ENSEMBLED METHODS

ADA BOOST

Mathematical Intuitions –

- ✓ Boosting is an ensembled Technique where we create models sequentially.
  - ↑ Combined multiple models
- ✓
- ✓
  - ① Ada Boost
  - ② Gradient Boost
  - ③ XGB

Ada Boost :-

Bagging ⇒ ✓ parallel creation of models
✓ Models are independent

Bootstrap → Aggregation

Eg:- Random Forest

→ ✓ Sequential implementation ⇒ $y\_Pred = DT_1\alpha_1 + DT_2\alpha_2 + DT_3\alpha_3 + \cdots + DT_n\alpha_n$

✓ Additive modelling → Addition of all models

✓ models are dependent on each other

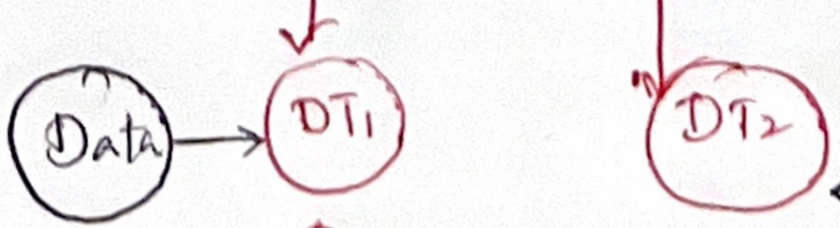
α = weight

Ada Boost :- $= h_1(\theta)\alpha_1 + h_2(\theta)\alpha_2 + h_3(\theta)\alpha_3 + \cdots + h_n(\theta)\alpha_n$

✓ Classification
✓ Regression

$h(\theta)$ = function ← Decision Tree

α = weight



I Stump → Weak learner

Training Accuracy → underfitting

Wrong Classification

$WL_1 + WL_2 + WL_3 \cdots$ = Strong Learner

Low bias (overfitting)
high variance

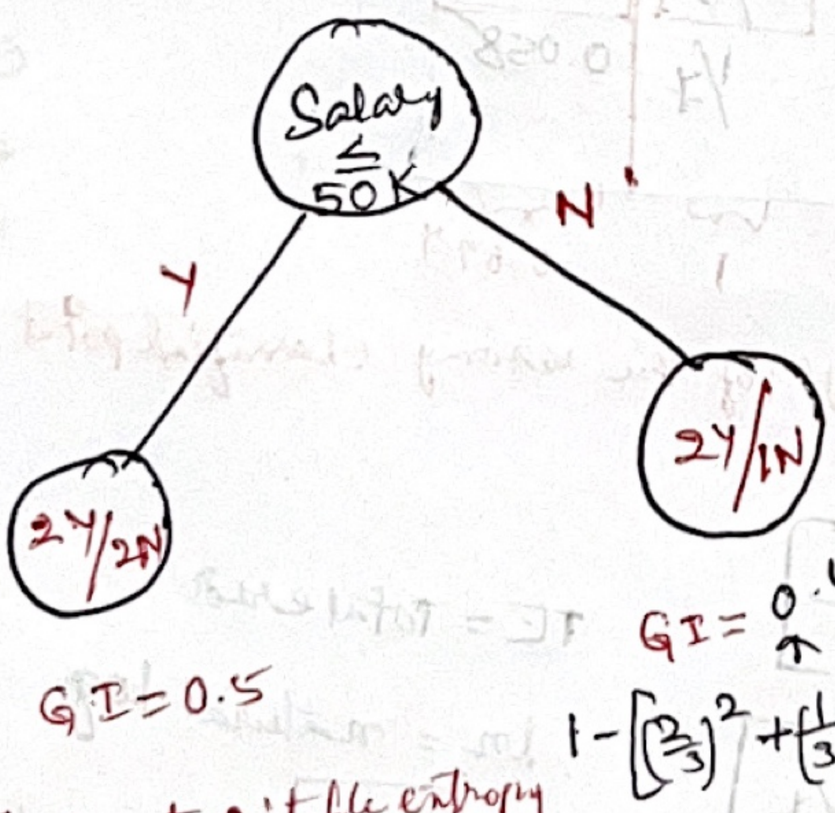
high bias (underfitting)
high variance

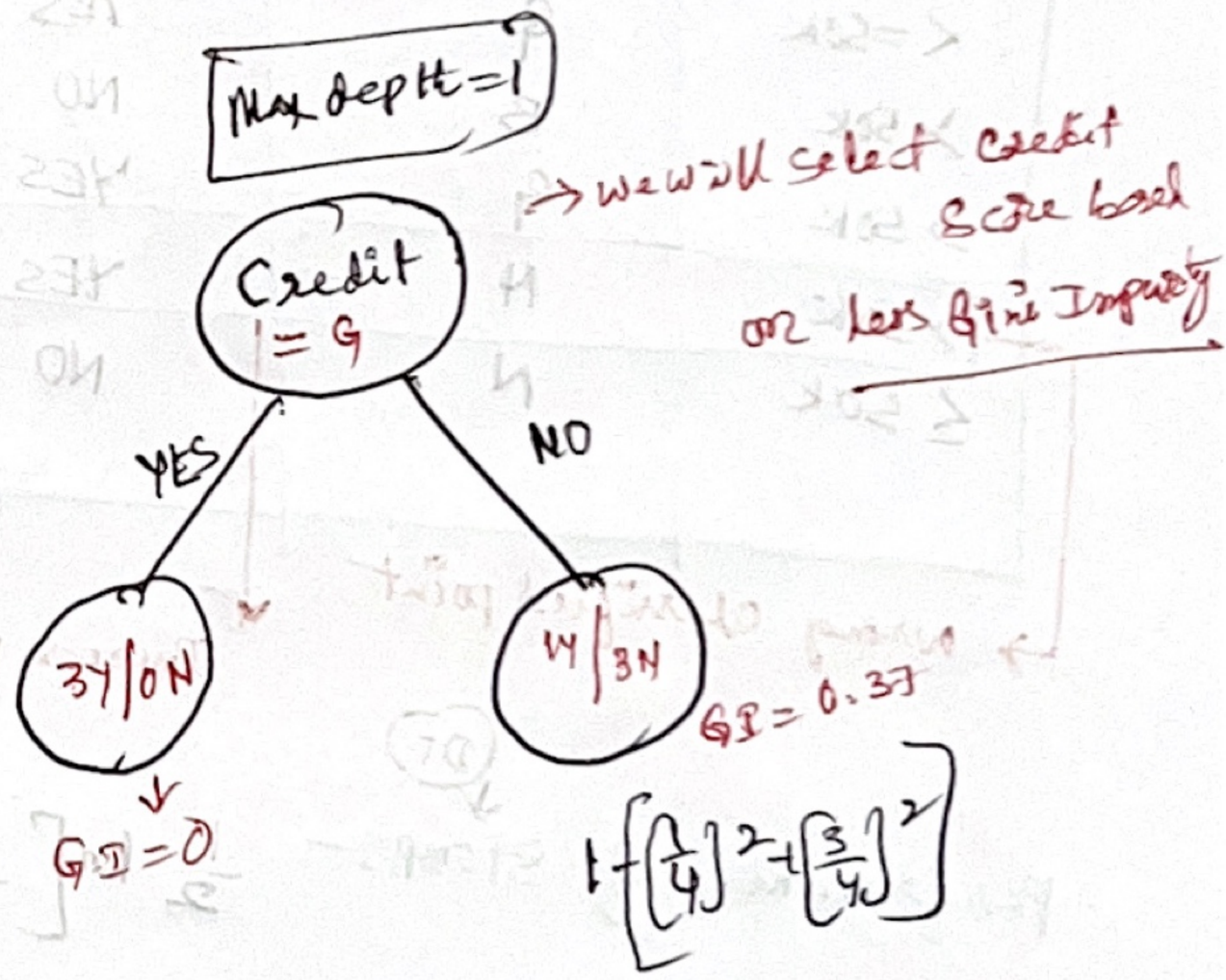
low bias
low variance
} Boosting

Data set

Salary	Credit Score	Approval
≤ 50K	B	NO
≤ 50K	G	YES
≤ 50K	G	YES
≥ 50K	B	NO
≥ 50K	N	YES
≥ 50K	G	YES
≤ 50K	N	NO

Step 1:- Create a Decision Tree





→ we will select credit score based on less Gini Impurity

Step 2:- Find out Suitable entropy
(Algorithms can be used)

ID3 = Entropy → Information Gain ← multi class classification

CART = Gini Impurity ← Support only Binary classification

Gini Impurity ⇒ $\sum_{i=1}^{n} (1 - P_i^2)$

$\left[\frac{4}{9} + \frac{1}{6} \quad \frac{4+1}{9} = \frac{5}{9}\right]$

$\left[\frac{6}{16} + \frac{9}{16} = \frac{10}{16}\right]$

$1 - \frac{10}{16} = \frac{6}{16} = 3/8$

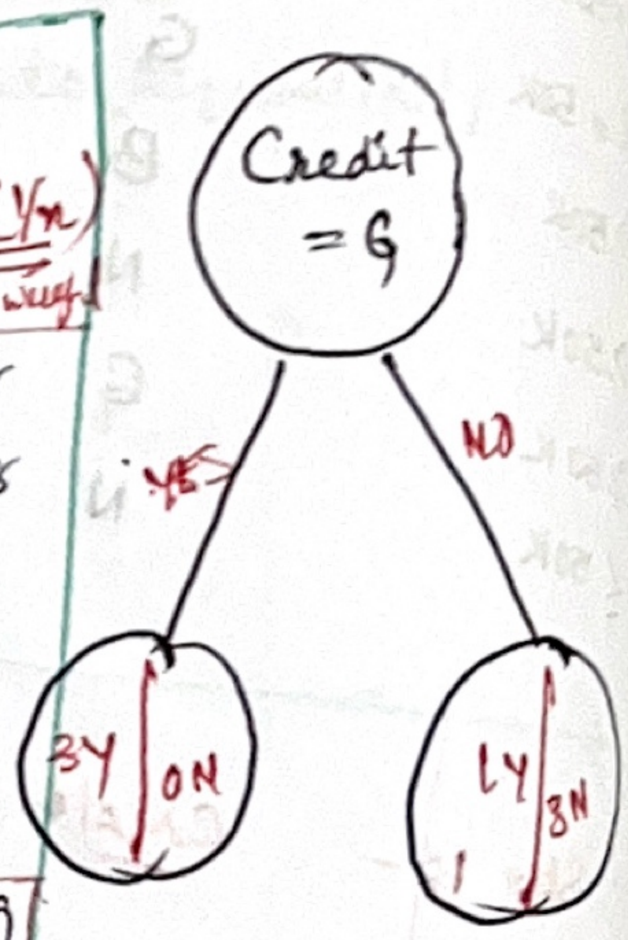
$\frac{1}{1} - \frac{5}{9} = \frac{9-5}{9} = \frac{4}{9}$

Step 3:-

① Decision Tree with maximum one (Layer) (Depth) is called 'Stomp' It is alre

② Weight — Assign Sample weight

Salary	Credit_Score	Target column: Approval	n = no of rows: Sample Weight (1/n)	Updated weight
<=50K	B	NO	1/7	0.058
<=50K	G	YES	1/7	0.058
<=50K	G	YES	1/7	0.058
>50K	B	NO	1/7	0.058
>50K	G	YES	1/7	0.058
>50K	N	YES	1/7	0.349
≤50K	N	NO	1/7	0.058



→ Wrong classified point

Increase the weight of the wrong classified point

Performance of Stomp:- (DT)

$$\frac{1}{2}\ln\left[\frac{1-TE}{TE}\right]$$

TE = Total error

$$= \frac{1}{2}\ln\left[\frac{1-1/7}{1/7}\right]$$

ln = natural log

$\log_e$ Log with base e

performance of Stomp $= \frac{1}{2}\ln[6] \approx 0.896$

$\alpha_1 = 0.896$

→ Now increase the weight of wrong classified point and decrease the weight of correctly classified points.

② Calculate the updated weight

Formula for correct classified points

$$\text{Weight} \times e^{-\text{Performance of stomp}}$$

$$\Rightarrow \frac{1}{7} \times e^{-(0.896)}$$

$$\Rightarrow 0.058$$

Weight × $e^{\text{Performance of Stump}}$

For wrongly classified points

$= \frac{1}{7} \times e^{(0.896)}$

$= 0.349$

Each records assign some bin size

Normalized weights

Bucket (Bin) size

Cumulative Sum

Normalized weights	Bucket (Bin) size	Cumulative Sum
0.083	0 – 0.083	— First bin size
0.083	0.083 – 0.16	— Second bin size
0.083	0.16 – 0.24	and soon
0.083	0.24 – 0.32	
0.083	0.32 – 0.40	
0.50	0.40 – 0.90	
0.083	0.90 – 0.96	

1

Based on bucket size only we will select wrong classified points

Range = [0–1] ⇒ 0.2, 0.4, 0.5, 0.6, 0.7

↑ Random selection

How to select points for 2nd DT

Data set 2

Salary	Credit score	Approval
<=50K	N	N
>50K	N	Y
>50K	N	Y
>50K	N	Y

Here chances for selection of wrong classified points is higher

$\frac{0.896}{\alpha 1} + \frac{0.650}{\alpha 2} + \frac{0.24}{\alpha 3} + \frac{0.30}{\alpha 4}$ (NO)

↑ additive learning

Performance of Stump :- 1.136 (yes) + 0.350 (NO)

O/P = yes

How many decision Trees → Hyperparameter → n_estimators = 50 (Default)

estimator = model

learning rate

GRADIENT BOOSTING (Classification & Regression)

sklearn.ensemble.GradientBoostingClassifier

Sequence of the Model

$y\_pred = M_0 + M_1 + M_2 + M_3 + M_4 + \cdots + M_n$

M_0 → Base Model → Average Value (leaf)

M_1, M_2, M_n → Decision Trees

$+$ learning rate (α) $+$ learning rate (α) $+$ learning rate (α)

$\alpha = [0, 1]$

Data Set

IQ	CGPA	Salary	Admission	Prediction
90	8	82	66	6.6
100	7	62	6.6	
110	6	52	6.6	
120	9	102	66	6.6
80	5	42	6.6	

$\frac{8+6+5+10+4}{5}$

Residual = Loss(Red) = (actual − pred)

1.4
−0.6
−1.6
3.4
−2.6

$(x_i, y_i)_{i=1}^{n}$ → n features

$L(y, F(x))$ → Loss Function

1. Initialize model with a constant value

$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$

2.

Gradient Boosting Regression

In [24]:
```
import pandas as pd
df=pd.DataFrame({
    'iq':[90,100,110,120,80],
    'cgpa':[8,7,6,9,5],
    'salary':[3,4,8,6,3]
})
df.head()
```

Out[24]:

	iq	cgpa	salary
0	90	8	3
1	100	7	4
2	110	6	8
3	120	9	6
4	80	5	3

In [25]:
```
import numpy as np
base_model=np.mean(df['salary'])
```

In [26]:
```
df['residual1']=df['salary']-base_model
```

In [27]:
```
df['residual1']
```

Out[27]:
```
0   -1.8
1   -0.8
2    3.2
3    1.2
4   -1.8
Name: residual1, dtype: float64
```

In [28]:
```
from sklearn.tree import DecisionTreeRegressor
model=DecisionTreeRegressor()
```

In [29]:
```
df['base_model']=base_model
```

In [30]:
```
df
```

Out[30]:

	iq	cgpa	salary	residual1	base_model
0	90	8	3	-1.8	4.8
1	100	7	4	-0.8	4.8
2	110	6	8	3.2	4.8
3	120	9	6	1.2	4.8
4	80	5	3	-1.8	4.8

In [31]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df[['iq','cgpa']],df.residual1,test_size=0.2,random_state=42)
```

In [32]:
```
model.fit(X_train,y_train)
```

Out[32]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [33]:
```
model.score(X_train,y_train)
```

Out[33]:
```
1.0
```

In [34]:
```
from sklearn import tree
import matplotlib.pyplot as plt
#fig,ax=plt.subplot(fisize=(10,5))
plt.figure(figsize=(10,5))
tree.plot_tree(model, filled=True)
plt.show()
```