it into 2D after that to put in next operation.
	2) log	b = np.log(Numpy array) b = Pd.Dataframe(b) → convert
Scaling	1) Normalization (MinMax)	from Sklearn.preprocessing import MinMaxScaler min_max = MinMaxScaler() min_max.fit_transform(col)

	2) Standardization	from sklearn.preprocessing import StandardScaler scaler = StandardScaler() scaler.~~min~~ fit_transform(col) Data points distribution is std-Normal ie mean = 0 & std = 1
Train Test Split data > 500		from sklearn.model_selection import train_test_split X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= , random_state=)
Cross Validation data < 500		from sklearn.model_selection import cross_val_score cv = cross_val_score(lr, x, y, cv = 5) ↳ No of folds for which model we are using obj of that model
Model Building and Prediction	1) import Model	from sklearn.linear_model import LinearRegression lr = LinearRegression() → for simple and Multiple regression from sklearn.preprocising import Polynomialfeature poly = Polynomialfeature(degree = 2) ↳ for polynomial regression import these.

	2) .fit slope (m) intercept(c)	lr.fit(X_train, Y_train) Poly_x = poly.fit_transform(X) ↳ independent data lr.fit(Poly_x, Y_train) ↳ what to put Here.
	3) .predict put x in formula & find Y $y = mx + c$	Y_Pred = lr.predict(X_test) Y_Pred = lr.predict(Poly_x)
	4) .score only cheak for regression for classfication use classification report.	lr.score(X_test, Y_test) → Testing score ↳ simple, Multiple lr.score(Poly_x, Y) ↳ Polynomial
Error	1) MSE	from sklearn.metrics import mean_squard_Error mean_squared_Error(Y, Y_Pred) actual data point predicted data point.

	2) MAE	from sklearn.metrics import mean_absolute_error mean_absolute_error(y, y_pred)
	3) R^2	from sklearn.metrics import r2_score r2_score(y, y_pred)
Regularization (Ridge L2, lasso L1)	Ridge (L2)	from sklearn.linear_model import Ridge L2 = Ridge(alpha=2) L2.fit(X_train, Y_train)
	Lasso (L1)	from sklearn.linear_model import Lasso L1 = Lasso(alpha = 150) L1.fit(X_train, Y_train)
Classification Report (only for classification)	Accuracy precision Recall F1 score	from sklearn.metrics important classification_report classification_report(y_test, y_pred)

Classification

If targeted variable has discrete or contineous data then it is a classification problem.

e.g. patient has cancer or not
fraud detection
email spam
signature is real or fake.

Different Types of classification are →

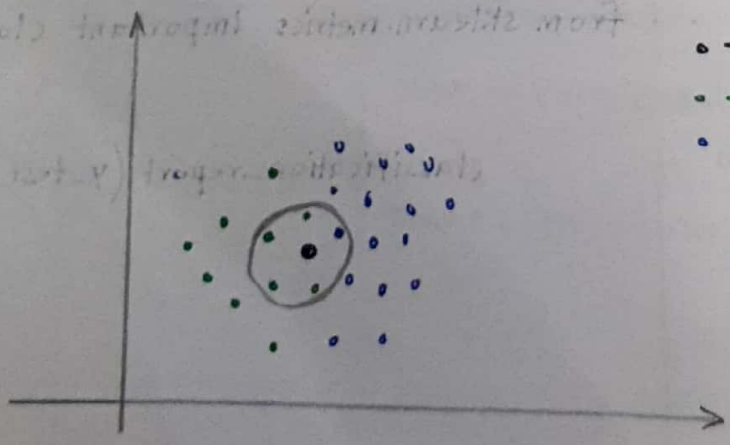
1) k-neart neighbour
2) Logistic Regression
3) Decision tree
4) k-~~te~~ Random forest
5) Support Vector Machine.

Matrix in classification

1) Accuracy
2) Recall
3) Precision
4) f1-score

1) k-nearst neighbour

It is baced on distance.



• → we are predicting for
• → Yes
• → No

k=5 → other than the predicting one it choose 5 data points which are most near to it

Now Here 4 are yes and 1 is No
so prediction will be yes

default value of $k = 5$

*ways to find minimum Distance *

for the nearst 5 data points we can find distance through 2 ways

1) Euclidean distance
2) Manthan distance

1) Euclidean distance

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

It is simple distance and all the distance of nearst points are found and then minimum from a them taken.

2) Manthan Distance

$$|X_2 - X_1| + |Y_2 - Y_1|$$

Confusion Matrix

predicted \ Actual	1	0
1	TP	FP
0	FN	TN

0 ≠ 1

So this is false

So that row is false Negative

Predict	Actual
P	T
N	f

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Sometime FN could be critical

At that time we can not use the accuracy

Because result of accuracy is always positive

So in these state we will use recall

man has cancer or Not

↳ so in this FN is critical

Because man has cancer and machine predicts it does not has cancer.

So for FN use recall

$$\text{Recall} = \frac{TP}{TP + FN}$$

If recall is very high then model is good.

Sometime fp could be critical
so at these time we will use precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

email is spam or not
↳ Here fp is critical
Because email was not spam but machine predict that it is spam.
So use precision

Sometime fp & fN both are critical
so at these time we will use
the f1 score.

Share market will crash or not
↳ it will not but machine predict it will
it will be crash but machine predict it will not
Both are harmful for us

f1 = Harmonic mean of Precision & recall

$$f1 = \frac{P \cdot R}{P + R}$$

fN critical	⟶	Recall
fp critical	⟶	Precision
Both critical	⟶	f1 score

KNN Codes

perform all action of EDA and make data clean.

Vse train test split

1) from sklearn.neighbours import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5)

↳ No of near points need to take

2) knn.fit(X_train, Y_train)

3) Y_Pred = knn.predict(X_test)

4) from sklearn.metrics import classification_report

classification_report(Y_test, Y_pred)

2) Logistic Regression

$$\text{Probability} = \frac{\text{No of occurance}}{\text{Total No of value}}$$

$$\text{odd ratio} = \frac{\text{Number of Success}}{\text{Number of failure}}$$

probability value is always positive
it always lies in 0 to 1

$$0 < \text{Probability} < 1$$

$$P = 0 \longrightarrow \frac{1}{2} \longrightarrow 1 \quad \text{probality}$$

$$OR = 0 \longrightarrow 1 \longrightarrow \infty \quad \text{odd ration}$$

$$\text{log odd} = -\infty \longrightarrow 0 \longrightarrow \infty \quad \text{log odd}$$

} lies between

$$\text{Prob success} = \frac{\text{No of success}}{N}$$

$$\text{Prob failure} = \frac{\text{No of failure}}{N}$$

from above formula we can find Number of sucess & failure

No of Succes = Prob of success x N

No of failure = Prob of failure x N

let

Prob success = p

Prob failure = q

let odd ration of them are as (Total No of them)

$$\text{Odd ratio} = \frac{n \times p}{n \times q} = \frac{p}{q}$$

$$\therefore \quad 1 = \frac{p}{q} \qquad p + q = 1$$

$$\boxed{q = 1 - p}$$

$$\text{odd ratio} = \frac{p}{q} = \frac{p}{1-p}$$

$$\text{odd ratio} = \frac{p}{1-p}$$

odd Ratio. $\quad 0 \leq \frac{p}{1-p} \leq 1$

log (odd Ratio) $\Rightarrow$ $(-\infty, \infty)$

line Range $\Rightarrow$ $(-\infty, \infty)$

$\therefore \quad y = mx + c = \log\left(\frac{p}{1-p}\right)$

$y = mx + c$

$y = \theta_0 + \theta_1 x$ } Range is $-\infty$ to ∞

odd ratio → Range is $-\infty$ to ∞

∴ So we can say

line of eqⁿ = odd ratio

$$y = \log\left(\frac{p}{1-p}\right)$$

$$e^y = e^{\log_e\left(\frac{p}{1-p}\right)}$$

$$\frac{e^y}{1} = \frac{p}{1-p}$$

$$\frac{1}{e^y} = \frac{1-p}{p}$$

$$\frac{1}{e^y} = \frac{1}{p} - \frac{p}{p}$$

$$\frac{1}{e^y} = \frac{1}{p} - 1$$

$$\frac{1}{e^y} + 1 = \frac{1}{p}$$

$$\frac{1}{p} = \frac{1+e^y}{e^y}$$

$$P = \frac{e^Y}{1+e^Y}$$

$$P = \frac{e^Y/e^Y}{\frac{1+e^Y}{e^Y}}$$

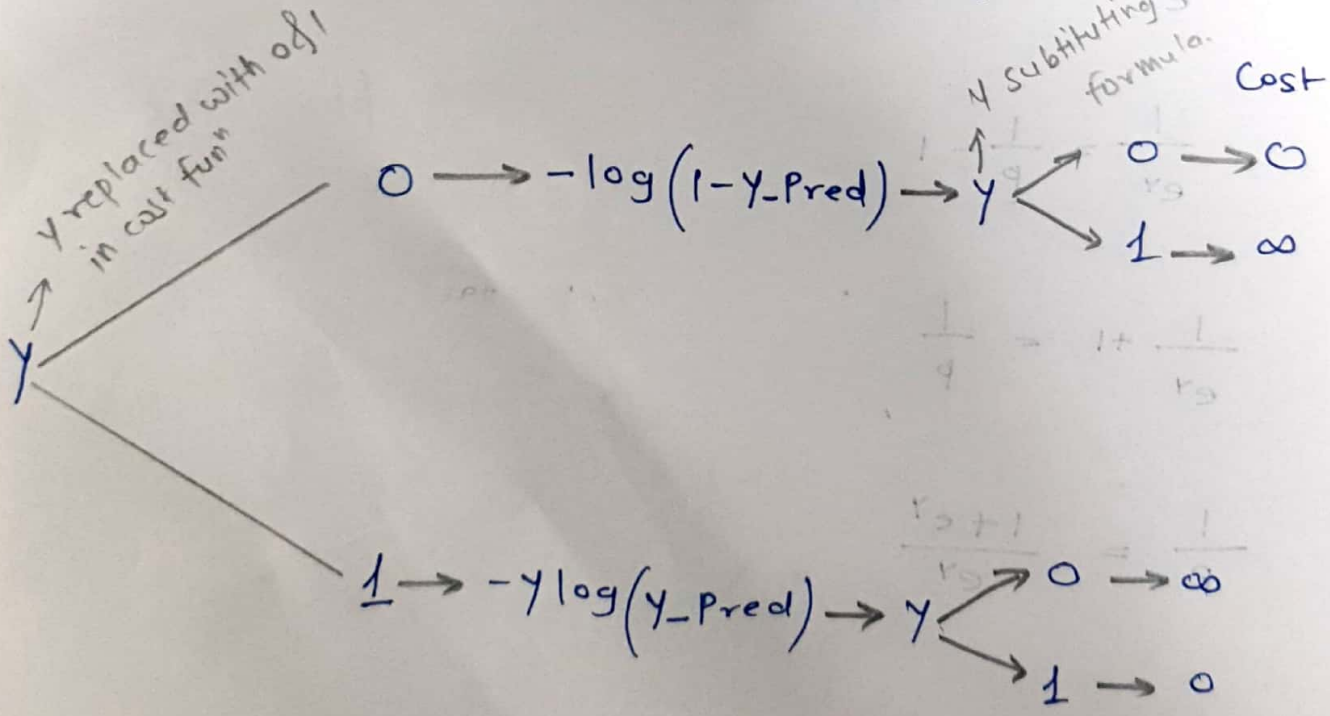
$$P = \frac{1}{e^{-Y}+1}$$

$$P = \frac{1}{1+e^{-Y}}$$ → logic function

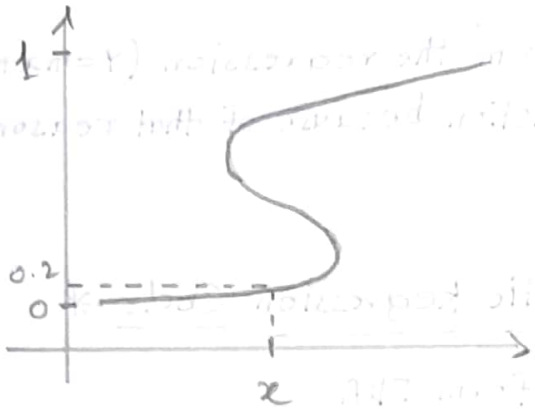
↳ sigmoid function

Cost function $= Y * \log(Y\_Pred) - (1-Y) * \log(1-Y\_Pred)$

So value of Y colube be 0/1



* Thresh-hold *



So here we are predicting x
~~x is 0 or 1~~ and it lies ~~between~~ at 0.2

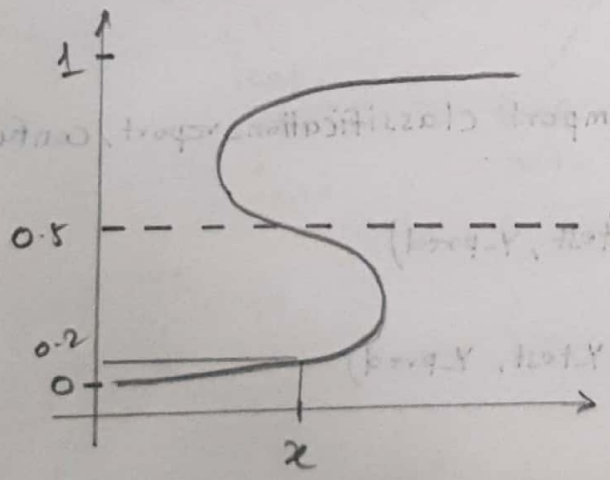
But our model only knows 0 & 1

So for that we will create a threshold
it lies at 0.5 → default

all the values below 0.5 threshhold are 0

all the values above 0.5 threshhold are 1

default threshhold = 0.5



0.2 < 0.5
so it will be 0
∴ x will be 0

Why Regression Name Is Present in logistic Regression

it starts the algorithm from the regression ($Y=mx+c$) and pass that line in sigmoid function because of that reason it is named logistic Regression.

* Logistic Regression Code *

Perform EDA
- Scaling
- Skewness
- Encoding

train test split

1)
```
from sklearn.linear_Model import LogisticRegression
model = LogisticRegression()
```

2) `model.fit(x_train, Y_train)`

3) `Y_pred = model.predict(x_test)`

`model.predic_proba(x_test)[:,1]` → To cheak all probality score

4) `from sklearn.metrics import classification_report, confusion_metrix`

`classification_report(y_test, y_pred)`

`cm = confusion_metrix(y_test, y_pred)`

5) `from sklearn.metrics import recall_score, precision_score`

`recall_score(y_test, y_pred)`

`precision_score(y_test, y_pred)`

3) Decision Tree

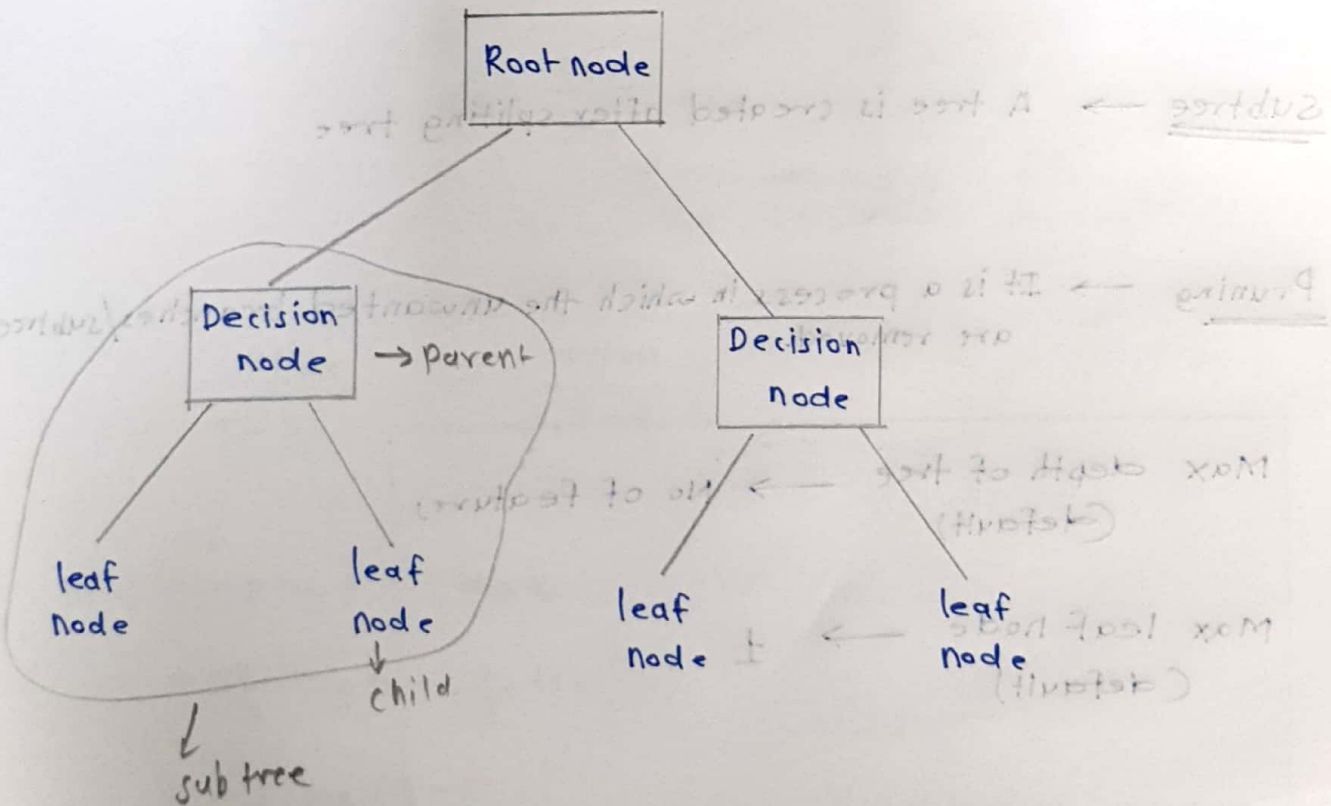
It is a superwised machine learning algorithm used for both classification & regression problem. But Most of time we use for classification.

In decision tree there are mainly two nodes.

1) leaf node
2) decision node

decision node are used to make any decision and it has many branches.

leaf node is output of those decision & does not has further branches.



There are two main node ⟶ parent node,
child node.

1) root node is called as parent node
2) Decision tree start from that root node
3) root node represent the entire data set.
4) which further get divide in two or more set.
5) other than root node all other are child node
6) leaf node are final output of the node

Spiliting ⟶ It is a process to divide the root node or decision node into sub nodes according to the given condition.

Subtree ⟶ A tree is created after spliting tree

Pruning ⟶ It is a process in which the unwanted branches/subtree are removed.

Max depth of tree (default) ⟶ No of features

Max leaf node (default) ⟶ 1

Ginni impurity and entropy Both are used to cheak impurity of Decision tree.

*GINI Impurity *

$$\text{Gini} = 1 - \Sigma(p)^2$$

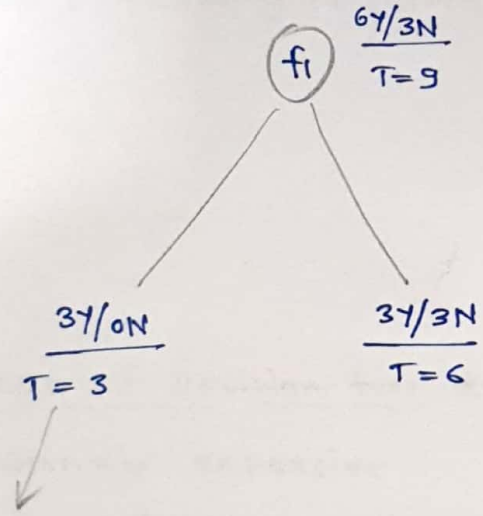
Σ(+veprob + -veprob)

* Entropy *

$$-p \log p - q \log q$$

p = positive Probability

q = Neg



$$H(C_1) = -p \log p - q \log q$$

$$= \frac{-3}{3} \log p - \frac{0}{3} \log\left(\frac{0}{3}\right)$$

$$= -1 \log(1) - 0$$

$H(C_1) = 0 \longrightarrow$ It is pure split

$\therefore$ it is a leaf node

$$H(C_2) = \frac{-3}{6}\log\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log\left(\frac{3}{6}\right)$$

$$= -\frac{1}{2}\log\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log\left(\frac{1}{2}\right)$$

$$= -\log\left(\frac{1}{2}\right)$$

$$= \log_2(2)$$

$H(C_2) = 1 \longrightarrow$ Impure split so it is not leaf.

$$GI(a) = 1 - \Sigma(p^2)$$

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

$$= 1 - [1+0]$$

$$= 1 - 1$$

$GI(C_1) = 0 \longrightarrow$ Pure split

$$GI(C_2) = 1 - \Sigma(p^2)$$

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

$$= 1 - \left[\frac{1}{4} + \frac{1}{4}\right]$$

$$= 1 - \frac{1}{2}$$

$$= \frac{1}{2}$$

$GI(C_2) = 0.5 \longrightarrow$ Impure

Gini impurity lie betn 0 to 0.5
entropy lies between 0 to 1

All the bad features of tree are present at the end
(overfitting)

to Remove that features we use pruning.

Decision tree has more chances of overfitting.

* Advantage of Decision Tree

1) able to classify non linear seprable data.
2) automatically select good feature.
3) less pre processing required (only need to do lable Encoding + Missing Values)

* Disadvantage of Decision tree *

1) Computationaly expensive
2) High Variance (High chance of overfitting)

* Imbalanced Data *

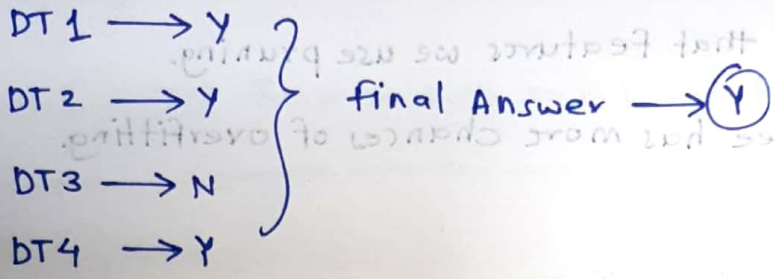
targeted Data has diff values and if these attribute data has imbalanced values, then we are using the

1) Under
2) over ⟶ random over sampling.

* Random forest *

Multiple Decision Trees are trained and after traning output of them are found

After that Max Number occurance output is taken.



Decision Tree & Random forest are not good for Multi Imbalanced Data.

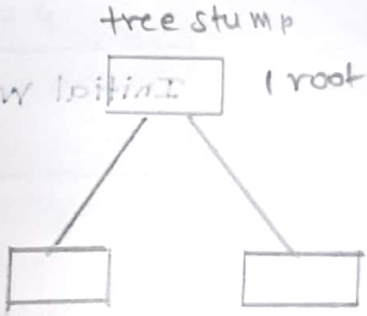
Boosting

There are Three Types in these

1) Ada Boosting
2) XG Boosting
3) Gradient Boost.

1) Ada Boosting

1) full form ⟶ Adaptive
2) we use tree stump in Ada Boosting
3) tree stump has one root node & one leaf node
4) Max depth of tree stump is 1



1) The next model try to fix the mistake done by the previous model this is called sequential traning.

2) we just increase the weight of wrong prediction of the first model.

Parallel traning ⟶ random forest
Sequential traning ⟶ Ada Boosting

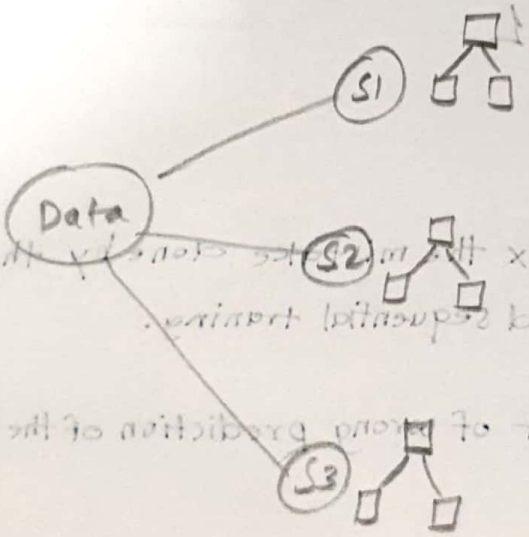
* way to increase weight *

$$\text{New Weight} = \text{old weight} \times e^{\text{stage}}$$

$$\text{Stage} = \ln\left(\frac{1 - \text{Miss classification rate}}{\text{Missclassification rate}}\right)$$

$$\text{Misclassification rak} = \frac{\text{No of wrong prediction}}{\text{Total prediction}}$$

$$\text{Initial Weight} = \frac{1}{\text{Total data point}}$$



	X_1	X_2	X_3	Y	w	pred	New w	final
1					0.2	✓	0.29	0.16
2					0.2	✓	0.29	0.16
3					0.2	✓	0.29	0.16
4					0.2	✗	0.29	0.24
5					0.2	✗	0.29	0.24
					1.0		1.81	0.96 → 4 pt missing.

Initial weight $= \frac{1}{5} = 0.2$

Initial weight $= 0.2$

Misclassification rate $= \frac{\text{wrong}}{T} = \frac{2}{5} = 0.4$

$M_{-r} = 0.4$

$$\text{Stage} = \ln\left(\frac{1 - M_r}{M_r}\right)$$

$$= \ln\left(\frac{1-0.4}{0.4}\right) = \ln\left(\frac{0.6}{0.4}\right)$$

Stage $= 0.40$

New weight = old weight $\times e^{\text{Stage}}$

$= 0.2 \times e^{0.40}$

$= 0.29$ — only place in wrong prediction

$$\text{final} = \frac{\text{New weight}}{\text{Total new weight}}$$

$$= \frac{0.29}{1.18} = 0.24$$

$0.29 \rightarrow$ final $= 0.24$

$0.2 \rightarrow$ final $= 0.16$

$$\frac{0.2}{1.18} = 0.16$$

Gradient Boosting → full tree Ada Boosting → tree stump	Gradient Boosting → Next model train on residual given by previous model. Ada Boosting → Next modle try to fix the mistake done by previous model.

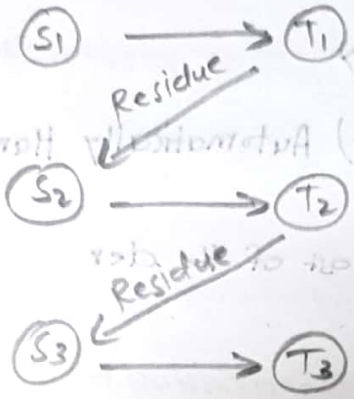
Random forest → Parallel traning Ada Boosting → Sequential traning	Random forest → tree Ada Boosting → tree stump

n_estimator = 50 → No of stump you are using in Ada Boosting.

2) Gradient Boosting

Next model train on residu of the first model.

At the end it tries to convert it in to but not direct zero



$$\hat{Y}_2 = Y + \text{learning rate} \times \text{residue 1}$$

$170 + 0.1 \times 4$

$Y - \hat{Y}_2$

Y_t	$\hat{Y}_1$	residu 1	$\hat{Y}_2$	residue 2
170	166	4	170.4	0.4
171	166	5	171.5	0.5
162	166	-4	161.6	-0.4
160	166	-6	154.4	-0.6

default tree = 100

default learning rate = 0.1

3) XG Boost

Extreme Gradient Boosting

This Boosting provides

1) Parallalization
2) it will optimize cache memory
3) Regularization (Ridge, Lasso) Automatically Handle.

Because all of these reason most of the der

SVM

1) It is a supervised learning Algorithm and it is used for classification

2) Classification of two group.

There are two types in these → 1) linear SVM

2) Non linear SVM

we found boundries to seprate two class → line → linear

→ circle (other) → Non linear

linear SVM

If data is in 2 dimension and we can seprate that Data using single straight line

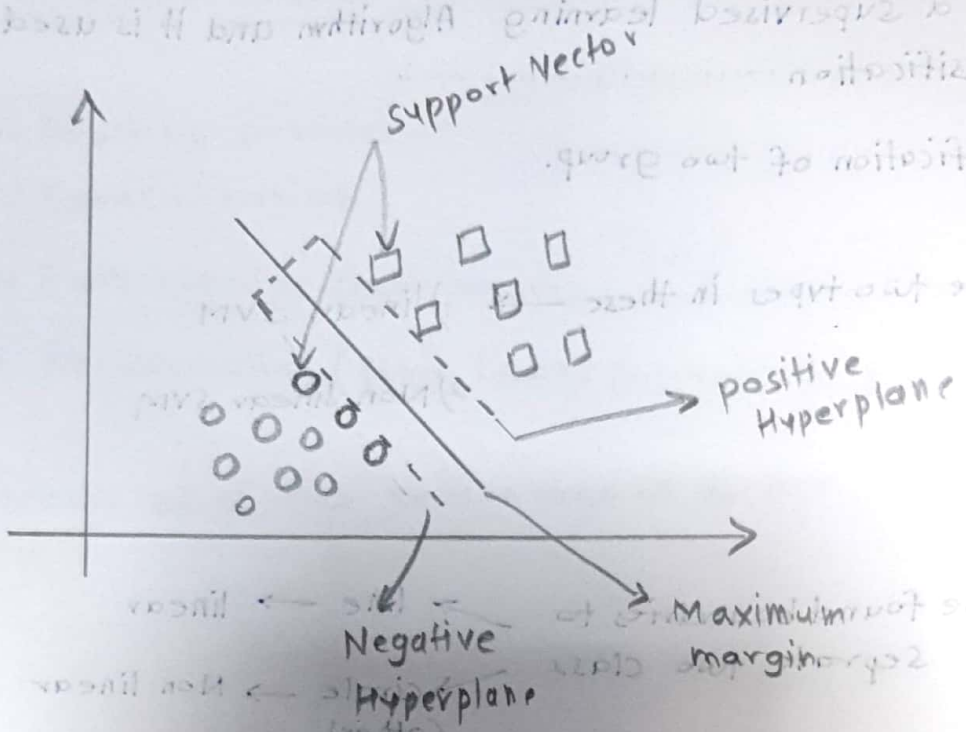
(The data points can be classified using a single straight line)

Non linear SVM

If the Data is in 2 dimension and we can not seprate that Data point with single straight line

for that we use some advance techniques like kernel tricks to classify them

circle eqn = $x^2 + y^2 = c^2$



margin → distance between Hyperplane & support Vector

Support Vector → The points that are closest to the hyperplane
There could be any number of support Vector
it depends upon the data.

If graph 2D → decision boundry is straight line

If graph has more dimension → decision boundry is Hyperplane

Best Hyperplane ⟶ Maximum distance from both the class.

eqⁿ of plane - $ax+by+cz=d$

Kernal ⟶ kernal is used for transforming higher dimen

from kernal we can transform complex data into higher dimensional simpler form

* Advantage of svm *

1) works better when data is linear
2) More effective in High dimension
3) with the help of kernal tricks we can solve any complex problem.
4) svm is not sensitive to outliers.
5) Help us with image classification.

* Disadvantage of svm *

1) chosing best kernal is not very simple/easy
2) doesn't show good result on big dataset.
3) svm hyperparameter are cost-c & gamma. It is not that easy to fine tune these hyperparameter.

KNN	from sklearn.neighbors import kNeighborsClassifier knn = kNeighborsClassifier(n_neighbors = 5) ↳default
logistic Regression	from sklearn.linear_model import LogisticRegression model = LogisticRegression()
Decision Tree	from sklearn.tree import DecisionTreeClassifier model = DecisionTreeClassifier(max_depth=5, max_leaf_nodes=5)
Random forest	from sklearn.ensemble import RandomforestClassifier RF = RandomforestClassifier(n_estimators = 100, max_depth = 4, max_leaf_node=__)
Ada Boosting	from sklearn.ensemble import AdaBoostclassifier AdaBoostclassifier(n_estimator= __ , random_state= __ , learning_rate = 1

Gradient Boosting	from sklearn.ensemble import GradientBoostingClassifier GB = GradientBoostingClassifier(n_estimator= ___ , max_depth= ___ , learning_rate = 0.1) for Regression problem ← Regressor ↓ for all the Boosting tech
XG Boost IMP Extreme gradient	pip install xgboost. from xgboost import XGBclassifier XG = XGBclassifier (randomstate=1, max_depth= _ , learning_rate= _ , n_estimator = ___)
SVM	from sklearn.svm import svc svm = svc (kernel = 'rbf', gamma = 'scale')
Random Over Sampling main focus is on Y variable always Y sampled then automatically other will be also sampled.	from imblearn.over_sampling import RandomOverSampler rs = RandomOverSampler () rs.fit_resample ( )
Random Under Sampling	from imblearn.under_sampling import RandomUnderSampler rs = RandomUnderSampler (random_state = 42)