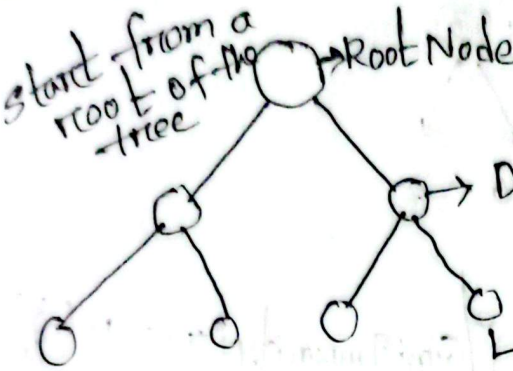
Machine Learning: (Final Topic)

Decision Tree: (Gini Impurity, Entropy, Info Gain)

↳ [1 - (Prob of yes)$^2$ - (Prob of No)$^2$]




↳ (when subnode splits into further nodes)

↳ (Node that doesn't split)

We stop, splitting it there are no decision rules left to the group impurity.

Adv: Easy to interpret, straight-forward visualization.

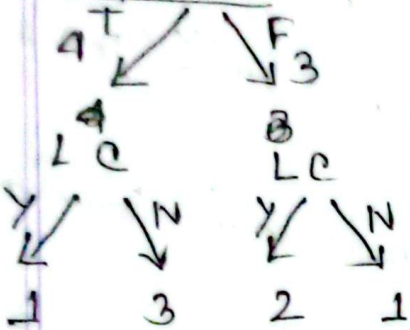
Disadv: ↳ Single DT more prone to overfitting.

↳ Early stop specifying the layer.

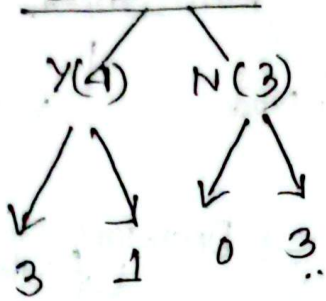
| Loves Popcorn | Loves Soda | Age | Loves Cool as ice |
|---|---|---|---|
| Y | Y | 7 → Y | N |
| Y | N | 12 → Y | N |
| N | Y | 18 → N | Y |
| N | Y | 35 → N | Y |
| Y | Y | 38 → N | Y |
| Y | N | 50 → N | N |
| N | N | 83 → N | N |

For (Y, Y, 15) what should be the output?

→ Pop Corn




Loves soda




Gini Impurity (T) $= 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2$

$= 0.375$

" (F) $= 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2$

$= 0.444.$

weighted avg of love popcorn $= \left(\frac{4}{7} \times 0.375\right) + \left(\frac{3}{7} \times 0.444\right)$

$= 0.405.$

Gini Impurity (T) $= 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2$

$= 0.375$

" (F) $= 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2$

$= 0$

w of loves-soda $= \left(\frac{4}{7} \times 0.375\right) + \left(\frac{3}{7} \times 0\right)$

$= 0.214.$

* For numerical value,

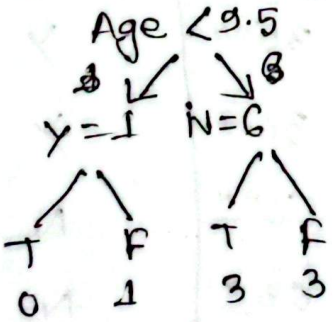
(i) Firstly, Row থেকে ছোট থেকে বড় সাজাব।

(ii) Avg age of all adjacent people.

(iii) Calculate gini impurity for each avg age.

Age

7 → 9.5
12 → 15
18 → 26.5
35 → 36.5
38 → 44
50 → 66.5
83



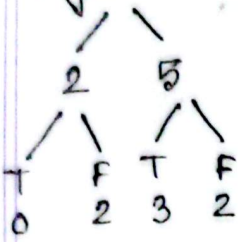

G.Im $= 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0.$

" $= 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$

w of age < 9.5) $= \left(\frac{1}{7} \times 0\right) + \left(\frac{6}{7} \times 0.5\right)$

$= 0.429$

Age < 15

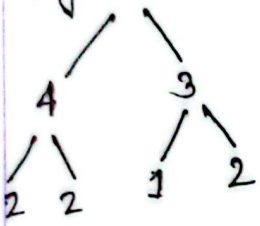



G.Im = $1-(\frac{0}{2})^2-(\frac{2}{2})^2=0$

u = $1-(\frac{3}{5})^2-(\frac{2}{5})^2=0.48$

w of (age<15) = $(\frac{2}{7}\times 0)+(\frac{5}{7}\times 0.48)$
= 0.343

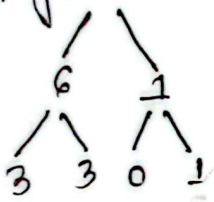
Age < 36.5 G.Im = 0.5




u = 0.414

w of Ag<36.5 = 0.476

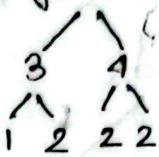
Age < 66.5 G.Im = 0.5




u = 0

w of Age<66.5 = 0.428

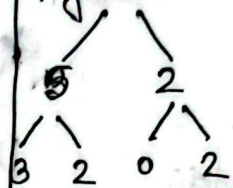
Age < 26.5 G.Im = 0.499




u = 0.5

w of age<26.5 = 0.476

Age < 44 G.Im = 0.48

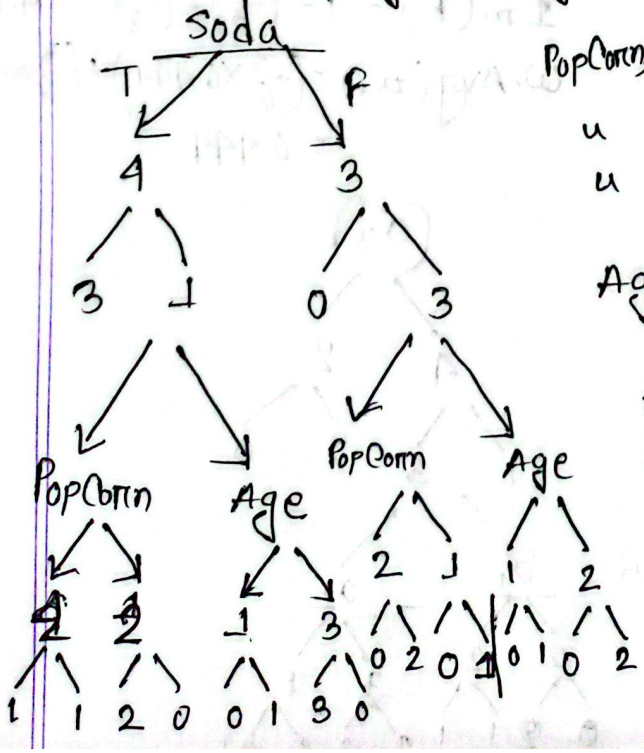



u = 0

w of ag<44 = 0.343.

∴ Among those value, lowest 0.343 [Same হলে যে কোনো নিলেই হবে]

Lower impurity among all columns – Soda (Top of the tree).




PopCorn, Im 1 = $1-(\frac{1}{2})^2-(\frac{1}{2})^2=0.5$

u Im 2 = 0

u Avg = $(\frac{2}{4}\times 0.5)=0.25$.

Age, Im 1 = 0

u = 0.

Age, Avg = 0.

PopCorn, Im 1 = 0

Im 2 = 0.

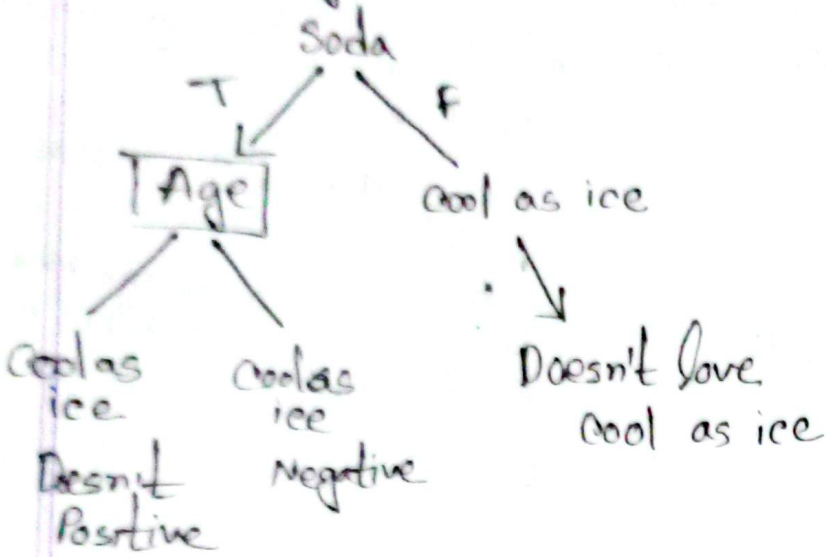
Avg = 0.

Age, Im 1 = 0.

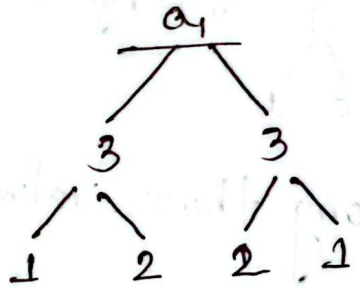
Im 2 = 0.

Avg = 0.

যেহেতু Age টেবিল, তাই আমরা 2nd step simplify.




| $a_1$ | $a_2$ | $a_3$ | Class |
|---|---|---|---|
| T | Hot | High | N |
| T | H | H | N |
| F | H | H | Y |
| F | Cool | Normal | Y |
| F | C | H | N |
| T | H | N | Y |

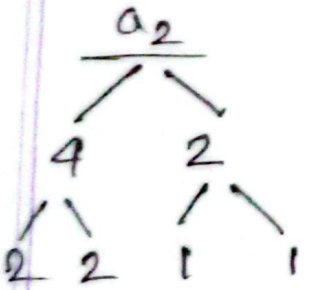



$Im(T) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.444$

$Im(F) = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0444$

$\omega.Avg(a_1) = \left(\frac{3}{6} \times 0.444\right) + \left(\frac{3}{6} \times 0.444\right)$

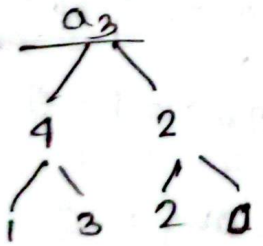
$= 0.444$




$Im(T) = 0.5$

$Im(F) = 0.5$

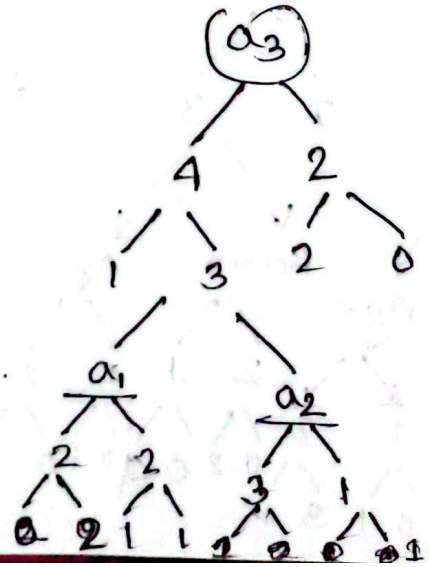
$\omega.Avg(a_2) = 0.5$




$Im(T) = 0.375$

$Im(F) = 0.$

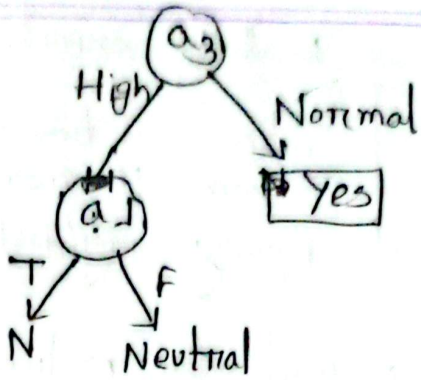
$\omega Avg(a_3) = 0.25.$



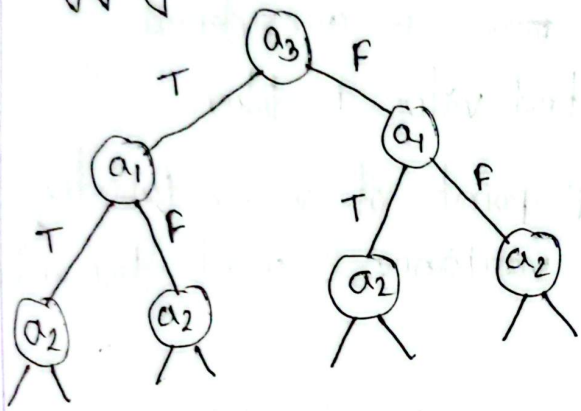
T, H, H,

$a_1$, Im(T) = 0
u(F) = 0.5
w. Avg ($a_1$) = 0.25

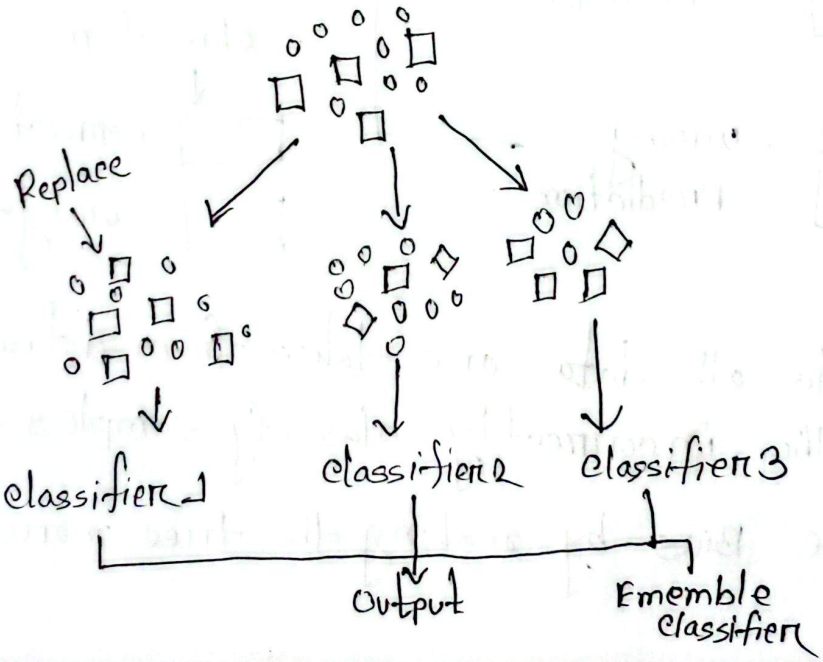
$a_2$, Im(T) = 0.444
u(f) = 0
w Avg ($a_2$) = 0.33.



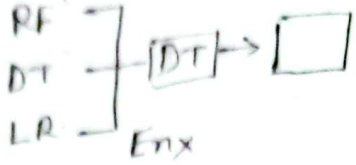

Bagging:




Bootstrap Aggregation:
sampling with replacement from the original data.



* Bagging → Parallelly execute
  → Takes less time to train the model
  → Equally weighted all

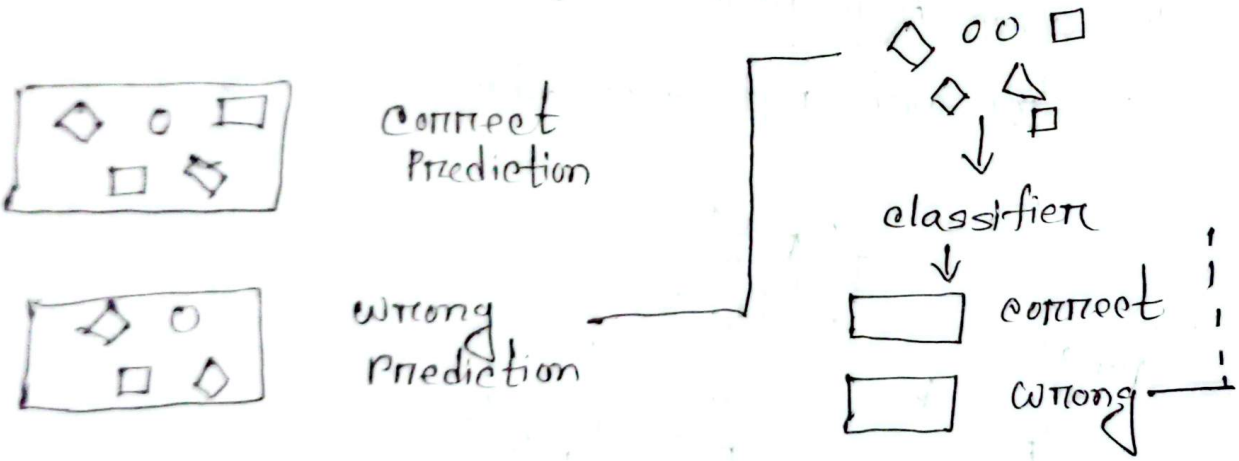



* Bagging is used for reducing variance.
* Prevents overfitting
* Improves robustness.

v for classification majority voting is considered.
v for regression, avg weighted value is taken.

From bias variance trade off point of view, Boosting starts with low bias + high variance & work towards reducing variances.

Works best with strong & cplx model




* No extra data all data are taken from dataset
* weights up the incorrectly classify samples.

Boosting → Reduce Bias by making the tree more complicated.

✓ Ada Boost
XGBoost

| CGPA | Predicte Job offiere | Actual Job offieere |
|---|---|---|
| >=9 | Y | Y |
| <9 | N | Y |
| >=9 | Y | N |
| <9 | N | N |
| >=9 | N | Y |
| <9 | Y | Y |
| >=9 | Y | |
| >=9 | | |

Adaboost steps:

(I) Assigning weights
(II) Creating all decision stumps
(III) Choosing the best decision stumps.
(IV) Calculating weight
(V) Adjusting "
(VI) Normalizing "
(VII) Preparing data for next step.
(VIII) Assigning next weight.

* Initial weight assigned to each item $= \frac{1}{n} = \frac{1}{6}$.

(i) Compute weighted error.

Error $\rightarrow \sum_{j=1}^{N} w_i(d_j)\, wt(d_j)$ → 0 for correct prediction, 1 " wrong "

$Error_{corr} = 4 \times 0 \times \frac{1}{6}$

" $Neg = 2 \times 1 \times \frac{1}{6} = 0.333$

(ii) Compute the weight of each weak learner

$\alpha_{CGPA} = \frac{1}{2}\left(\frac{\ln(1-Error)}{Error}\right) = \frac{1}{2}\left(\frac{\ln(1-0.333)}{0.333}\right)$

$= 0.347.$

(iii) Calculate Normalizing factor

$Z_{CGPA} = wt(correct) \times$ num of correct classification $\times e^{-\alpha}$

$+ wt(wrong) \times$ num of wrong classification $\times e^{\alpha}$

$= \left(\frac{1}{6} \times 4 \times e^{-0.347}\right) + \left(\frac{1}{6} \times 2 \times e^{0.347}\right)$

$= 0.9428.$

(iv) Update the weight of all data instances,

$wt(d_j)_{i+1} = \frac{wt(d_j) \text{ of correct Instance} \times e^{-\alpha}}{Z_{CGPA}}$

$= \frac{\frac{1}{6} \times e^{-0.347}}{0.9428} = 0.1249.$

$wt(d_j)_{i+1} = \frac{wt(d_j) \text{ of Incorrect Instance} \times e^{\alpha}}{Z_{CGPA}}$.

⊖ $= \frac{\frac{1}{6} \times e^{0.347}}{0.9428} = 0.2501.$

| Sample | Actual | Predicted |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 0 | 0 |
| 3 | 1 | 0 |
| 4 | 1 | 1 |
| 5 | 1 | 0 |

(i) Initially, we are taking random weight $= \frac{1}{n} = \frac{1}{5} = 0.2$

(ii) Compute weighted error

always 0 for correct prediction as $\mathbb{1}_i(d_j) = 0$,

$\text{Error}_{\text{wrong}} = 3 \times 1 \times 0.2 = 0.6.$

(iii) Compute learners weight

$\alpha = \frac{1}{2} \frac{\ln(1-\text{err})}{\text{err}} = \frac{1}{2} \frac{\ln(1-0.6)}{0.6} = -0.203.$

(iv) Compute ~~Normalizatio~~ Normalizing factor

$= 0.2 \times 2 \times e^{-(-0.203)} + 0.2 \times 3 \times e^{-0.203} = 0.9798.$

(v) Update the weight of all data instances.

⊕ → $\frac{0.2 \times e^{-0.203}}{0.9798} = 0.245$

⊖ → $\frac{0.2 \times e^{-0.203}}{0.9798} = 0.167$

In this math $\alpha$⊖

This means the weak learners performs worse than guessing error. So, Adaboost will effectively invert its predictions in future & down weight its import.

Ensemble :

- Bagging — Parallel — Reduce variance
- Boosting — Sequential — Reduce Bias
  - XGBoost
  - AdaBoost → compute weight of weak classifier, $\alpha = \frac{1}{2} \ln\left(\frac{1-\text{err}}{\text{err}}\right)$.
- stacking

XGBoost :

For classification :

(I) Probability based on binary classification

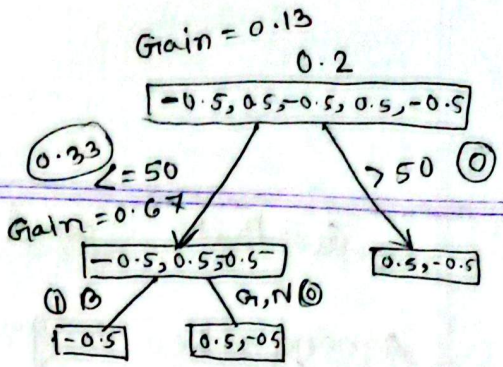
(II) Classification → Base value = $\log\left(\frac{\text{Prob}}{1-\text{Prob}}\right)$

(III) Residual = Act − Pred

(IV) similarity weight = $\Sigma(\text{Residual})^2 / \Sigma P_r(1-P_r)$

(V) calculate gain = All leaf similarity − similarity of Root.

| Salary | Credit | Approval | Res |
|---|---|---|---|
| <= 50 | Bad | 0 | −0.5 |
| <= 50 | Good | 1 | 0.5 |
| > 50 | Bad | 0 | −0.5 |
| > 50 | Good | 1 | 0.5 |
| <= 50 | Normal | 0 | −0.5 |




for salary:

(i) Probability = 0.5 as Binary.

(ii) Residual = (Ac - Pred)

(iii) Base model = $\log\left(\frac{Prob}{1-Prob}\right) = \log\left(\frac{0.5}{1-0.5}\right) = \log(1) = 0.$

(iv) Similarity weight

(a) for <=50, $SW = \frac{(-0.5+0.5-0.5)^2}{\{0.5(1-0.5)\}\times 3} = 0.33.$

(b) For >50, $SW = \frac{(0.5-0.5)^2}{\{0.5(1-0.5)\}\times 2} = 0.$

(c) SW for root $= \frac{(-0.5+0.5-0.5+0.5-0.5)^2}{\{0.5(1-0.5)\}\times 5} = 0.2$

(v) Gain = All leaf similarity - similarity of Root

$= (0.33+0) - 0.2 = 0.13.$

for credit:

(i) SW of Bad $= \frac{(-0.5)^2}{\{0.5\times(1-0.5)\}\times 1} = 1.$

SW of Good, Normal $= \frac{(0.5-0.5)^2}{\{0.5\times(1-0.5)\}\times 2} = 0.$

(ii) Info Gain, $(1+0) - 0.33 = 0.67.$

Among Bad, Good, ~~m~~ normal, we took Bad onesite, other are on the other side as we have to do Binary classification. Similarity, we will calculate the combination.

G – B, N and N → G, B.

Among the 3 gain, the highest value is kept for next step.

Considering the calculated one as highest gain.

New update ~~probab~~ prob of given point (L=50, Bad) → 0.1 / 0.2 / 0.5 (let)

$= \sigma$ (Base model + $\lambda$ x s w)

$= \sigma (0 + 0.1 \times 1)$

$= \sigma (0.1)$

$= \frac{1}{1+e^{-0.1}} = 0.6.$

∴ New ~~pred~~ residual = Ac – Pro = 0 – 0.6 = –0.6.

(for 1st point)

New value becomes –0.6 instead of –0.5. similarity, the proper will continue will all new values.

Parameter

λ – Regularization Parameter.

→ λ value ↑↑ ↓↓, impact of outlier ↓↓ ↑↑.

→ Pruning ↓↓ Aggressive / ↑↑ ↓↓.

Ensemble (15)

→ Adaboost → 5

→ XGB → 5

→ theory → 5.

γ – Gamma → Threshold – Auto pruning. ctrl overfitting classification ⇒ γ = Prob (1 – Prob)

= 0.5 (1 – 0.5) = 0.25

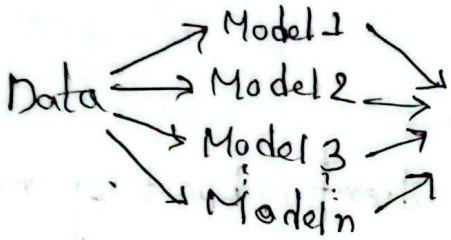
if gain < 0.25, we can prune that past from the tree, ε - How fast convergence happen (speed)
Prune (छाँट (छाना)

XG-Boost:
Just an extremely optimized gradient boosting model which by itself is a powerfull ML Algo,

why XGBoost is better
→ Missing values automatically handled.
→ Cross validation.
→ Interponates both L1 and L2 regularization technique to prevent overfitting and enhance model generalization.
→ Parallelization: At a time, more work possible.
→ Tree ~~pu~~ pruning: If possible, it prunes to make the tree smaller.

stacking:
- Ensamble method that combines heterogeneous weak learners, combined using a meta learning algorithm.
- Bagging and boosting can be used as ~~es~~ Ensamble at start of stacking.

Though ensamble is not interpretable, it can give highest accuracy. So, it is used when accuracy is our main target.




stacking steps:

Individual estimatory feed their prediction to the meta learn.

↓

Meta learner take them a ~~$~~ input features

↓

Work on how to best combine these predictions to improve the overall performance.

↓

their goal: correct the weakness or individual models by finding optimal weight.

↓

thus, ensures higher accuracy and generalization

XG Boost Regression:

(i) Base = Avg of output column.

(ii) Residual = Act - Base

(iii) Similarity weight = $\frac{\Sigma \text{Residual}^2}{\text{no of Residual} + \lambda}$

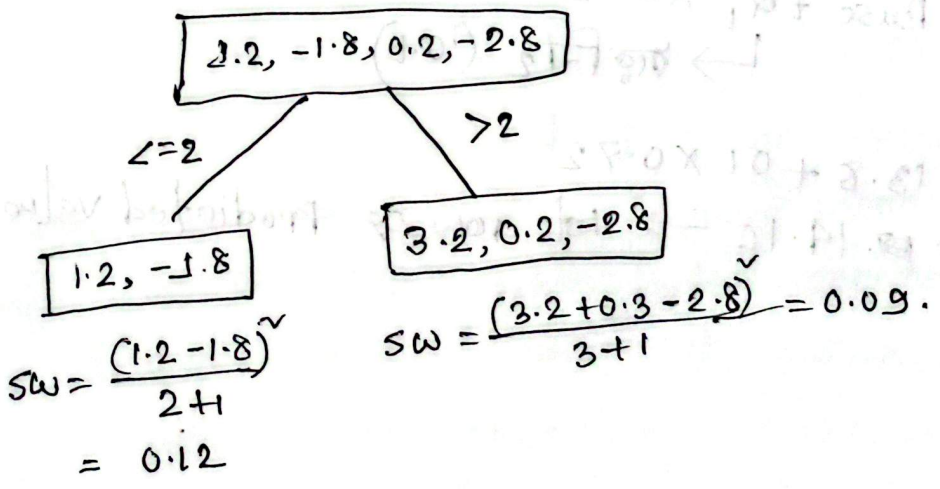
(iv) Info Gain = SW of a2 leaf - Root.

(v) Calculate new probability

| Ball | Color | size | Price | Residual |
|---|---|---|---|---|
| 1 | Red | 3 | 15 | 15 - 13.8 = 1.2 |
| 2 | Blue | 25 | 12 | 12 - 13.8 = -1.8 |
| 3 | Red | 4 | 17 | 17 - 13.8 = 3.2 |
| 4 | Blue | 35 | 14 | 14 - 13.8 = 0.2 |
| 5 | Red | 2 | 11 | 11 - 13.8 = -2.8 |

Base value = $\frac{\Sigma \text{Price}}{n}$

= 13.8




SW = $\frac{(1.2 - 1.8)^2}{2+1}$ = 0.12

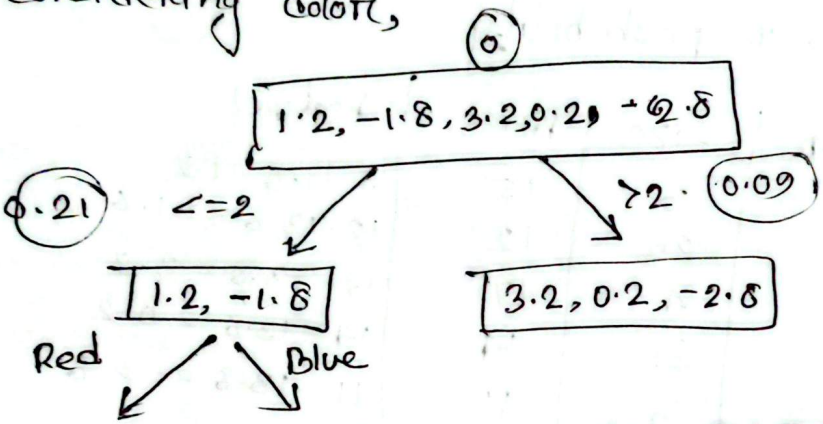
SW = $\frac{(3.2 + 0.3 - 2.8)^2}{3+1}$ = 0.09.

$$\text{Root} = \frac{(1.2-1.8+3.2+0.2-2.8)^2}{5+1} = 0$$

∴ Gain = $(0.12+0.09)-0 = 0.2$

for making the calculation easier, we are considering the above one gives the highest gain.

Now, considering color,




$$\frac{(1.2)^2}{1+1} = 0.72 \qquad \frac{(-1.8)^2}{1+1} = 1.62$$

size দেখানোর দরকার নাই?
2G feature দিয়ে দেখালেই হবে।

Considering the last cost,

= Base + $\alpha_1 \times 0.72$ → Possible case value → first row এর জন্য
↳ ধরে নিচ্ছি (0.1)

= $13.8 + 0.1 \times 0.72$

= 14.16 → 1st row এর Predicted value.

Performance Matrix :

TP : The positive class is predicted as positive.

TN : " Negative " " " " Negative.

FN : " Positive " " " " " "

fp : " Negative " " " " Positive.

Accuracy : Overall how often.

$$Acc = \frac{TP + TN}{TP + TN + fP + fN}$$

$$\text{Precision } (P) = \frac{TP}{TP + fP}$$

$$\text{Recall } (R) = \frac{TP}{TP + FN}$$

$$f1\text{-score} = \frac{2 \times P \times R}{P + R}$$

False Neg = 1 − R

" Pos = 1 − Specificity

$$= \frac{fP}{fP + TN}$$

| Actually \ Pre class | NO | Yes |
|---|---|---|
| No | 45 | 5 |
| Yes | 5 | 95 |

TN = 45

fP = 5

fN = 5

TP = 95

$$Acc = \frac{95 + 45}{95 + 45 + 5 + 5} = 0.933$$

$$R = \frac{95}{95 + 5} = 0.95$$

$$P = \frac{95}{95 + 5} = 0.95$$

$$f_1 = \frac{2 \times 0.95 \times 0.95}{0.95 + 0.95} = 0.95.$$

10,000 patient were tested to check corone effect Among them, 9000 were healthy & 1000 were actually sick. Those who are sick, when tested, 620 were found ⊕ & 380 where ⊖ similarity, when checking the healthy patient, 180 were found ⊕ corona & 8820 were ⊖ found precision, recall, F1.

| Actually | Predicted: Yes | Predicted: No |
|---|---|---|
| Yes | 620 | 380 |
| No | 180 | 8820 |

| True class | Predicted class 1 | 2 | 3 |
|---|---|---|---|
| 1 | 8 | 2 | 0 |
| 2 | 1 | 9 | 0 |
| 3 | 1 | 2 | 7 |

| TP | FN |
|---|---|
| FP | TN |

1 → 2,3

| 8 | 2 |
|---|---|
| 2 | 18 |

2 → 1,3

| 9 | 1 |
|---|---|
| 4 | 16 |

3 → 1,2

| 7 | 3 |
|---|---|
| 0 | 20 |

$$\text{Total Acc} = \frac{A_1 + A_2 + A_3}{3}$$

$$P = \frac{P_1 + P_2 + P_3}{3}$$

$$R = \frac{R_1 + R_2 + R_3}{3}$$

$$f1 = \frac{f1_1 + f1_2 + f1_3}{3}$$

Predicted class

| | A | B | AB | 0 |
|---|---|---|---|---|
| A | 350 | 11 | 30 | 9 |
| B | 15 | 359 | 21 | 5 |
| AB | 35 | 35 | 320 | 10 |
| 0 | 20 | 20 | 27 | 333 |

Find the acc & f1 score of the example.

A → B, AB, O → | 350 | 50 |
| --- | --- |
| 70 | 1130 |

$Acc = \frac{1480}{1600} = 0.925$

$P = 0.833, R = 0.875, f1 = 0.854$

B → A, AB, O → | 359 | 41 |
| --- | --- |
| 66 | 1134 |

$Acc = 0.933$

$P = 0.845, R = 0.898, f1 = 0.87$

AB → A, B, O → | 320 | 80 |
| --- | --- |
| 78 | 1122 |

$Acc = 0.901$

$P = 0.804, R = 0.8, f1 = 0.802$

O → A, B, AB → | 333 | 67 |
| --- | --- |
| 24 | 1176 |

$Acc = 0.943$

$P = 0.933, R = 0.832, f1 = 0.88.$

$$Acc = \frac{0.925 + 0.933 + 0.901 + 0.943}{4} = 0.925$$

$$f1 = \frac{0.854 + 0.87 + 0.802 + 0.88}{4} = 0.852.$$

Confusion / Error Matrix – Used to decision the performance of a classification model on test data.

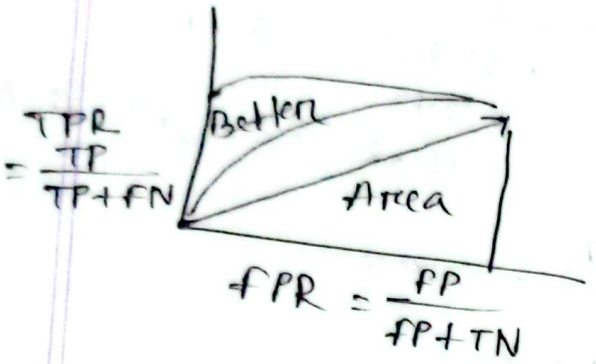
Instead of being overwhelmed with confusion, matrics, ROC graph provides a simple way to summarize all of the info.

ROC plot TPR vs FPR at diff classification threshold it shows how well ~~classification~~ classifier distinguishes ⊕ & ⊖ classes it can be 0 to 1.

Higher ROC AUC indicates better performance.

A perfect model would have AUC of 1, while a random make would have 0.5.

ROC – Receiver Operating characteristic curve.
AUC – Area under the curve.




যত কম false positive with যত বেশি true positive, classifier predict করতে পারে, তত model ভালো

Our idea point (0,1) which is rarely can be done by the classifier.

ROC — Plot the ROC curve

| Class | Prob | TP | FP | TPR | FPR |
|---|---|---|---|---|---|
| P | 0.9 | 1 | 0 | 1/5 | 0/5 |
| P | 0.8 | 2 | 0 | 2/5 | 0/5 |
| N | 0.7 | 2 | 1 | 2/5 | 1/5 |
| P | 0.6 | 3 | 1 | 3/5 | 1/5 |
| P | 0.55 | 4 | 1 | 4/5 | 1/5 |
| N | 0.54 | 4 | 2 | 4/5 | 2/5 |
| N | 0.53 | 4 | 3 | 4/5 | 3/5 |
| N | 0.51 | 4 | 4 | 4/5 | 4/5 |
| P | 0.5 | 5 | 4 | 5/5 | 4/5 |
| N | 0.4 | 5 | 5 | 5/5 | 5/5 |

| x | y | AC | Pred |
|---|---|---|---|
| | | $l_c$ 80 | 75 |
| $l_1$ | 80 | | |
| | | $l_7$ 75 | 85 |
| $l_2$ | 90 | | |
| $l_3$ | 100 | | |
| $l_4$ | 110 | | |
| $l_5$ | 120 | | |

suppose, now predicted values for $l_c$ and $l_7$ are 75 & 85. The actual value was 80 and 75. Calculate Mean Absolute error & mean square error.

(i) Mean Absolute Error $= \frac{1}{n} \sum_{i=0}^{n-1} (|y_i - \hat{y}_i|)$

$= \frac{1}{2} (|80-75| + |75-85|)$

$= \frac{15}{2} = 7.5$

(ii) Mean Square Error, $MSE = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2$

$= \frac{1}{2} ((80-75)^2 + (75-85)^2)$

$= 62.5.$

clustering:

- kmeans clustering
- DBScan.

(i) kmeans:

(i) select num of clusters you want to identify in your data. This is the k in kmeans clustering.

(ii) Randomly select k values there are initial clusters.

(iii) Measures the distances betn the first point & 3 ~~int~~ initial clusters.

(iv) Assign the first point to nearest cluster. similarly, continue for all.

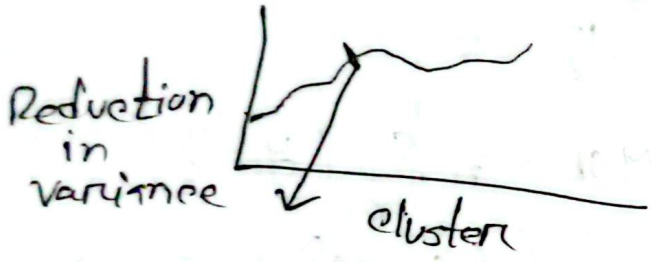
(v) Calculate the mean of 3 clusters.

(vi) Iterate (i) to (v) until clusters no longer change.

Selecting the value of k.

(i) Total variation within each clusters যখন সবচেয়ে কমে, k এর value নিব।

(ii) Reduction in variance for value k.




Hierarchical clustering:

→ No need to predefine the value of k, builds a tree like structure of clusters.

→ Similar data points are in same clusters, otherwise in different clusters.

Tree can be built in 2 ways:

(i) Agglomerative Bottom up Approach.

(ii) Divisive Top Down "

Consider the following set of 6, dimentional data points 18, 22, 25, 42, 27, 43 → sort করে 2 side এ বসাব।

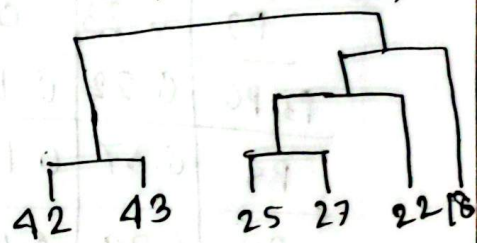
Apply the agglomerative hierarchical clustering to build the clustering dendogram (Tree like structure)

step:

→ Merge the clusters using min distance & update the proximity matrix accordingly.

→ Clearly present the proximity matrix in each iteration.

| | 18 | 22 | 25 | 27 | 42 | 43 |
|---|---|---|---|---|---|---|
| 18 | 0 | 4 | 7 | 9 | 24 | 25 |
| 22 | 4 | 0 | 3 | 5 | 20 | 21 |
| 25 | 7 | 3 | 0 | 2 | 17 | 18 |
| 27 | 9 | 5 | 2 | 0 | 15 | 16 |
| 42 | 24 | 20 | 17 | 15 | 0 | 1 |
| 43 | 25 | 21 | 18 | 16 | 1 | 0 |




| | 18 | 22 | 25,27 | 42,43 |
|---|---|---|---|---|
| 18 | 0 | 4 | 7 | 24 |
| 22 | 4 | 0 | 3 | 20 |
| 25,27 | 7 | 3 | 0 | 17 |
| 42,43 | 24 | 20 | 17 | 0 |

| | 18 | 22 | 25 | 27 | 42,43 |
|---|---|---|---|---|---|
| 18 | 0 | 4 | 7 | 9 | 24 |
| 22 | 4 | 0 | 3 | 5 | 20 |
| 25 | 7 | 3 | 0 | 2 | 17 |
| 27 | 9 | 5 | 2 | 0 | 15 |
| 42,43 | 24 | 20 | 17 | 15 | 0 |

| | 18 | 22,25,27 | 42,43 |
|---|---|---|---|
| 18 | 0 | 4 | 24 |
| 22,25,27 | 4 | 0 | 20 |
| 42,43 | 24 | 20 | 0 |

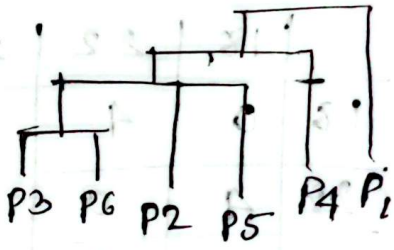
| | 18,22,25,27 | 42,43 |
|---|---|---|
| 18,22,25,27 | 0 | 24 |
| 42,43 | 24 | 0 |

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

| | x | y |
|---|---|---|
| P1 | 0.40 | 0.53 |
| P2 | 0.22 | 0.38 |
| P3 | 0.35 | 0.32 |
| P4 | 0.26 | 0.19 |
| P5 | 0.08 | 0.41 |
| P6 | 0.45 | 0.30 |

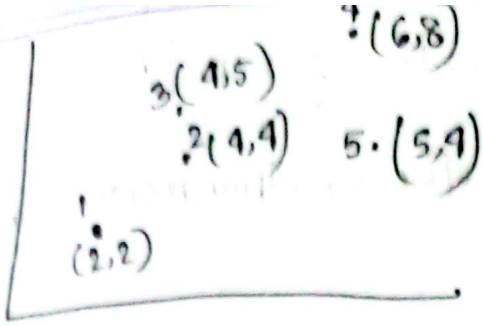
| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| P1 | 0 | 0.23 | 0.22 | 0.37 | 0.34 | 0.24 |
| P2 | 0.23 | 0 | 0.14 | 0.19 | 0.14 | 0.24 |
| P3 | 0.22 | 0.14 | 0 | 0.16 | 0.28 | 0.10 |
| P4 | 0.37 | 0.19 | 0.16 | 0 | 0.28 | 0.22 |
| P5 | 0.34 | 0.14 | 0.28 | ~~0.10~~ 0.28 | 0 | 0.39 |
| P6 | 0.24 | 0.24 | ~~0.28~~ 0.10 | 0.22 | 0.39 | 0 |

| | P1 | P2 | P3, P6 | P4 | P5 |
|---|---|---|---|---|---|
| P1 | 0 | 0.23 | 0.22 | 0.37 | 0.34 |
| P2 | 0.23 | 0 | 0.14 | 0.19 | 0.14 |
| P3, P6 | 0.22 | 0.14 | 0 | 0.16 | 0.28 |
| P4 | 0.37 | 0.19 | 0.16 | 0 | 0.28 |
| P5 | 0.34 | 0.14 | 0.28 | 0.28 | 0 |




| | P1 | P2, P3, P6, P5 | P4 |
|---|---|---|---|
| P1 | 0 | 0.23 | 0.37 |
| P2, P3, P5, P6 | 0.23 | 0 | 0.19 |
| P4 | 0.37 | 0.19 | 0 |

| | P1 | P2, P3, P4, P6, P5 |
|---|---|---|
| P1 | 0 | 0.23 |
| P2, P3, P4, P5, P6 | 0.23 | 0 |




* Ensemble (10/15)
  - XGBoost
  - Adaboost

Feature selection (5)
(PCA, UMAP, t-SNE)

Performance Matrix (5)

Clustering (5) ~~be~~
- Kmeans
- Hierarchical

Deep ~~learn~~ & learning (5)

DT *

Reinforcement learning:
Optimal behaviour is learnt through interaction & observations of how it responds.

(1) Agent or learner
↑
Environment the agent interacts with

(i) observation
(ii) Reward.
(iii) Action.

* How exactly it chooses action in general?

(i) Exploration: Act exploring the environment to find out info about it.

(ii) Exploitation: Act of exploiting info in order to maximize return.

Balance: Epsilon Greedy strategy. Epsilon=1
↓
Exploration
Decay & exploit

Reinforcement learning:

→ System, at first explores then, $\varepsilon = 1$.

→ Gradually, likelihood of the system decreases & it knows enough about the environment.

→ It takes to exploit the possible cases which one gives the highest accuracy.

→ Based on its prediction, a penalty or reward is given & it tries to further improve itself

→ Each time, it generates, a random number. If it is greater than the current $\varepsilon$ value, agent will choose exploitation.

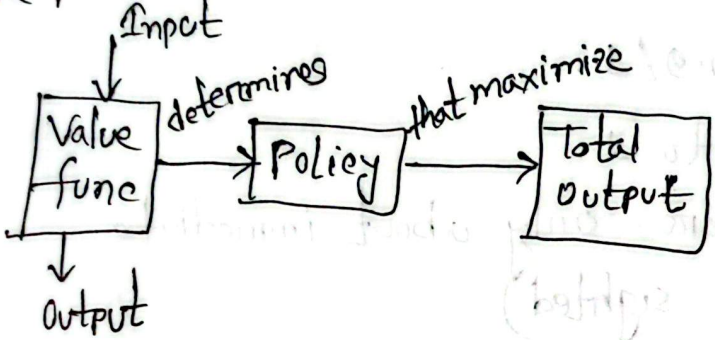
→ Update the state using Bellman eqn so that Q will eventually optimize.

Reinforcement learning are of 2 Types

- Value Based → [Value function] → [Optimal policy] — how a situation behave in a situation + previous effect. (Robot)
- Policy Based → [Optimal Policy] — how a situation behaves in a given situation.

Deep Q Learning:

→ Value based reinforcement learning method to solve the problem.




It handles both exploration & exploitation.

↳ Exploration → Variety situation knowledge.

Exploitation → Better Accuracy.

Both are needed, otherwise better option might miss.

It tries to learn something new even if it seems bad before discovering the better path.

$$q(s,a) = Q(s,a) + \alpha \left[ r + \gamma \max Q(s,a) \right]$$

- $Q(s,a)$ → Previous state of that situation
- $\alpha$ → learning rate
- $r$ → Immediate state
- $\gamma$ → Discount factor
- $\max Q(s,a)$ → Best previous value for future action
- action.

Learning rate

→ " Generally 0 to 1

→ Small $\alpha$ → Update are tiny, learning is slow but stable

→ large $\alpha$ → Update are big, fast-faster training but can be noisy.

→ Practically, we start with higher $\alpha$ & reduce it over time.

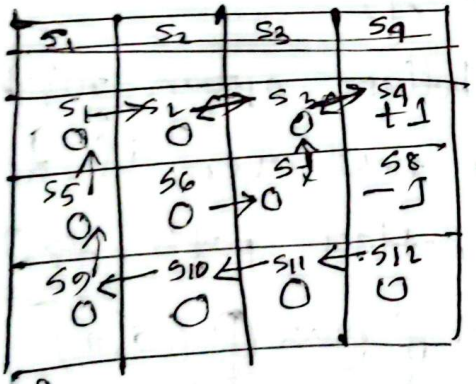
Discount factor: 0.9/1.

→ $\gamma$ generally 0 to 1.

→ $\gamma = 0$, agent cares only about immediate rewards (start sighted)

→ $\gamma = 1$, agent cares long-term rewards (plans ahead)

~~Usally~~ Usually starts with higher value.




(4,4)

$\gamma = 0$.

Robot/agent (0,0)

Backtracking

$V(S_3) = 0 + 1\,[1 + 0.9 \times 0] = 1.$

$V(S_2) = 0 + 1\,[0 + 0.9 \times 1] = 1 \times 0.9 = 0.9$

$V(S_1) = 0 + 1\,[0 + 0.9 \times 0.9] = 0.81$

$V(S_5) = 0 + 1\,[0 + 0.9 \times 0.81] = 0.73$

$V(S_9) = 0 + 1\,[0 + 0.9 \times 0.73] = 0.66$

$V(S_{10}) = 0 + 0.9 \times 0.66 = 0.59$

$V(S_{11}) = 0 + 0.9 \times 0.59 = 0.53.$

$V(S_{12}) = 0 + 0.9 \times 0.53 = 0.43$

$V(S_7) = 0 + 0.9 \times 1 = 0.9 \rightarrow$ New $(Q(s,a))$

$V(S_6) = 0 + 0.9 \times 0.9 = 0.81.$

$Q = 0$

→ Same path (Scenario)

→ Iteration fixed.

A robot in 5×5 grid must learn to reach a goal (5,5) while avoiding obstacles. Explain how Q learning updates it Q table over time & what role do $\alpha$ & $\gamma$ play in this.

Explainable AI (XAI) → Scenario based → Transparency
→ How solves Bias & fairness
→ Why XAI needed
→ Adv & disadv
→ local — Lime, shap
→ global — shap, eando

→ Refers to AI method's decision making process.

* ~~Process~~ SHAP (Shapely Additive Explanation).

→ Based on game theory.

→ Shows how much each feature adds & students. from the model's final prediction.

SHAP has added advantage that it can compute global feature importance by averaging.

SHAP values accuracy accross many simples.

* LIME (Local Interpretable Model Agnostic Explanation).

→ Builds a ~~smi~~ simple, local model around each individual prediction.

(i) Sampling & obtaining a surrogate dataset.
(ii) feature selection
(iii) Ridge Regression model
} Presented the output through a sample.

XAI help to

(i) Build Trust – Users see the reasoning behind. prediction.
(ii) Ensure fairness – Detect & Locate correct bias, revels the abnormal behaviour.
(iii) Improve Accountability :– Regulators & humans verify AI ~~behaviour~~ behavious.

(IV) Support debugging: fix wrong Logic.

Transformers:

(I) Attention: How model knows which were to attend to is all learned while training with back propagation. Attention mechanism has an infinite references ~~window~~ window.

(1) encoder: Maps an input sequence into an abstract continuous representation that holds learned info of that input.

(2) Decoder: Takes that continuous representation & step by step generate a single output, while are also being fed to previous input.

How Encoder (BERT, RoBERT, ALBERT) works?

(I) Input embedding - Vector representation of given information.

(II) Inject positional - Information through encoding (using sine & cosine func).

(III) Encoder layer (Multiple layer) - To map all input sequence into an abstract continuous representation that holds the learned info for that entire sequence. It contains 2 submodule.

(i) output goes to a linear layer, each output are ~~connect~~ concatenated to a single vector.

(I) Multihead Attention → We fed 3 distinct fully connected layers to create query key & value vector.
At first, query x key = score.
The higher score has higher priority.
then, $\text{softmax}\left(\frac{\text{score}}{\sqrt{\text{dimension}}}\right) \times \text{value} = \text{output}$.

(IV) Residual connection, layer Normalization & Pointwise FFNN:
Multithread all output + original input goes through these layers & final output are again fed to input. Continuous further for reacher representation.

→ How decoder (GPT works)?
(I) Input embedding.
(II) Inject positional Info
(III) Decoder layer.
The main difference lies in multihead attention.
Here, it can access the current position with all previous info. we cover the future words & gives it to predict.
This is called Masking.

Score + Look Ahead Mask = Masked score.

[1 for previous works, α for future works]

Here, when goes to sigmoid func, the future values become 0, then it continues & predict 1. output.

(iv) In second layer, final output & all input are passed and continued.

Math (25/30)

Ensemble - Compare
feature selection
XAI, TranX, VAE, GAN
NN, Gradient Descent.

(10/15)

Generative Adversarial Network (GAN)

(i) Generator → Takes random noise as input & tries to create fake data.

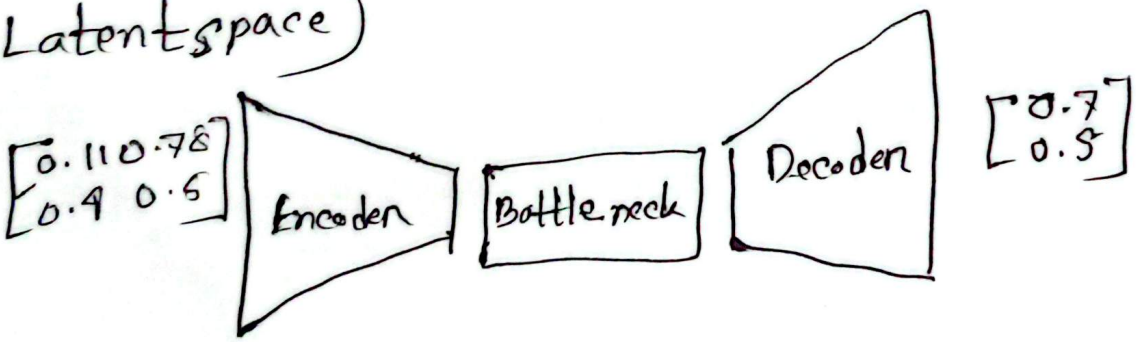
(ii) Discriminator → Receives real data, fake data from the generator & learns to distinguish betⁿ real and fake data.

(iii) Training is like competition. Generator gets better at producing realistic data. The discriminator can no longer tell real from fake. Thus GAN converges.

Variational Auto Encoder (VAE)

→ Unlike normal encoder, that outputs a fixed latent vector, a variational encoder outputs a distribution using mean ($\mu$) & variance ($\sigma^2$).

Basic concept of autoencoder is based on 3 component — Encoder, decoder, Bottleneck. (Latentspace)



(1) Encoder compresses input data into latent space representation. Latent space is low dimensional space that captures the essential features of the input data.

(2) Bottleneck layer holds the compressed representation

(3) Decoder reconstructs the input data from compressed representation produced by encoder & bottleneck.

Minimize the diff bet$^n$ original & reconstructed data. Improves decoders ability to accurately reconstruct the original data from its compressed representation. At the same time, Encoder becomes good at compressing data in a way that preserves critical info.

The diff of 2 images are measured in pixel value. Loss func $\rightarrow \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2$.

Try to reduce mean square error for each image it processes. At first, Reconstruction weight $\rightarrow$ High, Auto Encoder weight $\rightarrow$ Random. As training process, error decrease. Auto Encoder gets better at recreating.

* VAE learns a probabilistic latent space & reconstructs data using reconstruction loss, while GAN learns to generate data through Generator Discriminator adversarial networks.
* VAE — Output smooth but slightly blurry samples, GAN — More sharper, realistic, image.
* VAE — Anomaly detection, feature reduction
GAN — Produce New data.
* GAN - is harder to train.

Hyper parameter tuning:
Grid Search: Tries all combination of hyper parameters. ~~Bayesian~~
Bayesian Optimization: Instead of trying all combinations, it predicts which parameters are promising.

Best for example problems, when search space is ~~fear~~ large.

Transfer learning

↳ federated learning
- ✓ কীভাবে কাজ
- ✓ কখন কোনটা use.

→ Model Train
→ Data secure

* কোন Activation func কখন use } NN
* Gradient Descent

- ~~Ese~~ Ensemble
- feature selection
- XAI, GAN, VAE,
- Gradient descent/ Perceptron.