27/12/24

ML ALGORITHEM

Day-1

* Linear Regressions :-

=> 1 Introduction to ML (AI v/s ML v/s DL v/s DS)

=> 2. Supervised ML and Unsupervised ML

=> 3. Linear Regression (Maths & Geometric Intuition)

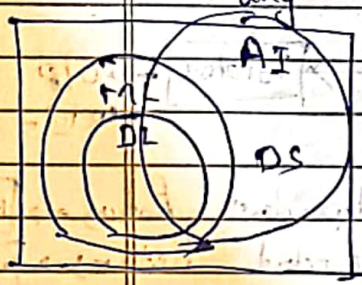
=> 4. R^2 & Adjusted R^2

=> 5. Ridge and Lasso Regression

=>

1. AI v/s ML v/s DL v/s Datascience

=> AI application is able to do it own task without any human interaction



Eg:-

Netflix → Action → Recomendation
↳ Comedy → Recomendation

Amazon → Iphone → Recomend to (the earphone)

=> ML → Subset of AI
→ Stat tools to analyze, visualize, prediction, forcasting

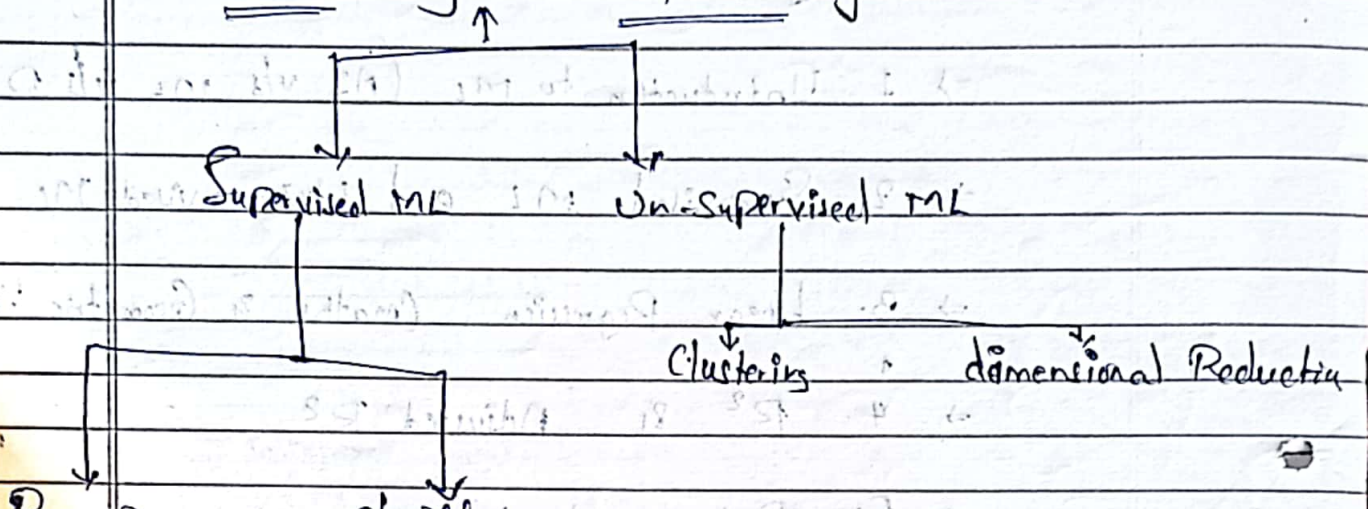
=> DL is a Subset of ML
→ mimic a human brain
→ multilayer neural network.

=> Datascience is is a Part of Everything.

vision

27/12/24

Machine learning & Deep learning :-



⇒ Supervised machine learning :-

→ Let have one dataset

Age	weight
24	62
25	63
27	72
27	62
29	73

Age → hypothesis → O/P weight

⇒ Independent feature → Age

⇒ Dependent features → weight

vision

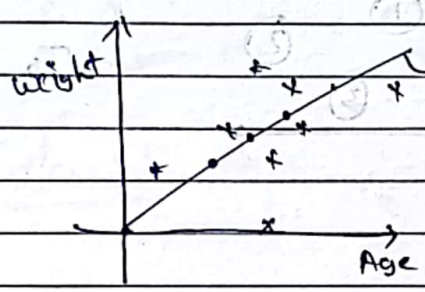
* Supervised machine learning is

1) Regression problem Statement

Age	weight	→ O/P variable
24	72	↓
23	71	Continuous Variable
25	71.5	↓
		Regression problem Statement

⇒ Whenever your output is continuous it is a Regression problem statement.

Eg:-



2) Classification problem Statement

Ex:-

No. of hours	No. of play hrs	No. of sleep hrs	O/P P/F
-	-	-	P
-	-	-	F
-	-	-	P
-	-	-	F

→ Whenever you have fix your output fix number of Categories that becomes a classification problem

Eg:- yes / No
Pass / Fail
Span / not Span

→ 2 outputs become a "Binary classification"
→ more than 2 outputs "Multiclass classification"

vision

* Un-Supervised Machine learning is

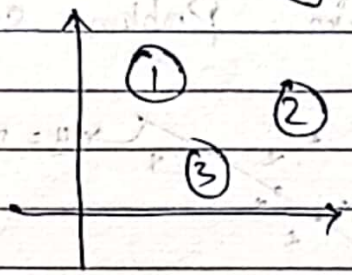
1) Clustering
2) Dimensional Reduction

i) Clustering is

Salary Age

{ No o/p variable
No dependent variable }

"clustering" → Customer Segmentation



→ To find groups/clusters

⇒ Grouping

Eg :-

Fom launching one "product"

↳ Product 1 → Rich People
Product 2 → Middle Class
Product 3 → poor people

" Add Marketting "
"Customer Segementation"

2) Dimensional Reduction :-

1000 → lower Dimension
↳ 100

↓

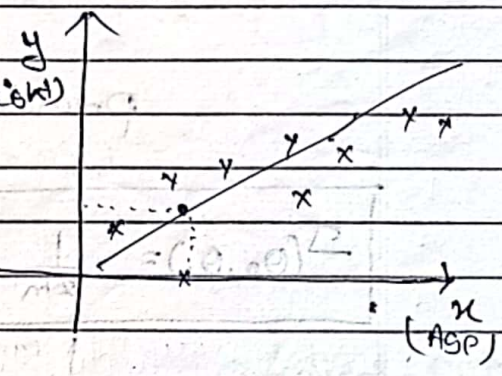
It's possible through using "PCA", "LDA", Etc.

Supervise.	Un-supervised
1) Linear Regression	1) K-means
2) Ridge & Lasso	2) DB Scan
3) Logistic Regression	3) Hierarchical clustering
4) Decision Tree	4) K-Nearest Neighbor Clustering
5) Ada Boost	5) PCA
6) Random Forest	6) LDA
7) Gradient Boosting	
8) Xg boost	
9) Naive Bayes	
10) SVM	
11) KNN	

*Explanation :-

1) Linear Regression :-

→ Training Dataset
↓
Model
↓
New Age → Hypothesis → O/P weight



→ y is a linear function of x.

Formulas :
- $y = mx + c$
- $y = \beta_0 + \beta_1 x$
- $h_\theta(x) = \theta_0 + \theta_1 x$

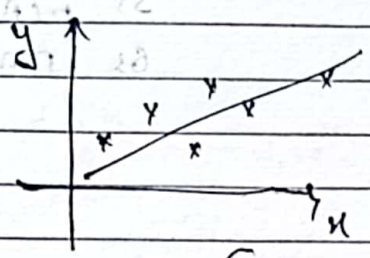
Equation of Straight line :-

$y = mx + c$

$y = \beta_0 + \beta_1 x$

$h_\theta(x) = \theta_0 + \theta_1 x$

$h_\theta(x) = \theta_0 + \theta_1 x$



θ_0 = Intercept (when $x = 0$)

θ_1 = slope or coefficient

x_i = data points

→ Cost function

⇒ distance formula ⇒ b/w predicted & Real time.

$(h_\theta(x) - y)$

predicted point is $h_\theta(x)$.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$ → [Derivation]

↓

Squared Error Function

⇒ What we need to solve

Squared Error Function:

minimize $\frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

↓

minimize $J(\theta_0, \theta_1)$

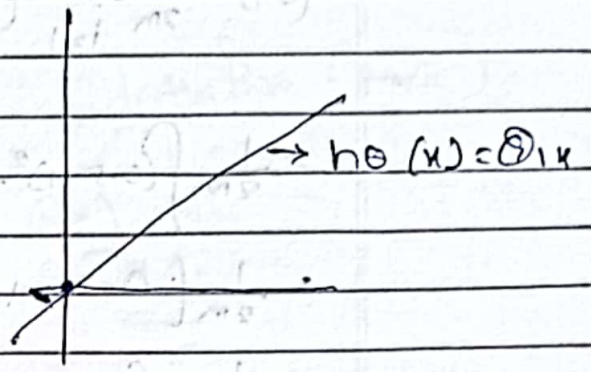
θ_0, θ_1

vision

*hypothesis

$h_\theta(x) = \theta_0 + \theta_1 x$

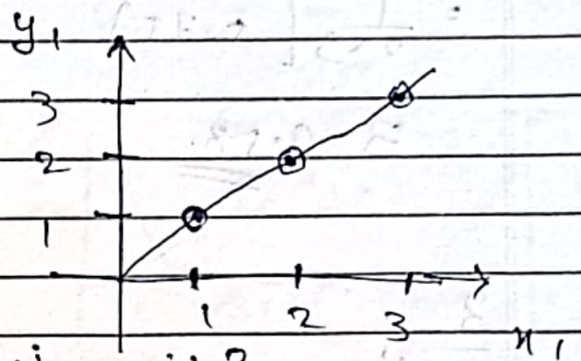
If $\theta_0 = 0$



*Example :-

$h_\theta(x) = \theta_1 x$

$\theta_1 = 1$ (1,1) (2,2) (3,3)

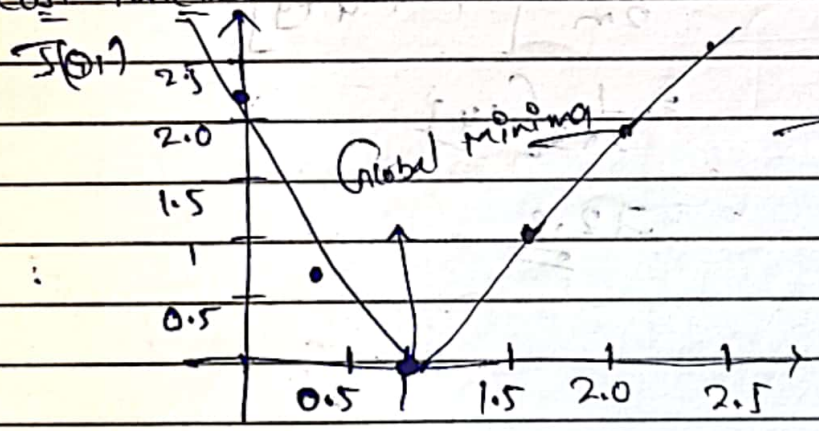


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{3} (h_\theta(x^i) - y^i)^2$$

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

$J_{\theta_1} = 0$ / $\theta_1 = 1$, $J_{\theta_1} = 0$

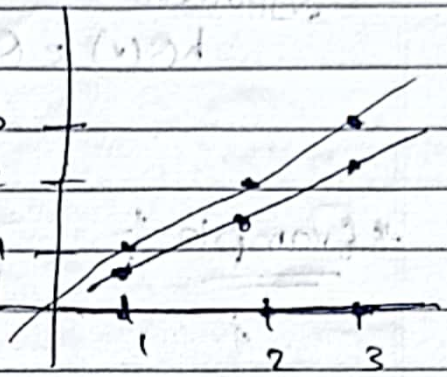
* Cost function



→ Gradient decent

Now, $\theta_1 = 0.5$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{3} (h_\theta(x^{(i)}) - y^{(i)})^2$$



$$= \frac{1}{2m} \left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2 \right]$$

$$= \frac{1}{2m} \left[0.25 + 1 + 2.25 \right]$$

$$= \frac{1}{2\times 3} \left[3.5 \right]$$

$$\approx 0.58$$

3

Now $\theta_1 = 0$

when $\theta_1 = 0$ it will be come in x axis

4

Now $\theta_1 = 0$

$$J(\theta_1) = \frac{1}{2m} \left[(0-1)^2 + (0-2)^2 + (0-3)^2 \right]$$

$$= \frac{1}{2m} \left[1 + 4 + 9 \right]$$

$$= \frac{1}{2\times 3} \left[14 \right]$$

$$\approx 2.3$$

Conversions Algorithms

Repeat until conversion.

{

$$\theta_j = \theta_j - \alpha \left[\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \right]$$

derivative (slope)

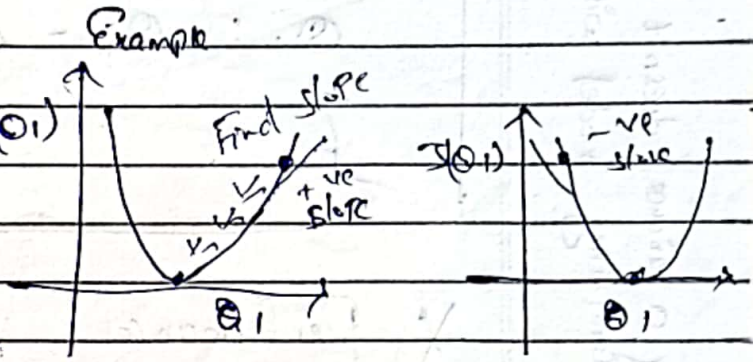
}

α → learning rate

always take α value less value only

very very good decision to take small value

Example



⇒ Gradient Descent Algorithm

repeat until Convergin

{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}

J is always 0 and 1

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

⇒

$$j = 0 \Rightarrow \frac{\partial}{\partial \theta_0} j(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

Conversion Algorithm

$$h_\theta(x) = \theta_0 + \theta_1(x)$$

Conversion Algorithem

$$j=1 \Rightarrow \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

Repeat until Convergen

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

Performance metrics

R^2 and Adjusted R^2

$h_\theta(x)$

$$R^2 = 1 - \frac{SS_{Residual}}{SS_{total}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

residual :- Sum of Squared residual

total :- Total sum of Squares

R^2 ranges 0 and 1

$= 1 -$ Low → Small numb

high → big numb

⇒ Adjusted R^2 formula

$$1 - \frac{(1 - R^2)(N-1)}{N-P-1}$$

N = No of data Points

P = No of Predictors

⇒ If P is increasing $(N-P-1)$ value is decreasing

big No / Small No → Sum value to get

When to use

→ R^2 is useful for understanding how much variance is explained by your model

→ Adjusted R^2 is better for model comparison and ensuring you don't overfit by adding irrelevant predictors

Day -2

* Agenda

1. Ridge & Lasso Regression
2. Assumption of linear Regression
3. Logistic Regression
4. Confusion metrix
5. practicals for linear, Ridge, Lasso, & logistic

Explanation :-

Overfitting (Low Bias)

model Performs well → Training data

fails to perform well → Test Data (High Variance)

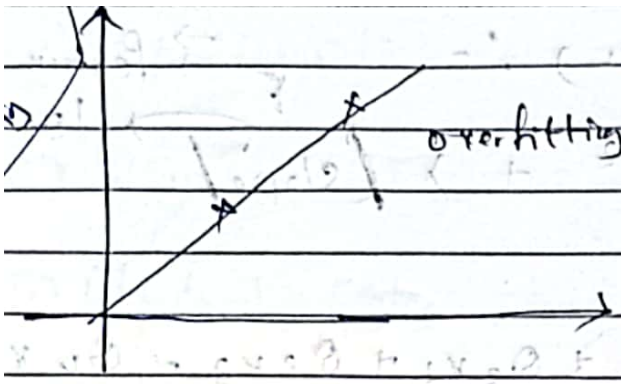
Underfitting (High Bias, High variance)

model Accuracy is bad with training data

model Accuracy is also bad with test data.

e.g. :-

Model - 1	Model - 2	Model - 3
Training Accuracy = 90%.	Training Acc = 92%.	Training Acc = 70%.
Test Acc = 80%.	Test Acc = 91%.	Test Acc = 65%.
↓↓	↓↓	↓↓
Overfitting	Generalized model	Underfitting
Low Bias & high variance	Low Bias & high variance	High bias & high variance



$h\theta(x) = \hat{y}$

$J(\theta_1) = 0$

$= \frac{1}{2m} \sum_{i=1}^{m} \left(h\theta(x^{(i)}) - y^{(i)}\right)^2$

here, $= (\hat{y}^{(i)} - y^{(i)})^2 + \lambda(\text{slope})^2$

$= 0$

Ridge (L_2 Regularization)

$h\theta(x) = \theta_1 x$

↳ Slope

when λ value is 1 ∴ slope is 1

$\theta + 1(2)^2 \equiv$ minimum

'λ' is hyper parameter

Now, when $\theta_1 = 0$,

$\theta + 1(2)^2$

(Prevent overfitting)

$= (\hat{y}^{(i)} - y^{(i)})^2 + \lambda(\text{slope})^2$

$= (\text{small value}) + 1(1.5)^2$

$= (\text{small value}) + 2.25$

$= 1 \simeq 3$

⇒ iteration {Hyper parameter}

R^2, Adjusted R^2

⇒ What is lambda (λ)

⇒ Hyper parameter

→ We get cross validation

Lasso (L1 Regularization) :- It will ux → Feature selection

$= (\hat{y} - y)^2 + \lambda |slope|$ → $|\theta_1 + \theta_2 + \theta_3 \cdots \theta_n|$

⇒

Example

$h\theta(x) = \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \cdots \theta_n x_n$

Imp
① preventing overfitting } L1 Regularization
② Feature selection } L2 Regularization

→ Cross validation

⇒

* Ridge Regression (L2 Normilization)

Cost Function = $(h\theta(x^{(i)}) - y^{(i)})^2 + \lambda (slope)^2$

* Purpose : preventing = overfitting

Loss Regression (L1 Reg)

Cost function = $(h\theta(x^{(i)}) - y^{(i)})^2 + \lambda |slope|$

* purpose : 1) Prevent overfitting
2) Feature Selection → $|\theta_1 + \theta_2 + \theta_3 \cdots \theta_n|$

* Assumption of linear Regression :-

① R in Normal / Gaussian Distribution → model will get trained well

② Standard Scaler / standarization (Scaling data) → z-score
mean = 0, Stand deviation (σ) = 1

③ Linearity

④ Multi-collinarity

Q5 :-

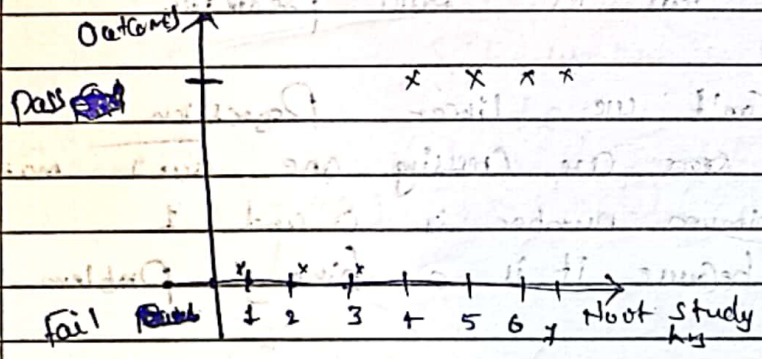
⇒ x3 [x1 x2] [y]

Variation Inflation factor

* Logistic Regression (Classification) :-

⇒ Binary Classification

No. of hrs Study	No of hrs Play	Pass / Fail
–	–	P
–	–	F
–	–	P
–	–	F



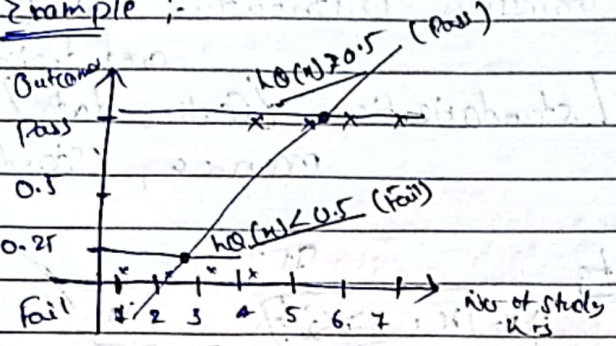
∴ If study less than 3 hrs "Fail"

∴ If Study more than 3 hrs "Pass"

⇒ Can we solve "linear Regression" ?

: yes we can solve ...

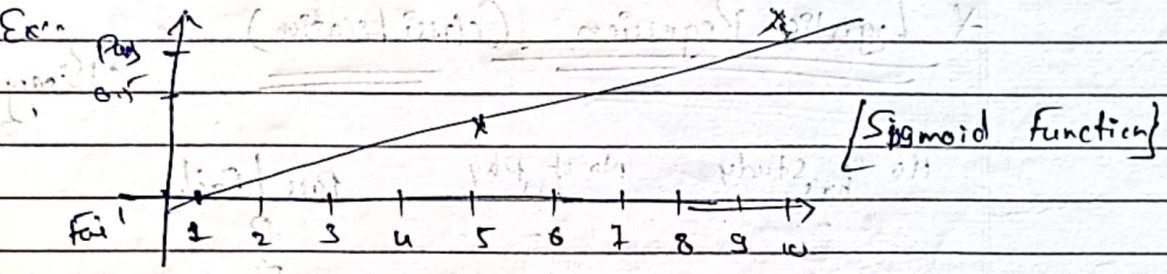
Example :-



$h_\theta(x) < 0.5 \Rightarrow 0 \rightarrow$ Fail

$h_\theta(x) \geq 0.5 \Rightarrow 1 \rightarrow$ Pass

* but we don't use because some confusion let's check



$h_\theta(x) = 10$

⇒ here 5 is pass but it's look like a fail its fault prediction

⇒ Can't use linear Regression we are crossing one one's maxima & minimum number is 0 and 1 because it is a binary problem

* Decision Boundary logistic Regression :-

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

$h_\theta(x) = \theta^{T} x$ (Transforms)

$h_\theta(x) = \theta^T x$

(Squash max Straight line)

hypthesis

$h_\theta(x) = \theta_0 + \theta_1 x_1$

$h_\theta(x) = g(\theta_0 + \theta_1 x_1)$

Let $z = \theta_0 + \theta_1 x$

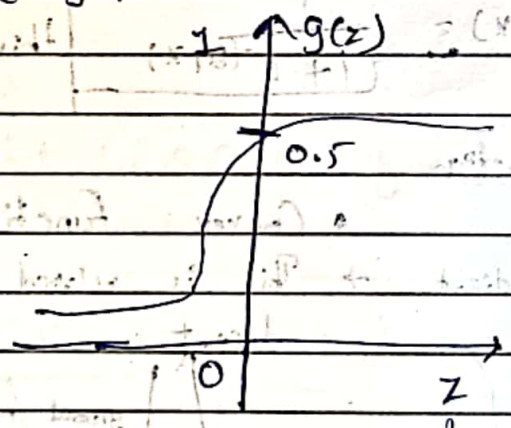
$h_\theta(x) = g(z)$

$h_\theta(x) = \frac{1}{1+e^{-z}}$ → This function is called Sigmoid (or) Logistic function

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0 + \theta_1 x)}}$$

↑ It works well because It is being to squash the function

Ex: graph:-



When $g(z) \geq 0.5$

$z \geq 0$

It will not confusion not get false value

⇒ Training set :-

$\{(x^1, y^1), (x^2, y^2), (x^3, y^3) \ldots\ldots (x^n, y^n)\}$

$y \rightarrow 0$ (or) $1 \longrightarrow$ 2 o/p

Sigmoid

$$h_\theta(z) = \frac{1}{1+e^{-z}}$$

$$z = \theta_0 + \theta_1 x_1$$

$$\downarrow$$

$$z = \theta_1 x$$

Change parameter θ_1

⇒ Cost function

Linear Regression $J(\theta_1) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} (h_\theta(x^i) - y^{(i)})^2$

Logistic Regression $h_\theta(x) = \frac{1}{1+e^{-(\theta_1 x)}}$

here,

Logistic Regression Cost Function :-

$$\frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

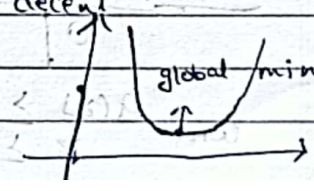
here, $h_\theta(x) = \frac{1}{1+e^{-(\theta_1 x)}}$

⇒ We can't use this cost function because $\frac{1}{1+e^{-(\theta_1 x)}}$ this non-convex function

⇒ Latest is non-convex function

* Non-Convex Function
  → This is related to Gradient decent
  → here, so many local minima
  So we can't reach global minima
  → Vision that is the reason we can't use Cost Fn

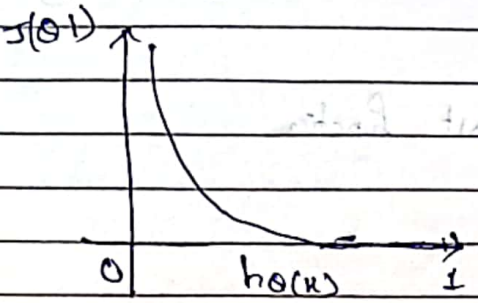
* Convex Function
  → This is related to Gradient decent



Logistic Regression Cost function

$$J(\theta) = \begin{cases} -\log(h_\theta(x^{(i)})) & y = 1 \\ -\log(1 - h_\theta(x^{(i)})) & y = 0 \end{cases}$$

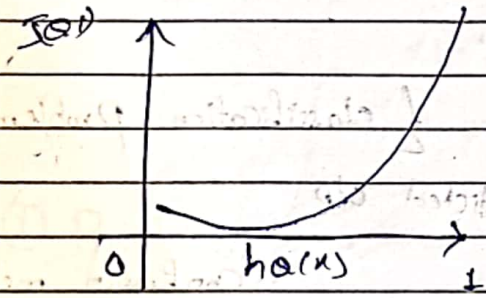
→ if $y = 1$



Cost = 0 if $[y = 1, h_\theta(x) = 1]$

Classification problems are abt

→ if $y = 0$



→

$$\text{Cost}(h_\theta(x^{(i)}), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Final cost Function ⇐

$$\text{Cost}(h_\theta(x^{(i)}), y) = -y \log(h_\theta(x^{(i)})) - (1-y) \log(1 - h_\theta(x^{(i)}))$$

vision

if $y = 1$

$\text{Cost}(h_\theta(x^i), y) = -\log(h_\theta(x^i))$

if $y = 0$

$\text{Cost}(h_\theta(x^i), y) = -\log(1 - h_\theta(x^i))$

} this both function are proved by final Cost function

$\Rightarrow$

$$J(\theta_1) = \frac{-1}{2m} \sum_{i=1}^{m} \left[(y^i \log(h_\theta(x)^i) + (1 - y^i) \log(1 - h_\theta(x)^i) \right]$$

$\downarrow$

→ This becomes Entire cost function

Repeat until Convergen . . . .

{

$$\theta_j := \theta_j = \alpha \frac{\partial}{\partial \theta_j} (J(\theta_1))$$

}

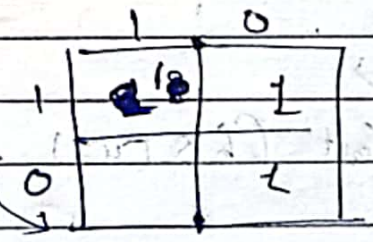
$\Rightarrow$ Performance metrics { classification problem }

y → actual o/p

$\hat{y}$ → predicted o/p

x_1	x_2	y	$\hat{y}$
–	–	0	1
–	–	1	1
–	–	0	0
–	–	1	1
–	–	1	1
–	–	0	1
–	–	1	0

Confussion metrics look like this



Predicted \ Actual	1	0
1	TP	FP
0	FN	TN

→ Confusion matrix

→ This always gives right o/p

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

$$= \frac{3+1}{3+2+1+1} = \frac{4}{7}$$

$$= 0.57 \Rightarrow 57\%$$

→ when we have 0's are 900, 1's are 100 } It is imbalance data

when,

0's → 600, 1's → 400 } Balanced data

⇒

0 : 900, 1 : 100 } bad It is a Bias data

{model → 0 = 900/1000 = 90%}

Formula's

① Precision: $\frac{TP}{TP+FP}$

② Recall: $\frac{TP}{TP+FN}$

③ F-Score

$$F\text{-}Beta = (1+\beta^2)\frac{Precision \times Recall}{\beta^2 \times Precision + Recall}$$

$$\beta=1 = (1+1)\frac{Precision \times Recall}{Precision + Recall}$$

$$= 2\frac{(Precision \times Recall)}{Precision + Recall}$$

Harmonic mean

$$\frac{2xy}{x+y}$$

This is called a "Harmonic mean" this focuses both False negative & False positive

$\beta = 0.5$ $\left(1+(0.5)^2\right)\frac{P \times R}{(0.25)P+R}$

F 0.5 score

$\beta = 2$ FN >> FP.

F. 2 score

06/01/25

Day-3

* Today Agenda :-
1) Practical (Simple examples) → 1. Linear Regression 2. Ridge & lasso 3. Logistic Regression
2) Naive Baye's Intuition
3) KNN algorithems

⇒ Linear Regression, Ridge and Lasso :-

Here Code :-

⇒ importing all libraries
```
from sklearn.datasets import fetch_california_housing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

⇒ # load the data set
```
df = fetch_california_housing()
```
⇒ We check type here of dataset
```
type(df)
```
⇒ we print dataset
```
df          # Now we shown some data & target data
```

```
# we Convert dataset to DataFrame
dataset = pd.DataFrame(df.data)        → In dataset one column name
dataset.Columns = df.feature_names
dataset.head()  # print a ~~not~~ entire data set in DataFrame
                ↳ This data is independent feature.
print(dataset.head())  # its print full
```

vision

Dividing the dataset into independent and dependent features

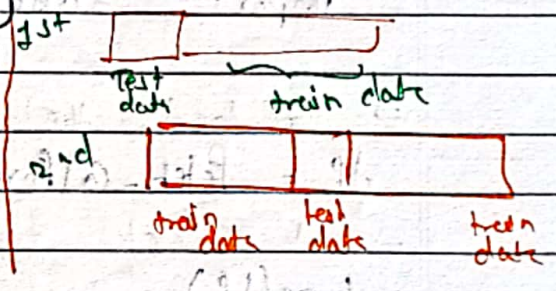
```
X = dataset.iloc[:, :-1]      # skipping the last feature
                              # Independent feature
y = dataset.iloc[:, -1]       # dependent feature

X.head()
```

Now we use algorithms

Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
lin_reg = LinearRegression()
mse = cross_val_score(lin_reg, X, y
      , scoring = 'neg_mean_squared
      Error', cv=5)
```

cross_val_score




```
mean_mse = np.mean(mse)
print(mean_mse)          # output :- -0.5582901717186536
```

* Ridge algorithms

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
ridge = Ridge()
params = {'alpha' : [1e-15, 1e-10, 1e-8, 1e-3,
                     1e-2, 1, 5, 10, 20]}
ridge_regressor = GridSearchCV(ridge, params, scoring
                  = 'neg_mean_squared_error', cv=5)
ridge_regressor.fit(X, y)

print(ridge_regressor.best_params_)
print(ridge_regressor.best_score_)
```

#output :-

```
{'alpha' : 20}
-0.5581020035825638
```

Ridge tries to reduce the overfitting

* Lasso Regression :-

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
lasso = Lasso()
params = {'alpha': [1e-15, 1e-10, 1e-8, 1e-3,
                    1e-2, 1, 5, 10, 20]}
lasso_regressor = GridSearchCV(lasso, params, scoring
                  = 'neg_mean_squared_error', cv=5)
lasso_regressor.fit(X, y)

print(lasso_regressor.best_params_)
print(lasso_regressor.best_score_)
```

* Using train test split

```
X = dataset.iloc[:, :-1] # Independent feature
y = dataset.iloc[:, -1] # dependent feature

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
mse = cross_val_score(lin_reg, X_train, y_train, scoring
  = 'neg_mean_squared_error', cv=5)
print mean_mse = np.mean(mse)
print(mean_mse)
```

⇒ Ridge Regression Algorithms :-

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
ridge = Ridge()
params = {'alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2, 2,
                    1, 10, 15, 20]}
ridge_regressor = GridSearchCV(ridge, params, scoring='neg_mean_
    Squared_Error', cv=5)
ridge_regressor.fit(X_train, y_train)

print(ridge_regressor.best_params_)
print(ridge_regressor.best_score_)
```

* Lasso Regression

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
lasso = Lasso()
params = {'alpha': [1e-15, 1e-10, 1e-8, 1e-3, 1e-2,
                    1, 2, 10, 20]}
lasso_regressor = GridSearchCV(lasso, params, scoring =
                  'neg_mean_squared_error', cv=5)
lasso_regressor.fit(X_train, y_train)

print(lasso_regressor.best_params_)
print(lasso_regressor.best_score_)
```

→

```
# using r2 score (r square)
y_pred = lasso_regressor.predict(X_test)
from sklearn.metrics import r2_score
r2_score1 = r2_score(y_pred, y_test)
print(r2_score1)
```

* Logistic Regression :-

```
from sklearn.linear_model import LogisticRegression
import pandas as pd
from sklearn.datasets import load_breast_cancer
df = load_breast_cancer()
df
X = pd.DataFrame(df['data'], columns = df['feature_names'])   # independent Variable
X.head()
y = pd.DataFrame(df['target'], columns = ['Target'])
y.
y['Target'].value_counts()
        # using train_test_split :-
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
                X, y, test_size = 0.33, random_state = 42)
params = {'C': [1, 5, 10]}, {'max_iter': [100, 150]}
model1 = LogisticRegression(C=100, max_iter=100)
GridSearchCV(estimator = model1, param_grid = params,
                        scoring = 'f1', cv=5)
model1.fit(X_train, y_train)
model1.best_params_
model1.best_score_
y_pred = model.predict(X_test)
y_pred

from sklearn.metrics import Confusion_matrix, classification_report

Confusion_matrix(y_test, y_pred)
accuracy_score(y_test, y_pred)
Print(Classification_report(y_test, y_pred))
```

* Naive Baye's Intuition :-

{Specifically use for classification}

↓

{Base Theorem}

Rolling a Dice

{1, 2, 3, 4, 5, 6}

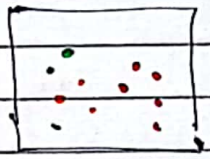
$P(1) = \frac{1}{6}$ $P(2) = \frac{1}{6}$

$P(2) = \frac{1}{6}$

{Independent Events}

Dependent Event :-

I have a Bag of marble



$P(R) = \frac{7}{11}$ ← First Element removed , $P(G) = \frac{3}{10}$ ← Second element removed

$P(R \text{ and } G) = P(R) * P(G/R)$

$P(A \text{ and } B) = P(A) * P(B/A)$

$P(A \text{ and } B) = P(B \text{ and } A)$ {we can write like this "Yes"}

Formula :-

$P(A) * P(B/A) = P(B) * P(A/B)$

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

→ Baye's Theorem

Extra Naive Baye's

→ I have Same Features like

x_1 x_2 x_3 x_4 ..... x_n y

Independent Feature — dependent Feature

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

$$= \frac{P(y) * P(x_1/y) * P(x_2/y) * \dots * P(x_n/y)}{P(x_1) * P(x_2) * P(x_3) * \dots * P(x_n)}$$

⇒ Dataset :-

x_1	x_2	x_3	x_4	y
				yes
				No

Remember

Formula

$$P(y=yes/x_i) = \frac{P(yes) * P(x_1/yes) * P(x_2/yes) * P(x_3/yes) * P(x_4/yes)}{P(x_1) * P(x_2) * P(x_3) * P(x_4)}$$

remember

Constant (Fixed) →

$$P(y=No/x_i) = \frac{P(No) * P(x_1/No) * P(x_2/No) * P(x_3/No) * P(x_4/No)}{P(x_1) * P(x_2) * P(x_3) * P(x_4)}$$

Constant (Fixed) →

x_i → yes / No

$P(yes/x_i) = 0.13$ $\quad P(No/x_i) = 0.05$

$\geq 0.5 \Rightarrow 1$

$< 0.5 \Rightarrow 0$

$P(Yes/Xi) = \frac{0.13}{0.13+0.05} = 0.72 = 72\%$ Final Answer

$P(No/Xi) = 1 - 0.72 = 0.28 = 28\%$.

* Dayset :- Independent Future | Binary Output

Day	Outlook	Temperature	Humidity	Wind	Play Tenn
D1	Sunny	Hot	High	Weak	N
D2	Sunny	Hot	High	Strong	N
D3	Overcast	Hot	High	Weak	Y
D4	Rain	Mild	High	W	Y
D5	Rain	Cool	Normal	W	Y
D6	Rain	Cool	Normal	S	N
D7	Overcast	Cool	Normal	S	Y
D8	Sunny	Mild	High	W	N
D9	Sunny	Cool	Normal	W	Y
D10	Rain	Mild	Normal	W	Y
D11	Sunny	Mild	Normal	S	Y
D12	Overcast	Mild	High Normal	S	Y
D13	Overcast	Hot	Normal High	W	Y
D14	Rain	Mild	High Normal	S	N

=> Outlook :-

	Yes	No	Probability(yes) P(Y)	Probability(no) P(N)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
Total	9	5		

* Temparature :-

	Yes	No	P(Y)	P(N)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cold	3	1	3/9	1/5
Total	9	5		

* Play Tennis :-

		P(yes)	P(No)
yes	9		
No	5		
Total	14		

* Play Tennis :-

		P(yes)	P(No)
yes	9	9/14	
No	5		5/14
Total	14		

Now,

→ Test (Sunny, Hot) → o/p ?

→ $P(yes, (Sunny, Hot)) = P(yes) * P(Sunny/yes) * P(Hot/yes)$

$P(Sunny) * P(Hot)$

$= \frac{2}{14} \times \frac{2}{9} \times \frac{2}{9}$

$= \frac{2 \cdot 7}{7 \times 9} = \frac{2}{63} = 0.03$

$P(No) \times P(Sunny|No) \times P(Hot|No)$

$P(Sunny) \times P(Hot)$ → Constant

$P(No \mid Sunny, Hot) = 5/14 \times 4/5 \times 3/5$

$= \frac{3}{35} = 0.085$

$\Rightarrow P(Yes \mid Sunny, Hot) = 0.031 = 1 - 0.73 = 0.27 = 27\%$

$\Rightarrow P(No \mid Sunny, Hot) = 0.085 = \frac{0.085}{0.031 + 0.085}$ } Normalized factor

$= 0.73 = 73\%$

When → (Sunny, hot) → Yes or No

Answer → No

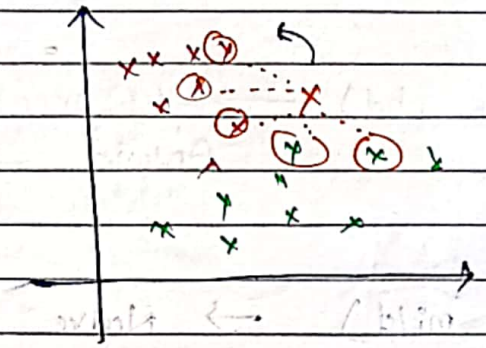
Assignment Now, new sum:-

(Overcast, mild) → Naive Bayes ?

2) KNN Algorithm :- (K-Nearest Neighbour)

Classification | Regression

i) Classification :- ⇒ K-Nearest Neighbour

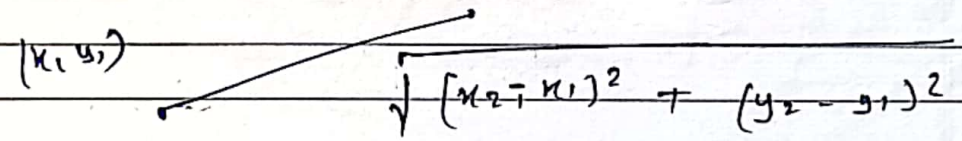


Note: K=5

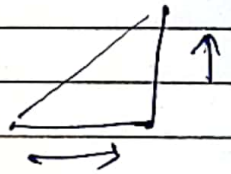
Calculate distance Red - 3
Green - 2

→ We use two distance
i) Euclidean Distance
ii) Manhattan Distance

⇒ i) Euclidean Distance



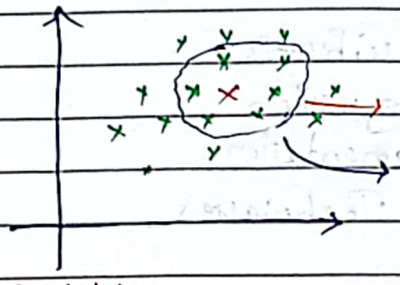
ii) Manhattan Distance



K=1 to 50

ii) Regression :-

K=5 → HyperParameter

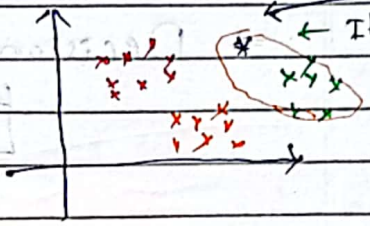


Avg all point we get o/p

Independent Future

* Imp Note

* KNN works very bad
* Outliers
* Imbalanced



Outlier

← It's calculate one group

vision

Day - 4

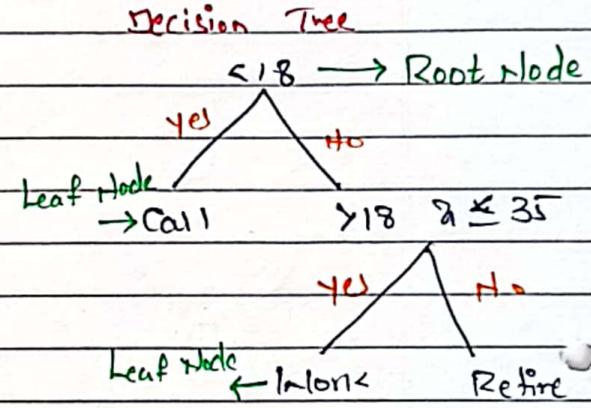
* Agenda

1. Decision Tree Classification
2. Decision Tree Regression
3. Practical Implementation
4. Ensemble Techniques.

⇒ Decision Tree { Solving many usecases }
- → Regression
- → Classification }

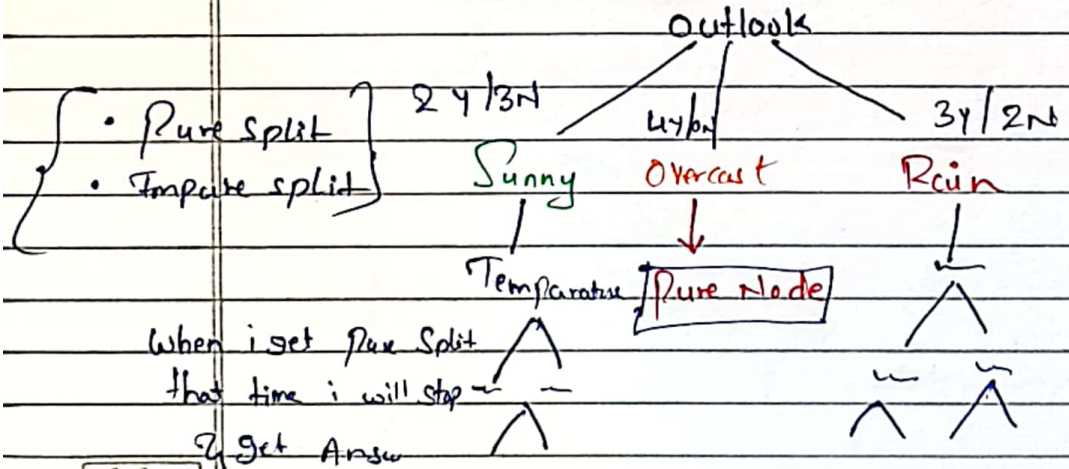
⇒

```
if age ≤ 18:
    print ("Go to college")
Elif age >18 and age ≤ 35:
    print ("work")
Else:
    print ("Retire")
```



* Dataset (page No - 31) → Classification

- Pure split
- Impure split



When i get Pure Split that time i will stop
& Get Answ

vision

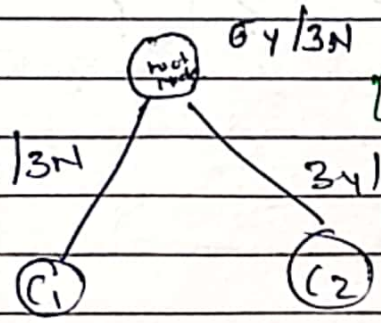
1. Purity → Pure split ???
   - ↳ Entropy
   - ↳ Gini coefficient / Impurity

2. How the features are selected
   - ↳ Information gain ??

⇒ ⇒

1 Entropy

$H(S) = -P_+ \log_2 P_+ - P_- \log_2 P_-$



P_+ = Probability of Yes

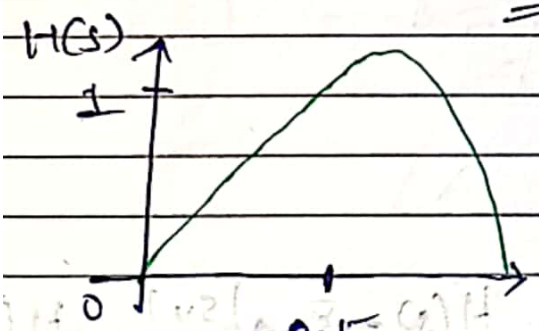
2. Gini Impurity

$G_I = 1 - \sum_{i=1}^{n} (P_i)^2$

Entropy : $H(S) = -\frac{3}{3} \log_2 \frac{3}{3} - \frac{0}{3} \log_2 \frac{0}{3}$

$= -1 \log_2 1$

$= 0$ → Pure split 100 %.



Pure split 100%.

$P^+ = 0.5$
$P^- = 0.5$
$P = 1 - Q$
$Q = 1 - P$

$\Rightarrow$

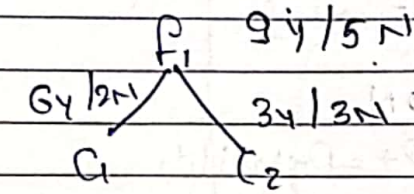
$$H(S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$$

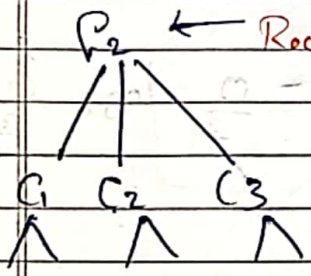
$$= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}$$

$$= 1 \longrightarrow \text{Impure split} \quad 50\% - 50\%$$

$\Rightarrow$ Your always Etropy between 0 to 1

$\rightarrow$ (2) Which feature to split ??? In my data like F_1 F_2 F_3



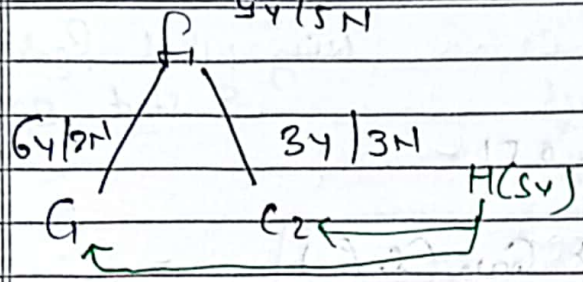


Like basically we "Information Gain"

Formula :-

~~Gain (S~~

$$\text{Gain}(S, F_1) = H(S) - \sum_{v \in \text{values}} \frac{|S_v|}{|S|} H(S_v)$$



H(S) = Root Node

$$H(S) = -P_+ \log_2 P_+ - P_- \log_2 (P_-)$$

$$= -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \left(\frac{5}{14}\right)$$

$$\simeq 0.94$$

For F_1

$$H(S_v) = -\frac{6}{8} \log_2 \frac{6}{8} \log_2 - \frac{2}{8} \log_2 \frac{2}{8}$$

$$\boxed{H(F_1) = 0.81}$$

For F_2

$$H(S_v) = -\frac{3}{6} \log_2 \frac{3}{6} - \log_2 - \frac{3}{6} \log_2 \frac{3}{6}$$

$$\boxed{H(F_2) = 1}$$

Finally,

$$\text{Gain}(S, F_1) = 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$$

$$\text{Gain}(S, F_2) \simeq 0.048$$

For Future 2 :-

Using which feature should I start splitting first

If i get

Gain $(S, f_2) = 0.051$

$$\text{Gain}(S, f_2) \gg \text{Gain}(S, f_1)$$

$\rightarrow f_2$ - to start the split

* Gini Impurity :-

Formula :-

$$G.I = 1 - \sum_{i=1}^{n} (p)^2$$

n = no of outputs

We have Node ◯ 24/27

When Etrophy = 1

When Gini = 0.5

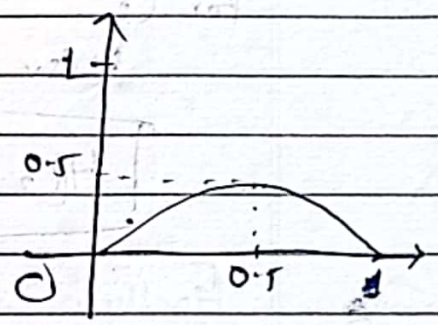
$$= 1 - [(p_+)^2 + (p_-)^2]$$

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

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

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

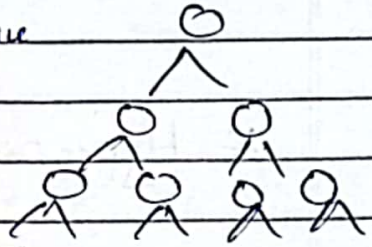
$$= 0.5$$



When we use Gini impurity (or) Entropy

→ More time where it will probably take for Execution Purpose

→ Decision tree having a worst time complexity like 100 feature



Entropy → log function

Gini impurity → Simple Maths

⇒ More taken time "Entropy"

When small no of feature then it will use

Outerwise

Gini impurity is best

Fast

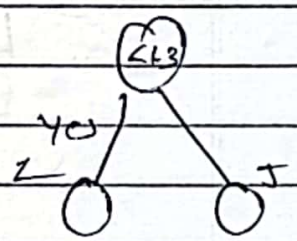
Gini >> Entropy

~~Let I Example~~ Example

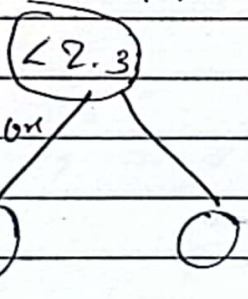
→ continuous

f_1	O/P	⇒	f_1	O/P
2.3			1.3	
1.3			2.3	
4			3	
5			4	
3			5	
~~7~~			7	
3				

Information gain



Leaf node

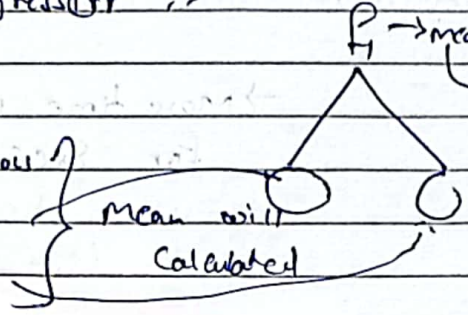


Best Information gain

Decision Tree regressor :-

$f_1 : f_2$ O/P

Continuous



We use MSE(OR) MAE

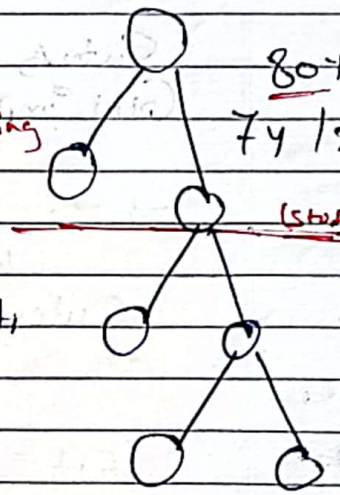
$\frac{1}{2m}\sum_{i=1}^{n}(y_i - y_i)^2$

* Hyperparameters :-

Decision →

Post pruning

pre-pruning



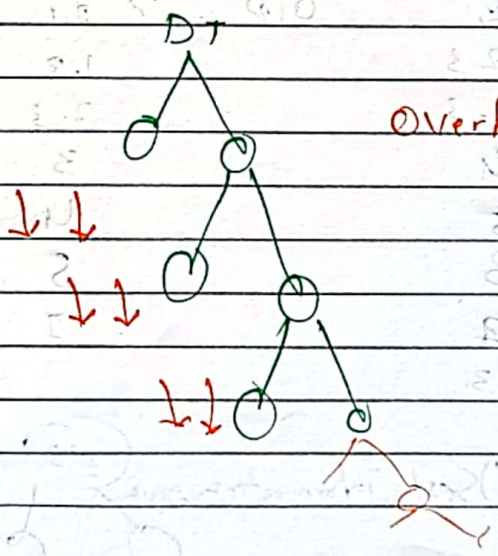
Prepruning decided by Hyperparameters

max, depth, max. leaf.

→ GridsearchCV

What is the main Problem of Decision Tree

Low Bias
High variance



Overfitting ??

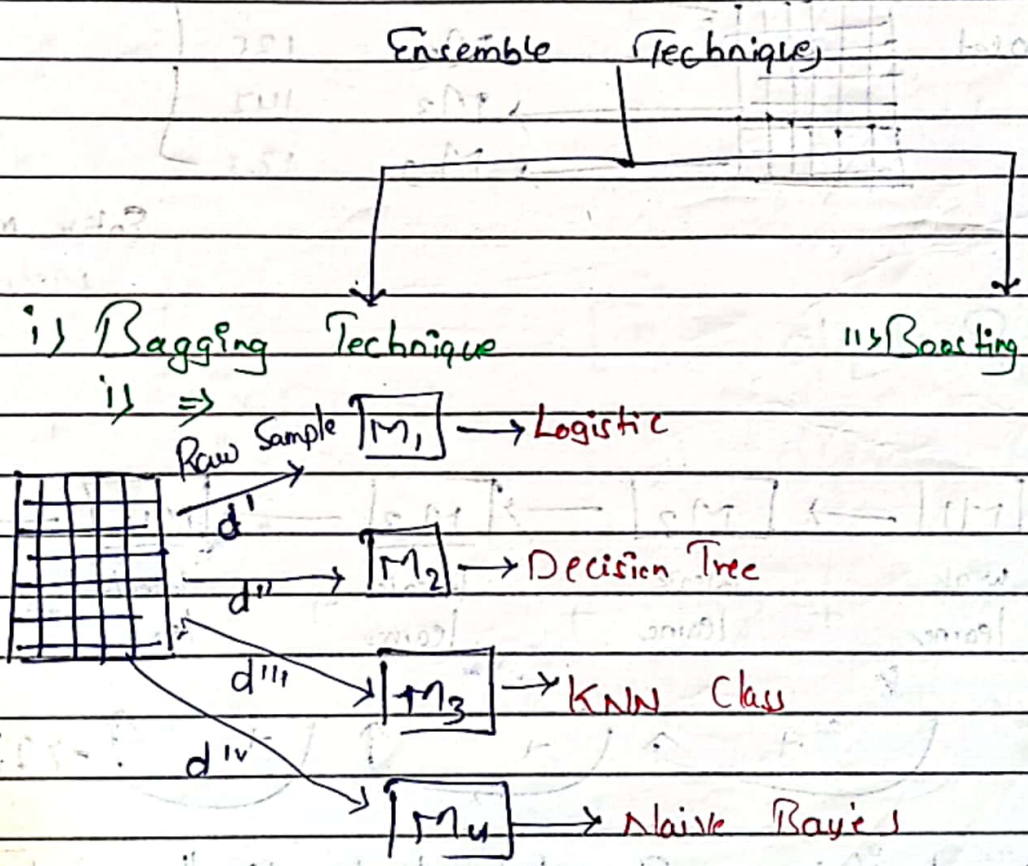
Day-5

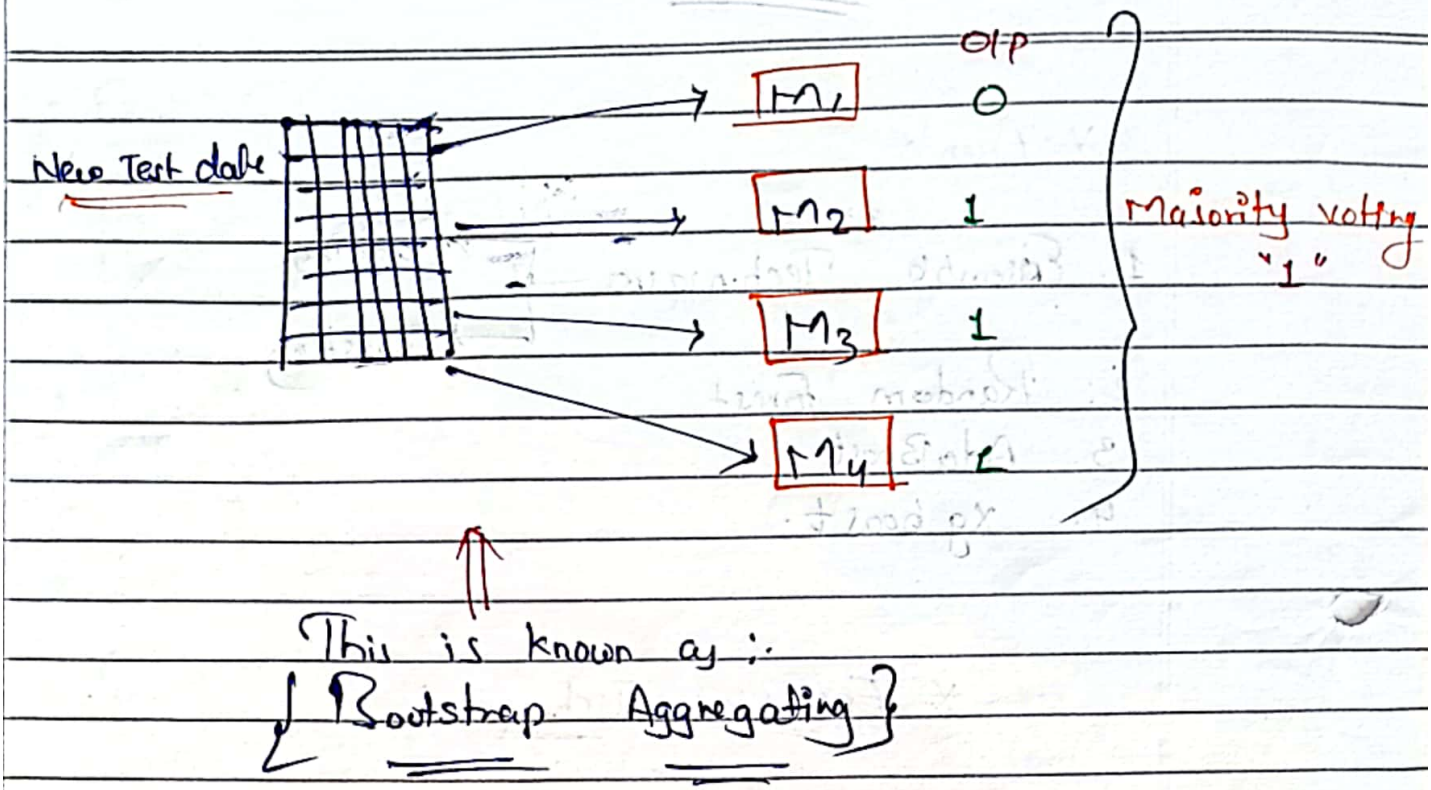
* Agenda

1. Ensemble Techniques → Bagging, Boosting
2. Random Forest
3. AdaBoost
4. xg boost.

* Ensemble Techniques :-

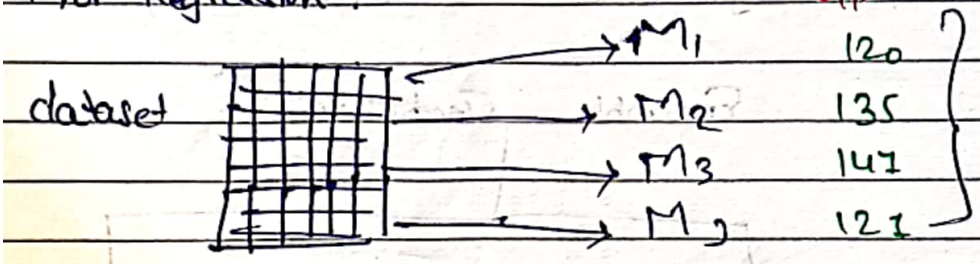
Multiple Algorithms to Solve a problem ???





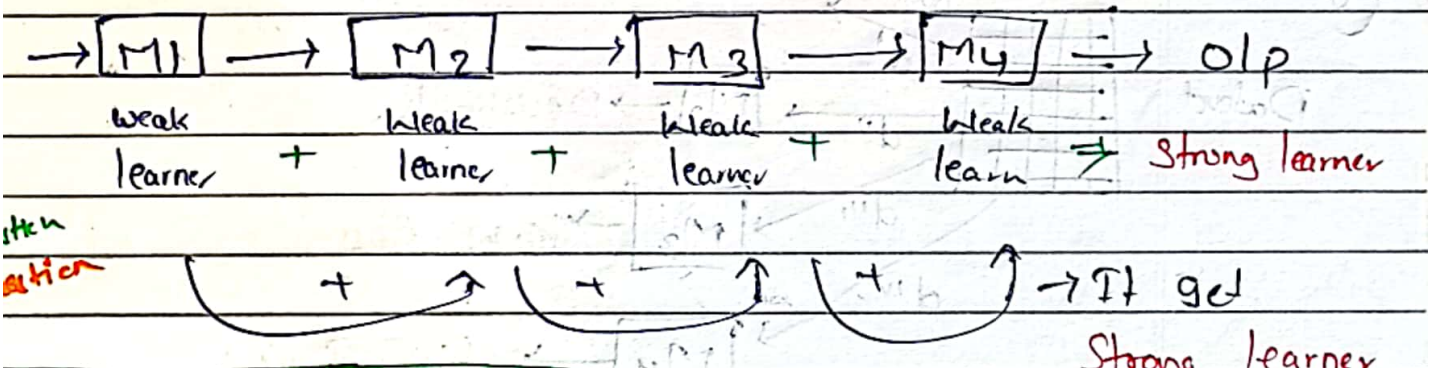
This is known as :-
{ Bootstrap Aggregating }

→ for Regression :-



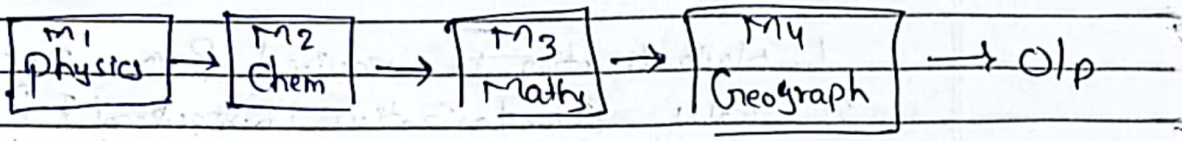
Entire mean → O/P
Mean → O/P

ii) Boosting :-



Boosting :- it is a Sequential set of all the models combined and these all models that i initialize are usually weak learners when they are combined together they become a strong learner and

⇒ Example :-



⇒

Bagging

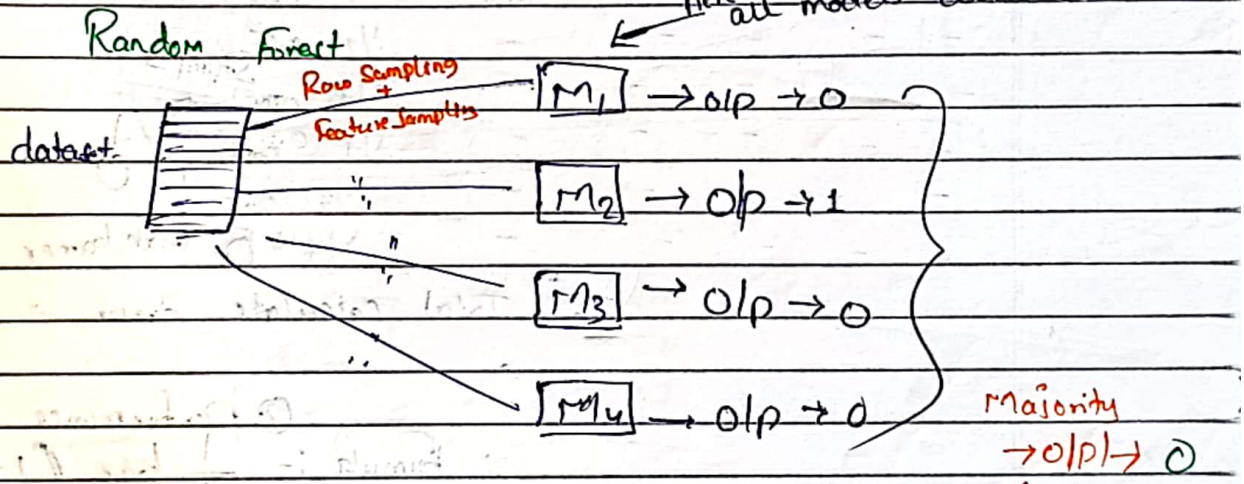
1. Random Forest classifier
2. Random Forest Regressor

Boosting

1. Ada boost
2. Gradient boost
3. Xg boost

⇒

- **Random Forest Classifier and Regressor :-**



{ ⇒ for Regression to calculate mean of model output }

Important point of in Random Forest :-

1. Normalization is required ? (or) Decision Tree ⇒ No
2. KNN (standarization is required ? ? ? ⇒ Yes
   Ecluedian distance & manhtian distance
3. Random Forest impacted by outlies ? No
4. KNN is impacted by outlies ? Yes.

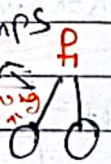
Bagging ⇒ Random Forest
you can create custom bagging.

* Boosting
is AdaBoost :- (Adaptive Boosting) :-

I have Dataset like

F1	F2	F3	F4	O/P	Weight → New update weight	We define a Weight
-	-	-	-	Yes	1/7	0.5
-	-	-	-	No	1/7	0.05 → Information gain (Gini (or) Entropy
-	-	-	-	-	1/7	0.05
-	-	-	-	-	1/7	0.05
-	-	-	-	-	1/7	0.3
-	-	-	-	-	1/7	0.5
-	-	-	-	-	1/7	0

Stumps



weak learner

Total calculate Error = $\frac{1}{7}$

② Performance of Stumps

Formula :- $\frac{1}{2}\log_e\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$

here,

$$\frac{1}{2}\log_e\left(\frac{1-1/7}{1/7}\right)$$

$$= 0.895$$

vision

(3) New Sample weight = Update all weight of my dataset

Formula = weight X e^{-Ps}

(Ps : Performance of Stumps)

$= \frac{1}{7} \times e^{-0.695}$

$= 0.05$

for,

Incorrect records = weight X e^{Ps}

$= 1/7 \times e^{0.895}$

$= 0.349$

{e = approax 2.71828}

New weight	It is 1 27	Bucket
0.05 10.6ug	Normalize weight.	0 - 0.07
0.05 10.6ug	0.07	0.07 - 0.14
0.05 10.6ug	0.07	0.14 - 0.21
0.349	0.537	0.21 - 0.747
0.05 10.6us	0.07	0.747 - 0.751
0.05 10.6ug	0.07	0.751
0.05 10.6us	0.07	
0.6ug		

$\lambda = 1$

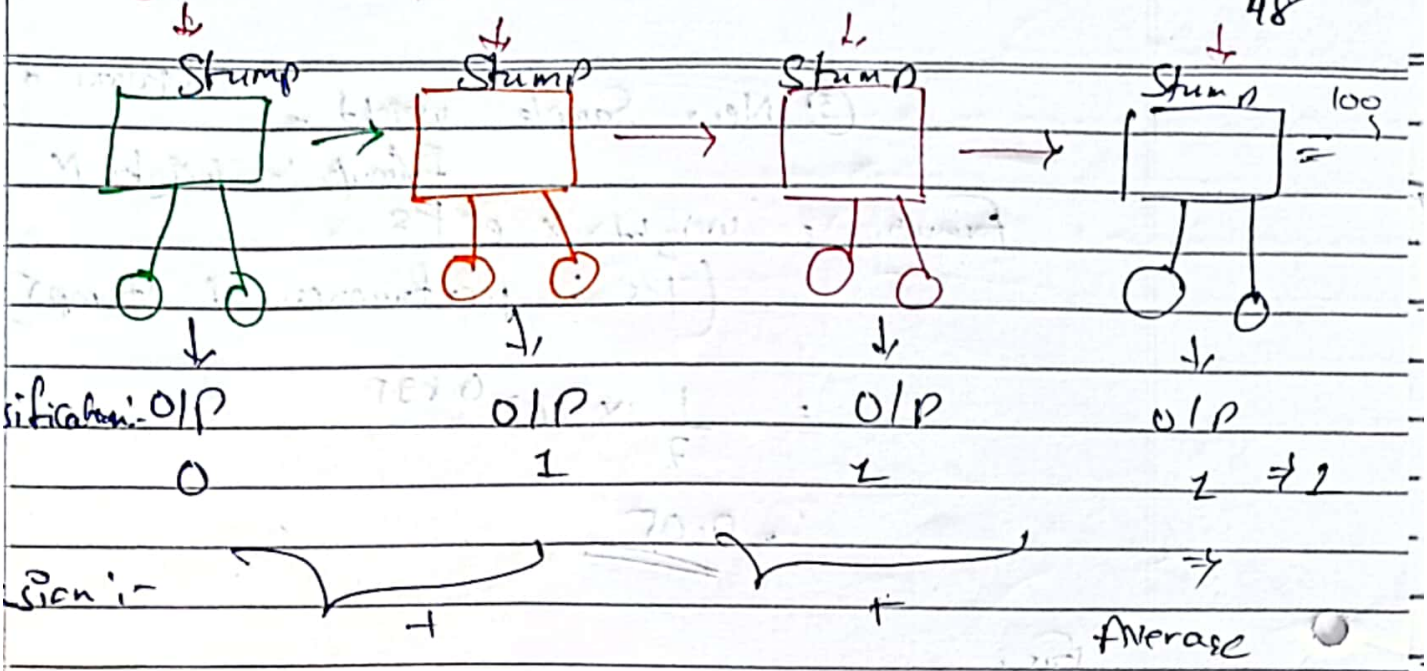
Bucket

Ex:- 0.537

0.21

0.747

Every decision Tree o/p is important



sification:-

sion:-

* Agenda :-

Un-supervised ML

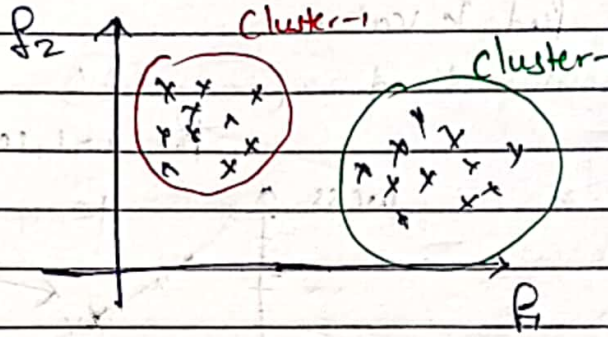
1. K Means clustering
2. Hierarchical Clustering
3. Silhoutte Score
4. DBScan Clustering

} Unsupervised ML → We don't have specific O/P.

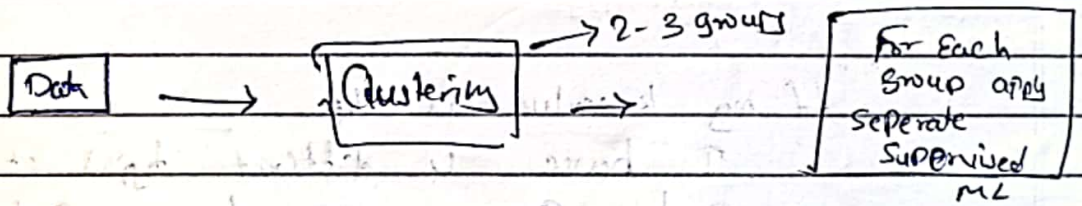
⇒

K-means clustering :-

Suppose we have datapoints



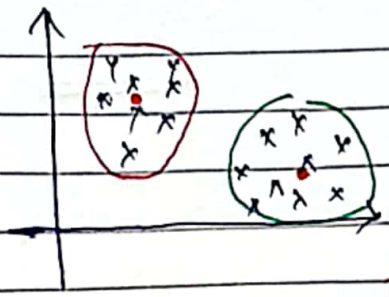
⇒ Custom Ensemble Techniques :-



K-Means

↑

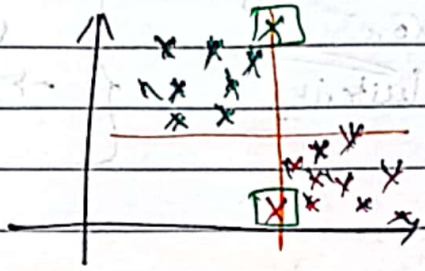
K = Centroids



Centroid

Step-1 :- Initially k values => Suitable . let k = 2

Step-2 :- Initialize k number of centroids

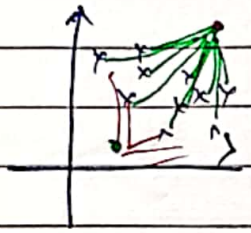


Step-3 :- Compute average to update centroid

=> How we find k value

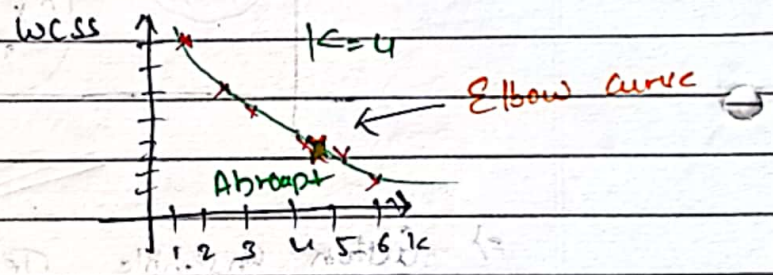
Elbow method (k value)

for i = 1, 10.



k = 1

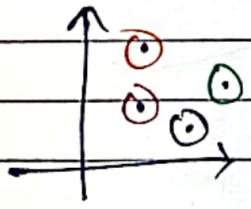
k = 2



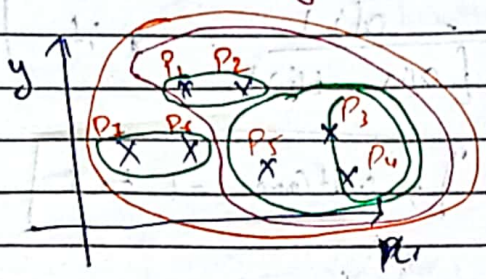
(WCSS :- within Cluster Sum of Square)

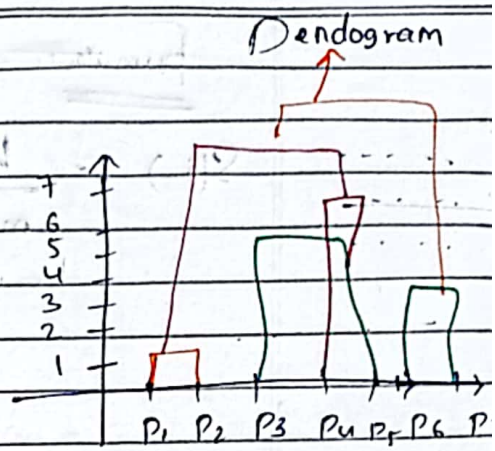
If my k value is 4.

I have 4 different types of group and Every group have Centroid



2. Hierarchical clustering :-





{ You need to find the logest vertical line That has no horizontal line past through it }

Q. Maximum Time is taken by K-means or Hierarchical Clustering ??

→ Hierarchical Clustering will taking timing more time

→ If dataset is small go with Hierarchical clustering

→ If dataset is large go with Clustering.

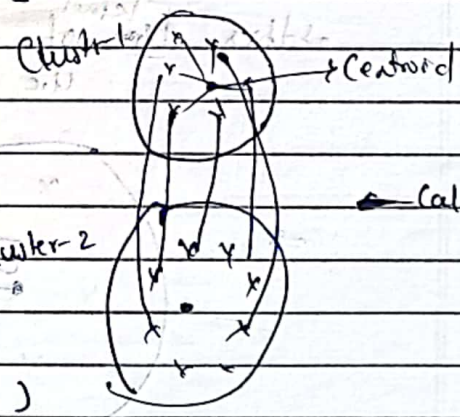
* Validate clustering models :-

Silhouette clustering

We calculating Average of Distance

a(i)

b(i)



Good cluster :-

$a(i) \ll b(i)$

Good model - $b(i) \gg a(i)$

Formula:-

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad \text{if } |C_I| > 1$$

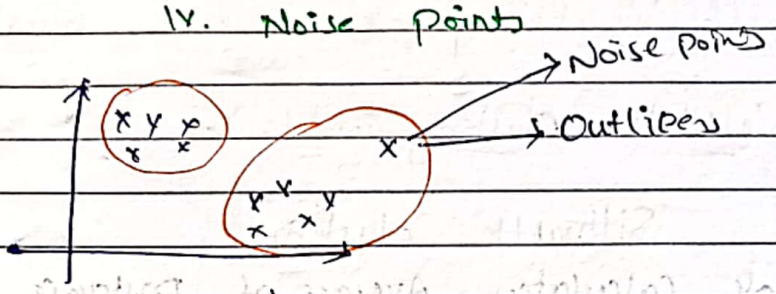
⇒ Outcome -1 to 1

* D.B.Scan Clustering :-

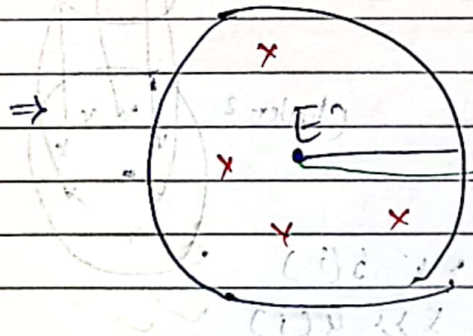
Density - Based Spatial Clustering of Applications with Noise (DBSCAN)

i. Min points
ii. Core point
iii. Border point
iv. Noise Points

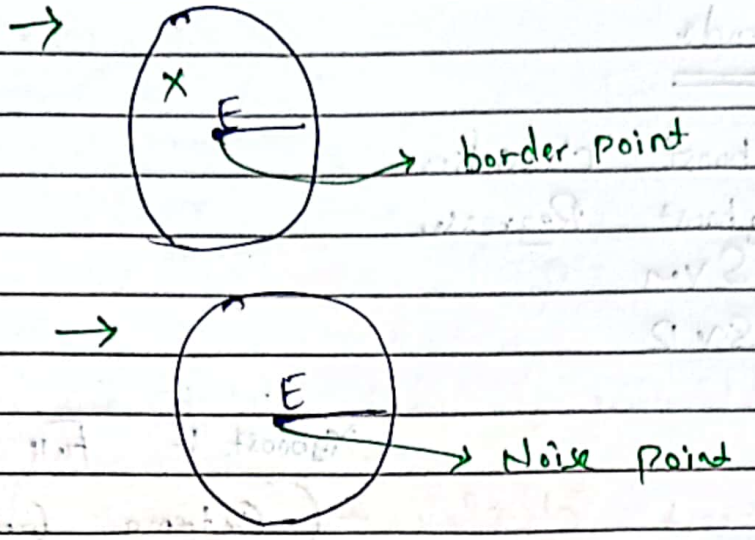
* Epsilon (radius of circle)



⇒ this kind of algorithm Skip the Outliers we use DBSCan



vision


X
E
border point
E
Noise point

18/01/25

Day-7

* Agenda

1. Xgboost classifiers
2. Xgboost Regressor
3. SVM
4. SVR

Xgboost :- Full Form :-

1. Xgboost classifiers (- Extreme Gradient Boosting) :-

⇒ Dataset :-

Salary	Credit	Approval	Residia (Approval - Base model)	
≤ 50	Bad	0	-0.5	(0 - 0.5)
≤ 50	Good	1	0.5	(1 - 0.5)
≤ 50	Good	1	0.5	(1 - 0.5)
> 50	Bad	0	-0.5	(0 - 0.5)
> 50	Good	1	0.5	(1 - 0.5)
> 50 K	Normal	1	0.5	(1 - 0.5)
≤ 80 k	N	0	-0.5	(0 - 0.5)

It is just a dummy :.

Base Model ⟶ Probability = 0.5

Step-1 :- Create a binary Decision tree using the features

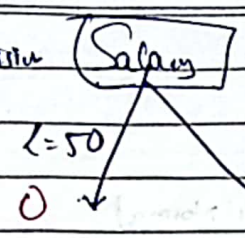
Step-2 :- Calculate the Similarity weight

Formula :- $\frac{\Sigma (\text{Residia})^2}{\Sigma (\text{Probability}(1-\text{Pr})) + \lambda}$

Step-3 :- Information gain

vision

Binary Decision Tree



[-0.5, 0.5, 0.5, 0.5] [-0.5, 0.5, 0.5]

Now, we calculate Similarity weight :-

<=50 :-

$$= \frac{(-0.5 + 0.5 + 0.5 + 0.5)^2}{0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5)}$$

$$= 0$$

Now,

>50 :-

Similarity weight = $\frac{(-0.5 + 0.5 + 0.5)^2}{0.5(1-0.5) + 0.5(1-0.5) + 0.5(1-0.5)}$

$$= \frac{0.25}{0.75} = \frac{1}{3} = 0.33$$

Now, we calculate Information gain

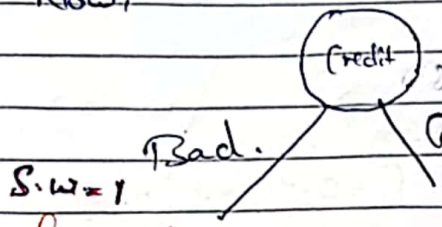
Similarity weight = $\frac{0.25}{1.75} = \frac{1}{7} = 0.142$

We Calculate Information gain

Information = 0 + 0.33 - 0.14

= 0.19

Now,



Now,

Similarity weight :-

$$\frac{0.25}{1.5(1-0.5)} = \frac{0.25}{0.25} = 1$$

Now,

$$\frac{0.25}{0.75} = 0.33$$

$$\Rightarrow 1 + 0.33 - 0 \Rightarrow 1.33$$

How we calculate the real probability :-

$$\log(p) \quad \log\left(\frac{p}{1-p}\right) = \log\left(\frac{0.5}{0.5}\right) = 0$$

$$\sigma\left[0 + \alpha(1)\right] \qquad \alpha = 0.01$$

α → Learning rate

$\Downarrow$

O/P 0 to 1

New Record O/P

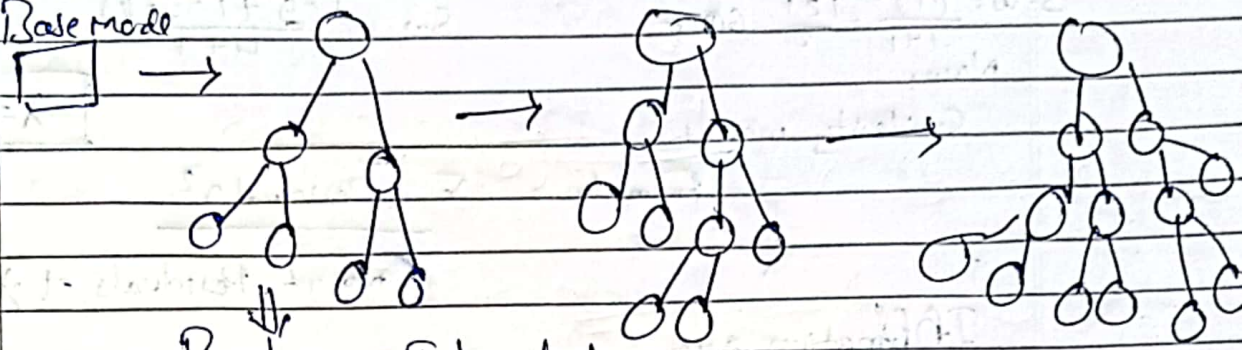
$$\sigma\left[0 + \alpha_1(DT_1) + \alpha_2(DT_2) + \alpha_3(DT_3) + \alpha_4(DT_4) + \cdots \alpha_n(DT_n)\right]$$

Xgboost → Black box model.

λ ⇒ Cross validation

$0 + \lambda_1(DT_1) + \lambda_2(DT_2) + \dots + \lambda_n(DT_n)$

Base model



Based on Independent feature.

*

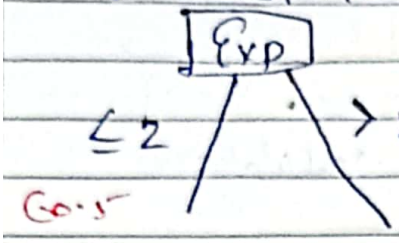
Xgboost Regressor :-

Exp	Gap	Salary	→ O/P Residual
2	Yes	40 k	-11k
2.5	Yes	42k	-9k
3	No	52 k	1 k
4	No	60 k	9 k
4.5	Yes	62 k	11 k

Now, we create a Base model :-

Base model → Avg 51 k

[-11, -9, 1, 9, 11] → Sw = $\frac{(-11-9+1+9+11)^2}{5+1} = \frac{1}{6}$ =



∴ [-11] [-9, 1, 9, 11]

w = $\frac{(11)^2}{1+1} = \frac{121}{2} = 60.5$ Sw = $\frac{(-9+1+9+11)^2}{4+1} = \frac{144}{5} = 28.8$

Now,

$\lambda = 0$

Similarity weight

Formula = $\frac{\Sigma (\text{Residual})^2}{\text{No. of Residuals} + \lambda}$

Information gain

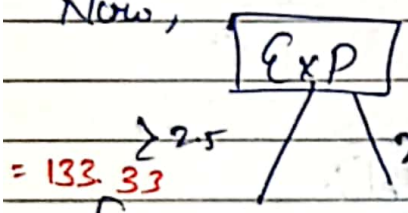
= [illegible] / 6

= [illegible]

= 60.5 + 28.8 − 1/6

= 89.13

Now,



$Sw = \frac{(-11-9)^2}{2+1}$

$= \frac{(-20)^2}{3} = 133.33$

$Sw = \frac{(1+9+11)^2}{1+3}$

$\frac{(21)^2}{4}$

$441/4 = 110.25$

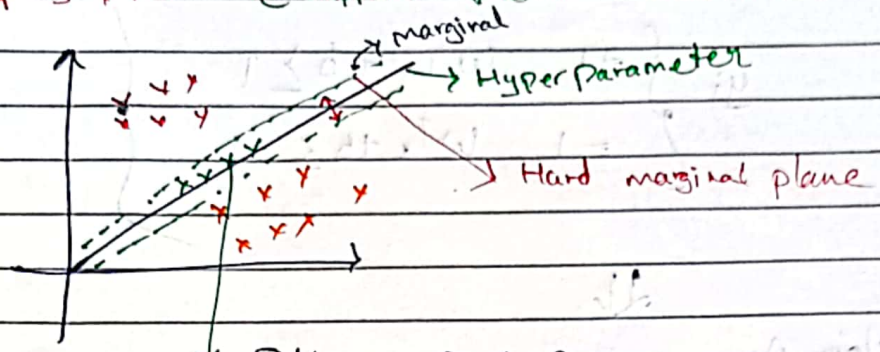
Now, here,

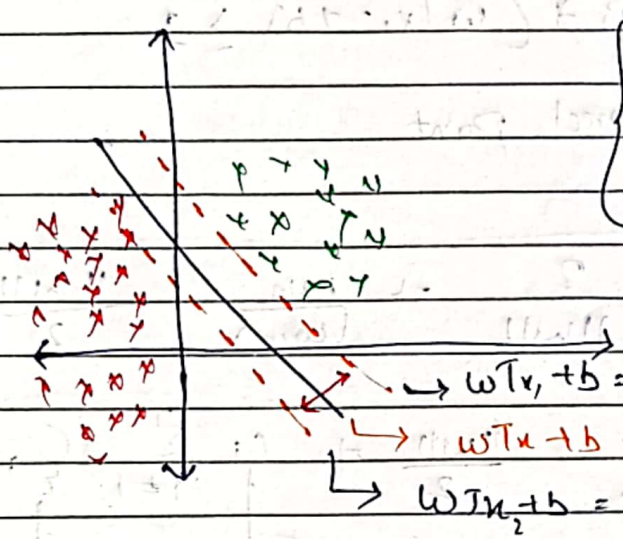
$51 + \alpha_1(-1) + \alpha_2 + 0 T_2 + \alpha_n + 0 T_3$

Now, here

$51 + \alpha_2(7)$

* SVM (Support vector machine) :-





$y = mx + c$

$ax + by + c = 0$

$w^T x + b = 0$

Same

$w^T x_1 + b = -1$

$w^T x_2 + b = 1$

(Vector + magnitude)

$- \quad - \quad +$

$$\frac{w^T(x_1 - x_2)}{|w|} = \frac{2}{|w|}$$

Maximize cost function

Maximize $\frac{2}{\|w\|}$ ⇒ Marginal plane

(w, b)

Such that,

$$y_i \begin{cases} +1 & w^T x + b \geq 1 \\ -1 & w^T x + b \leq -1 \end{cases}$$

Main Aim

$$y_i * (w^T x_i + b) \geq 1$$

For Correct Point

$$\underset{(w,b)}{\text{Maximize}} \quad \frac{2}{\|w\|} \Rightarrow \underset{(w,b)}{\min} \quad \frac{\|w\|}{2}$$

$$\underset{(w,b)}{\min} \quad \frac{\|w\|}{2} + C_i \sum_{i=1} \xi_i$$

$\sum_{i=1} \xi_i$ → Summation of the distance of the wrong point

C_i → How many errors we can have.

* SVM Kernel it → Convert 2D to 3 Dimensional

Split it