These are the core ML steps:

1. Data ingested
2. EDA
3. Processing or FE
4. Model building
5. evalution

Core step

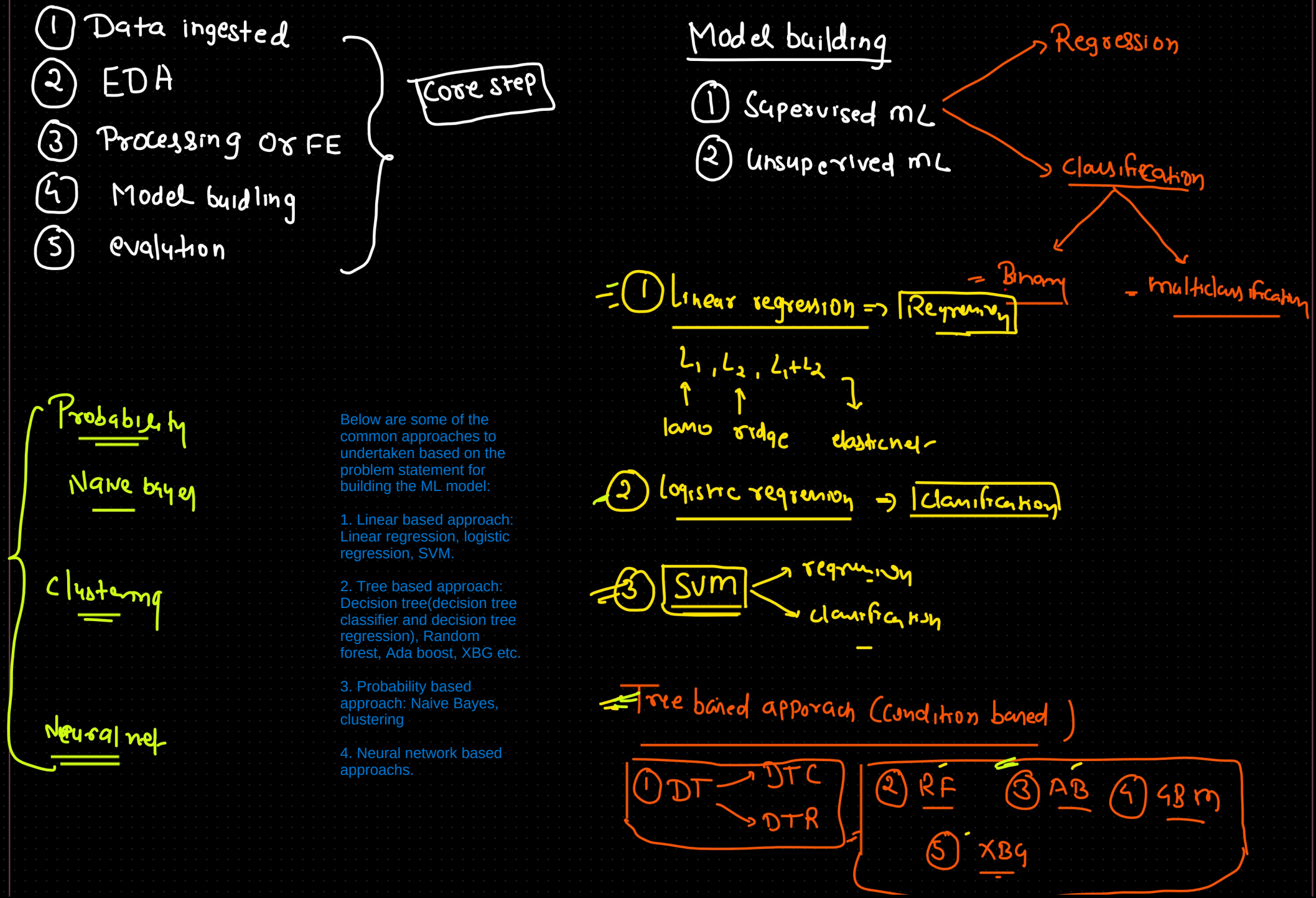
Below are some of the common approaches to undertaken based on the problem statement for building the ML model:

1. Linear based approach: Linear regression, logistic regression, SVM.

2. Tree based approach: Decision tree(decision tree classifier and decision tree regression), Random forest, Ada boost, XBG etc.

3. Probability based approach: Naive Bayes, clustering

4. Neural network based approachs.



Decisontree

1. DTC
2. DTR

Decision tree can be used for both classification(Decission tree classifier) and regression problems(Decision tree regressor).

Different approach/algorithm of DT: ID3, CART,C4.5(C4.5 is extended version of ID3)

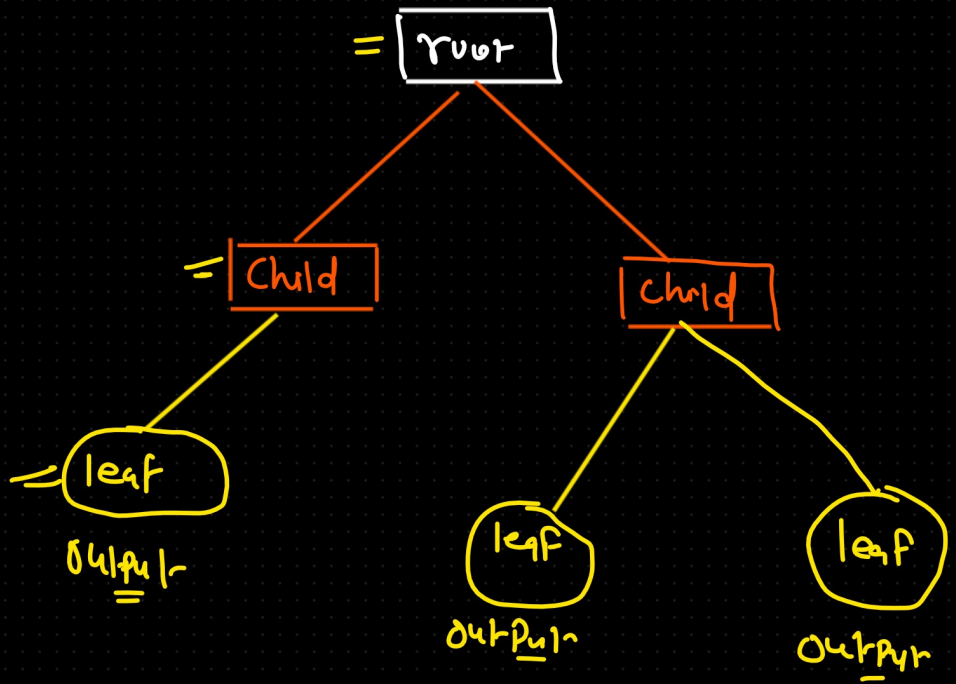
1. DTC
2. Entropy
3. Gini impurity | Gini coff | Gini index
4. IG (Information gain)
5. Pruning → Pre / Post-
6. DTR

ID3, CART, C4.5

Decision tree is condition based approach. Let's visualize this below:

age = 15 | age = 17 | age = 23

```
if (age <= 15):
    Print ("school")
elif (age > 15 and age <= 21):
    Print ("College")
else:
    Print ("working")
```



--> In above age can be considered as ROOT node.
--> Condition applied on age using if, elif and else as CHILD node.
--> Whereas, condition based output as a LEAF node.

Decision-tree
1. ID3 = Iterative Dichotomiser 3 ⇒ C4.5
2. CART = Classification and regression tree

ID3

1. Uses, Entropy ⇒ Information gain (IG)
2. Used for Categorical (classification)
3. it may divide the root feature into more than 2 category

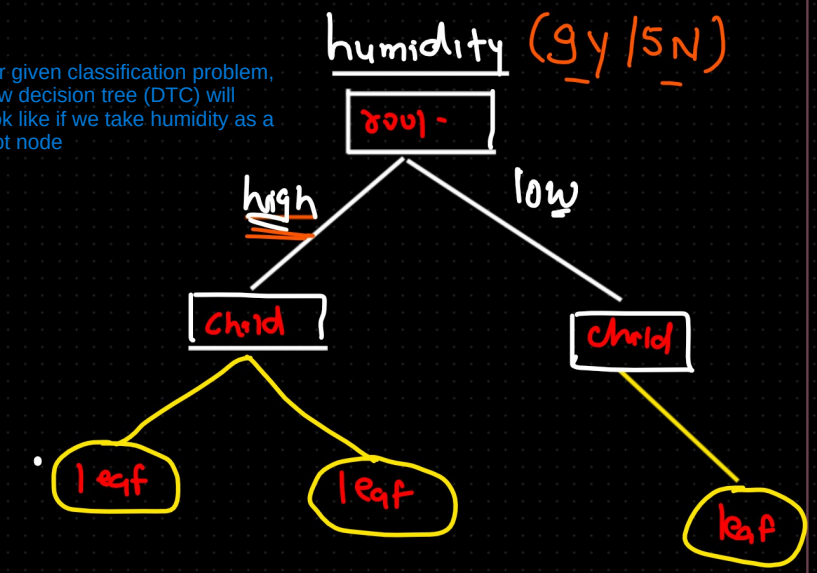
CART

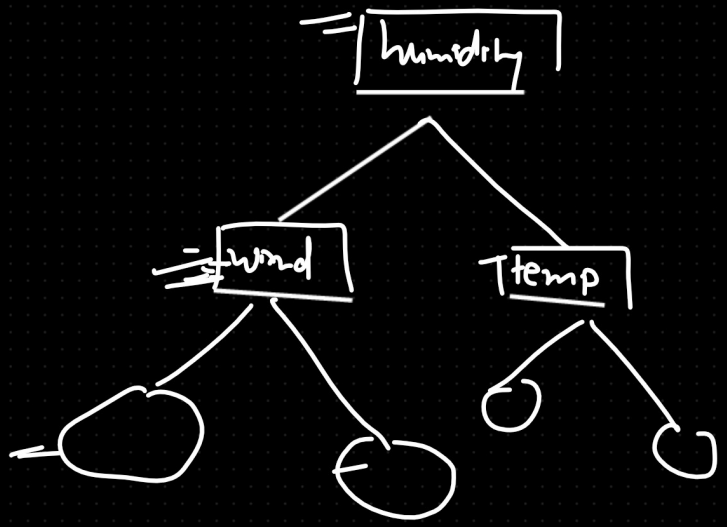
1. Uses, Gini impurity
2. Used for classification and regression
3. It always divide root-feature into two category.

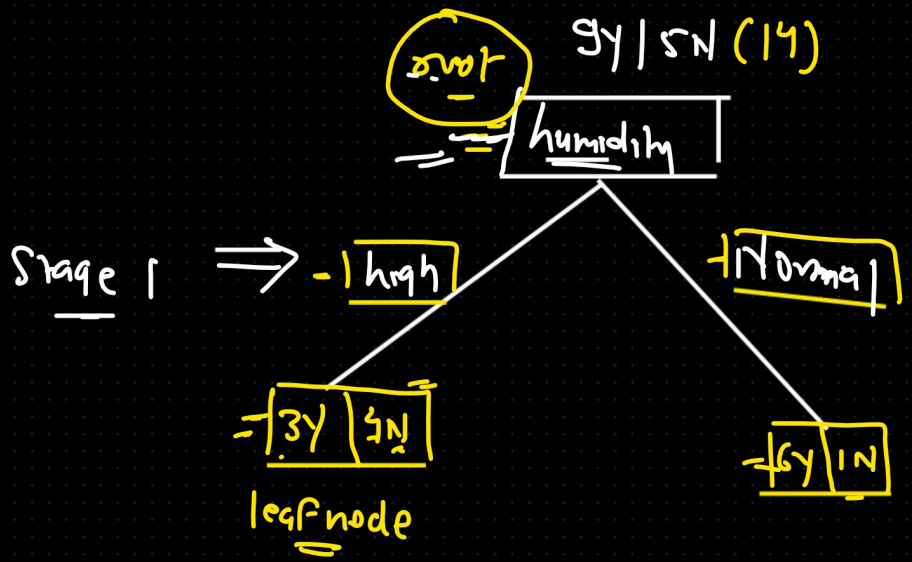
Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Data set taken is classification problem(binary classification)

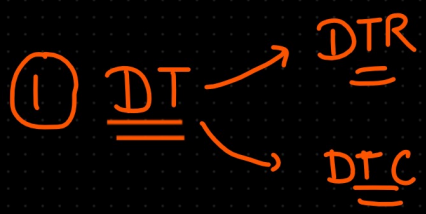
For given classification problem, how decision tree (DTC) will look like if we take humidity as a root node

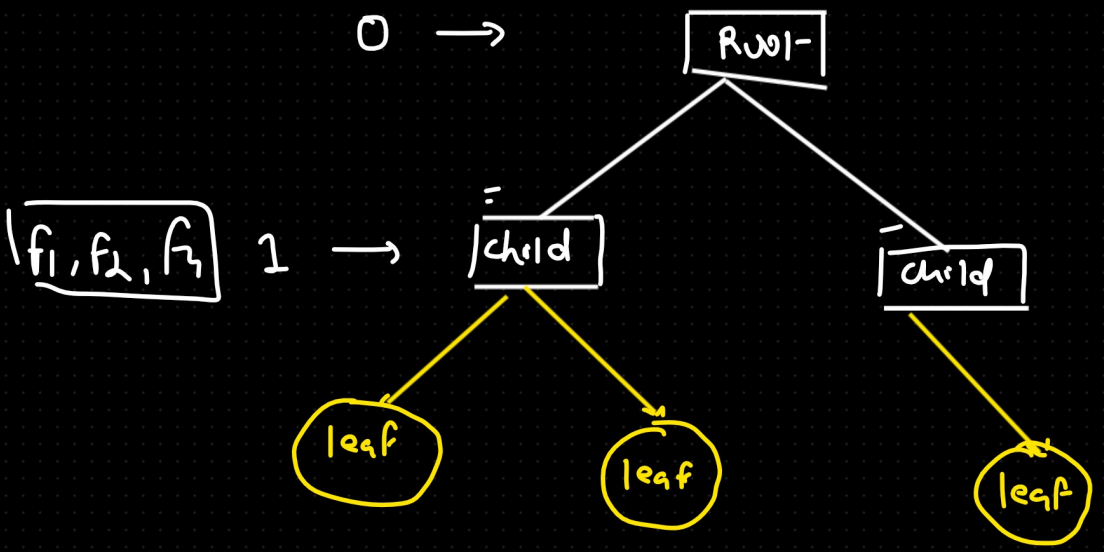






We need to decide the hierarchy that whether humidity will be a root feature or wind will be root feature what will be child nodes at different level 1, 2..so on). We will decide this hierarchy using different DT approaches: ID3, CART, C4.5





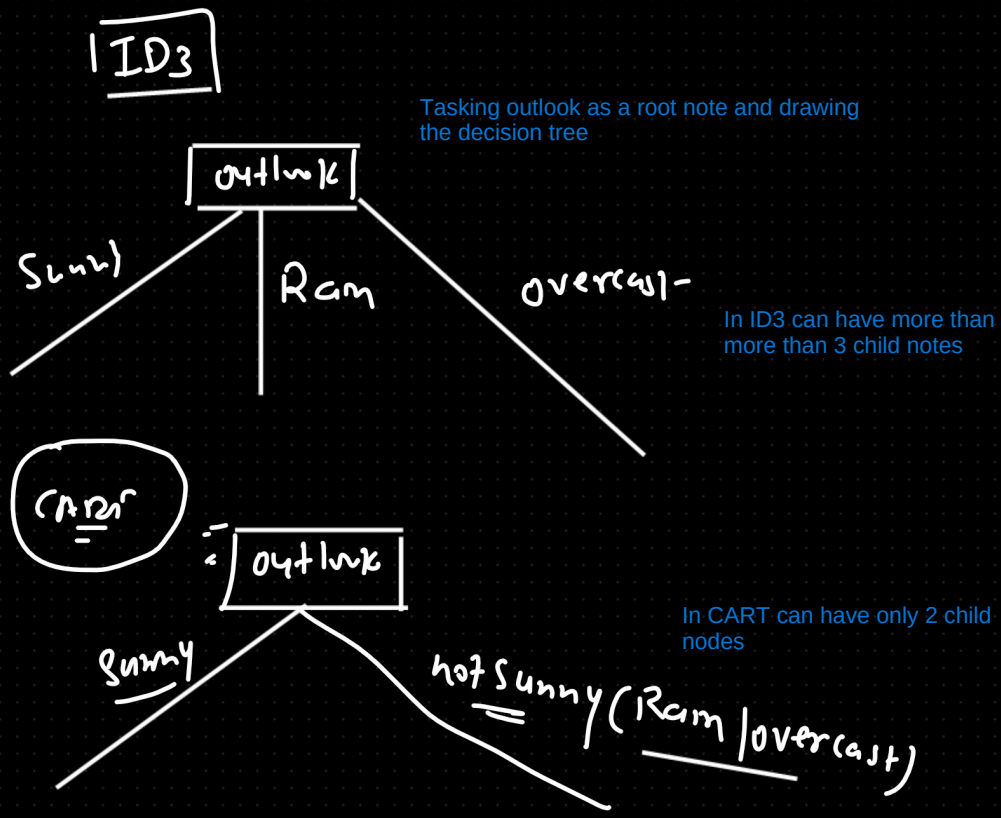
Data

f1 f2 f3 f4 O/P

=ID3 ⇒ Entropy → Information gain

=CART ⇒ Gini impurity

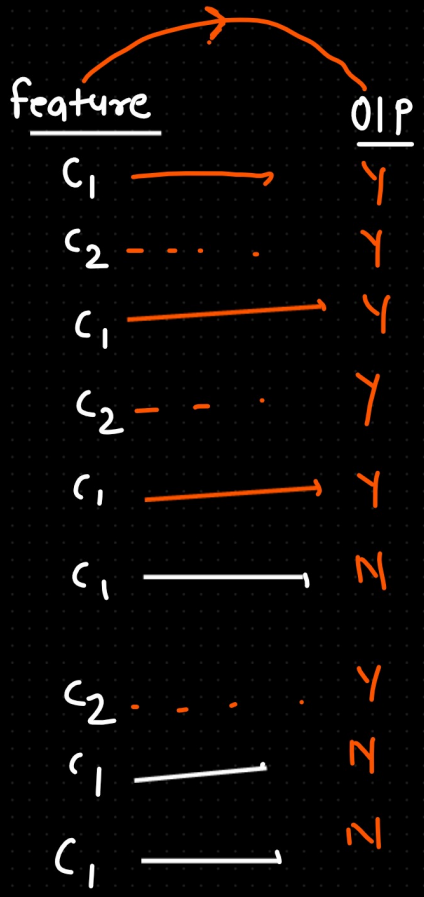
Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No



Tasking outlook as a root note and drawing the decision tree

In ID3 can have more than more than 3 child notes

In CART can have only 2 child nodes



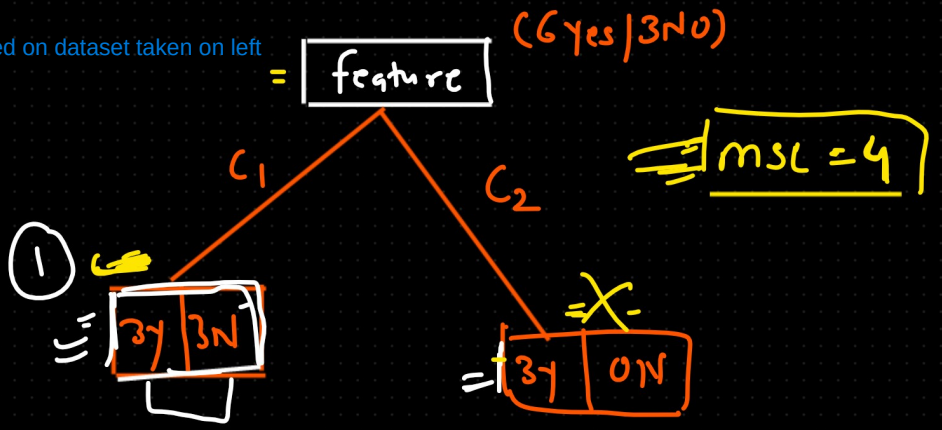
--> Entropy denoted as H(S).

--> Calculation of Entropy is applicable in case of ID3 which is only used for classification problem statement.

--> Since, in our current problem in target there are 2 classes that are getting classified (Y and N) so, entropy is calculated based on probability of Y and Probability of N.

--> That is in the entropy equation P represents probability of occurrence of each output class, whereas, i=1toN represents each of these output classes.

This DT is built based on dataset taken on left



ID_3 = entropy

CART = Gini coff

$$\text{ENTROPY} = -\sum_{i=1}^{N} P_i \times \log_2(P_i) \quad (2 \text{ class}) \quad i = 2$$

$$H(S) = -P_Y \times \log_2(P_Y) - P_N \times \log_2(P_N)$$

$\log_e^{!} e = 1$

First Entrophy is calculated wrt to left split.

$$H(S)(3Y|3N) = -\frac{3}{6} \times \log_2\left(\frac{3}{6}\right) - \frac{3}{6} \log_2\left(\frac{3}{6}\right)$$

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

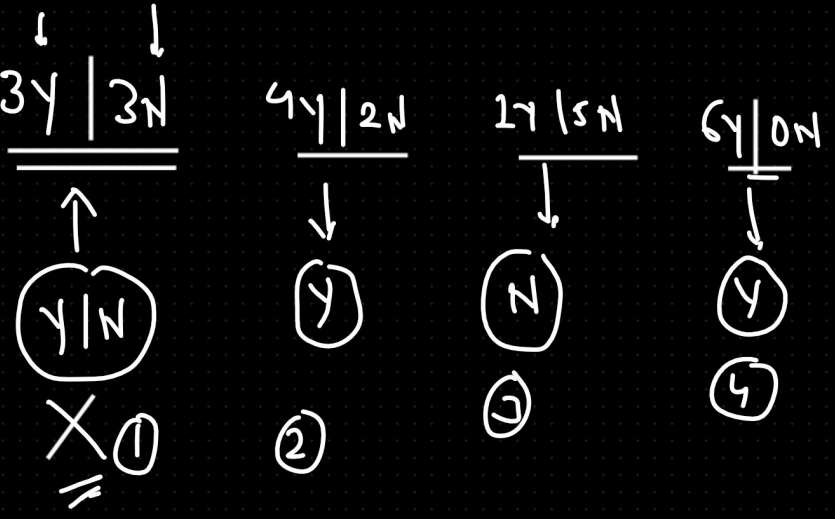
$$= \frac{-1}{2}\left[\log_2(1) - \log_2(2)\right] - \frac{1}{2}\left[\log_2(1) - \log_2(2)\right]$$

$$= \frac{-1}{2}[0-1] - \frac{1}{2}[0-1]$$

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

$$H(S) = \frac{1}{2} + \frac{1}{2} = 1$$

(3Y|3N)



Second Entropy calculated wrt to right split.

$$H(S)(3Y|0N) = -\frac{3}{3}\log_2\left(\frac{3}{3}\right) - \frac{0}{3}\log_2\left(\frac{0}{3}\right)$$

$$H(S) = -\frac{3}{3}\log_2(1) = 0$$

$$H(S) = (3Y|0N)$$

Entropy of root node=H(S)(3Y/3N)+H(S)(3Y/0N)
=1

feature with less impurity is considered to be the root feature?

Ans: Yes.

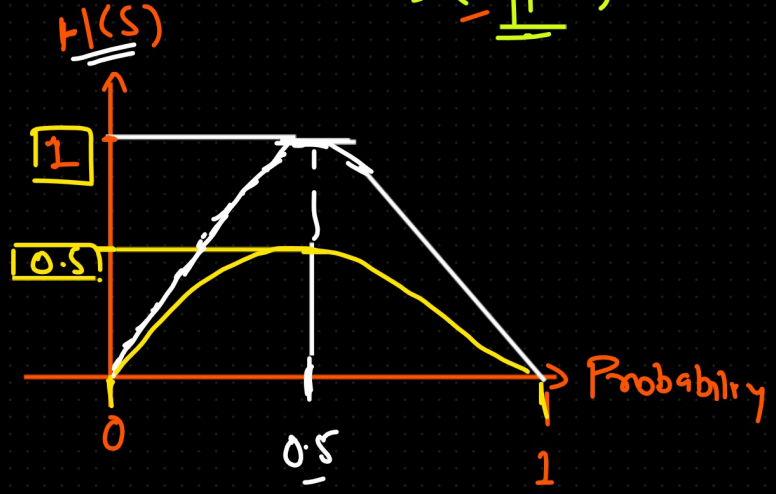
--> Entropy is the measure of impurity or randomness or uncertainty.

--> Max value of entropy is 1 and Min value of entropy is 0.

-->Entropy for (3Y/0N) is 0 because there is no randomness since this split has only one type of value that is Ys. So, output becomes quite predictable that is Y (since it is highest and there is 0 Nos) hence, it is not uncertain.

--> Entropy for (3N/3Y) is 1 because this is ideal case of randomness where we can not literally predict whether in this split output will be Y or N (since both are in equal count that is 3). uncertainty will be highest in this case.

--> H(S)(3Y/3N) > H(S)(3Y/1N) > H(S)(3Y/0N)



Important graph.
Lets understand this with the help of Y or N split. let's assume Pr represents probability of getting Y or N in a particular split. At start when Pr is 0 meaning no Pr of determining Y or N which will fall under least random category hence H(S) is also 0. Then H(S)will increase with increasing Pr until Pr =0.5. At 0.5 there will be equal probabilities of getting Y or N hence, randomness/uncertainty of selecting either Y or N will be also high that is why at this point H(S) is 1. After this H (S) decreases as we move towards Pr=1. At Pr =1 means pt where we are able to accurately predict b/w Y or N at this point randomness will be low since we are 100% able to predict either Y or N hence
H(S) = 0.

Assignment

$$\left(\frac{2!}{2!}\,|\,\frac{3!}{3!}\right) \quad H(S) = ?$$

$$-\sum_{i=1}^{N} P_i \times \log_2(P_i)$$

$2+3=5$ $\quad$ $0.97 \rightarrow 1$

$$\Rightarrow -P_Y \log(P_Y) - P_N \log(P_N)$$

$$\Rightarrow -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log\left(\frac{3}{5}\right)$$

$$\Rightarrow 0.97$$

Gini Coff or Gini Impurity $\Rightarrow$

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

Gini impurity concept will be used in the CART based approach for Decision tree.

Symbolic representation is same as what we discussed in ID3

For classification problem where o/p is classified into 2 classes Y and N, Gini impurity will be given as following:

$$G.I. \Rightarrow 1 - \left[P_Y^2 + P_N^2\right]$$

① 3Y|3N ⇒

② 3Y|0N ⇒

Gini impurity for split (3Y/3N)

① 3Y|3N ⇒ $\quad 1 - \left[\left(\frac{3}{6}\right)^2 + \left(\frac{3}{6}\right)^2\right]$

Gini impurity for split (3Y/0N)

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

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

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

$$\Rightarrow 1 - \left[\frac{2}{4}\right]$$

$$\Rightarrow 1 - \frac{1}{2} \Rightarrow \boxed{0.5}$$

5 Dt | 3Y | 3N

impure → 0.5

entropy → 1

3Y | 0N ⇒ (Y) ⇒ Impurity = 0

Gini = 0

entropy = 0

$$1 - [(1)^2 + 0]$$

$$1 - 1 = 0$$

--> Highest value of Gini impurity is 0.5 representing highly impure/randomness/uncertain.

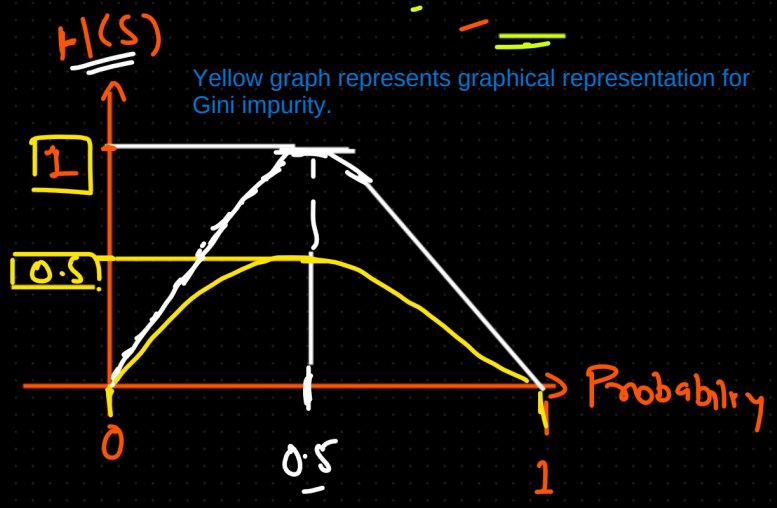
--> Lowest value of Gini impurity is 0 representing lowest impurity/randomness/uncertain.

Please note that both Entropy and Gini coefficient/index/impurity is used as a measure of randomness/impurity/uncertainty.

For classification problem both Entropy (ID3) and Gini index (CART) can be used for building a decision tree. So which one is preferred for classification problems?

Ans: For small dataset there will be negligible difference in terms of complexity(time and space) and hence, either of them can be used. But, in case of large dataset Entropy based approach ID3 will be slower as compared to Gini index based approach CART. This is because for calculation of Entropy Logarithm of base 2 is needed that increases complexity(time and space) as compared to Gini index which does not use that Logarithm.

Therefore, Gini index based approach CART should be preferred over Entropy based approach ID3 in case of large datasets for classification based problems.



Entropy = slower

gini-index = 0.5

faster

Which one should use

ID3 = Entropy

CART = Gini-index

large Dataset → ID3 ⇒ SLOW

CART

Gini coff

Internal Assignment

question ① 4Y | 8N ⇒ 0.44

② 8Y | 2N ⇒ 0.32

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

Information gain

$$G.I. \Rightarrow 1 - \left[P_Y^2 + P_N^2 \right]$$

IG is the difference between base entropy(root node) and new entropy(child node)

$$Gain = H(S) - \sum_{v \in Val} \frac{|S_v|}{|S|} H(S_v)$$

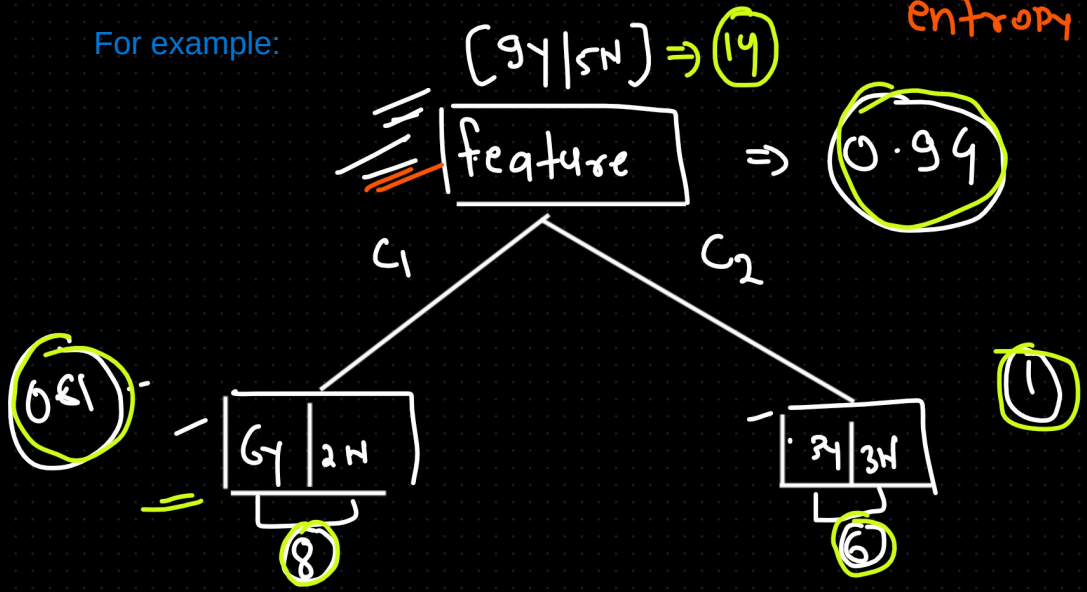
root Node entropy

Total number of data points before split.

child nod. entropy or entropy after split

| Sv | represents total number of data points after split.

For example:



$H(S)$ = root feature entropy $= -P_Y \log_2(P_Y) - P_N \log_2(P_N)$

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

$= -(0.64)\log_2(0.64) - 0.35 \log_2(0.35)$

$H(S)(\text{root}) \approx 0.9402$

$6Y|2N = -\frac{6}{8}\log_2(6/8) - \frac{2}{8}\log_2(2/8) = 0.81$

$3Y|3N = -\frac{3}{6}\log_2(3/6) - \frac{3}{6}\log_2(3/6) = 1$

$$\text{gain} \Rightarrow H(S) - \sum_{v \in \text{val}} \frac{|S_v|}{|S|} H(S_v)$$

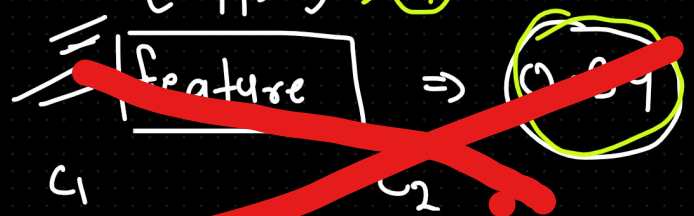
Checkout below pdf to visualize information gain.

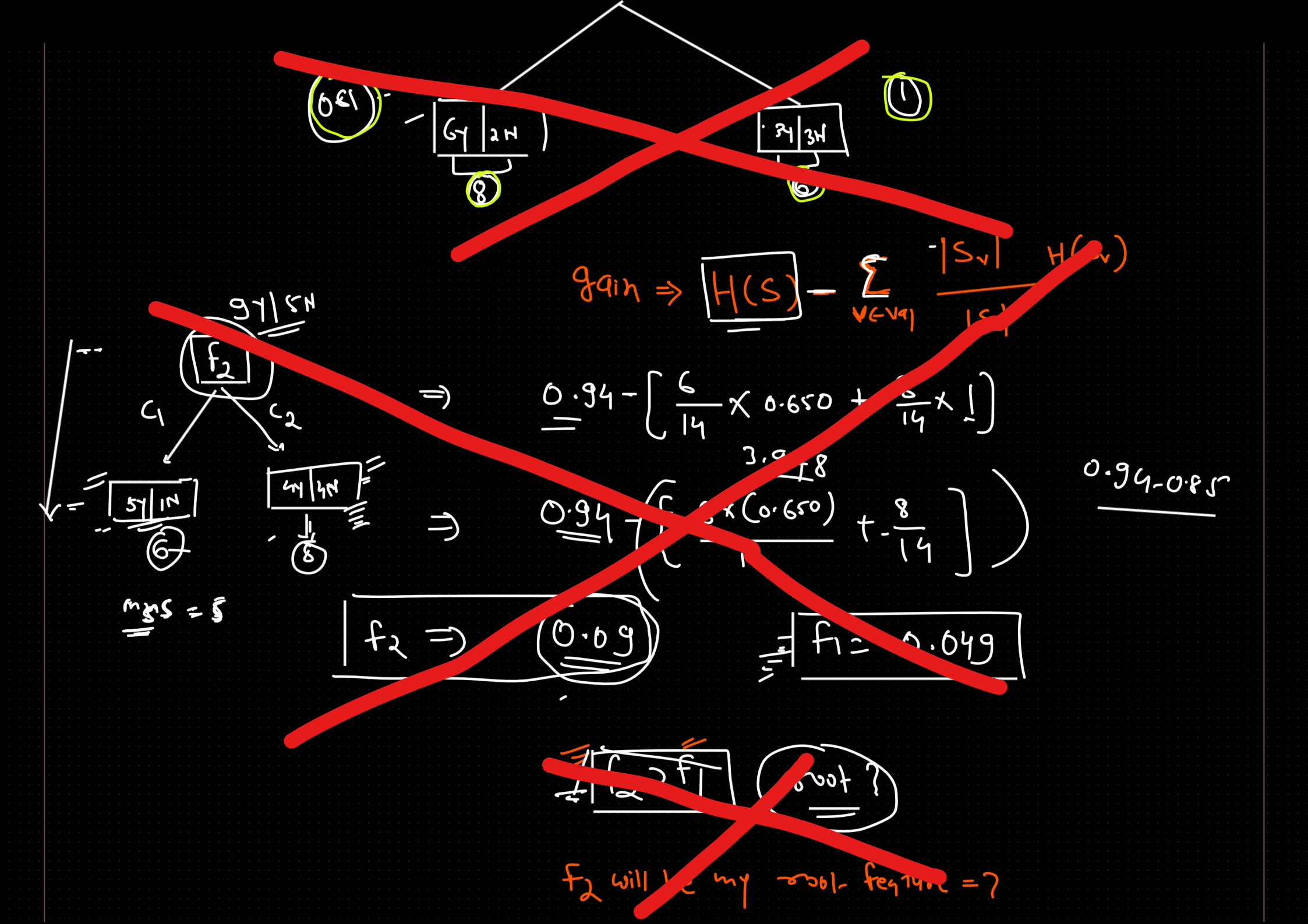
Lecture-4 (62)/IG_and_DT_implementation/infogain-example%20

$$\Rightarrow 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$$

$$\Rightarrow 0.94 - [0.462 + 0.42] = 0.0485 \approx 0.049$$

$[9Y|5N] \Rightarrow 14$




9+|5N
f2
C1
C2
5+|1N
4+|4N
mgs = 5
gain ⇒ H(S) − Σ |Sv| H(v)
0.94 − [6/14 × 0.650 + 8/14 × 1]
0.94 − 0.892
0.94 − (3.9/8 × (0.650) + 8/14)
f2 ⇒ 0.09
f1 = 0.049
f2 will be my root feature =?

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

(1)

task

IG

- Outlook = 0.01
- temp = 0.5
- Humidity = 0.67
- wind = 0.012

Here, we have calculated the Information gain for each of the features. Now, one where we are getting the highest information gain (i.e; Humidity) will be selected as a root node. This will be done at each levels. This is how hierarchy is decided for using ID3 approach of DT.

Now, we will make each feature root note and calculate IG and that hierarchy will be used where IG is more than others.

And this will be done on each level

(1) Build the complete DT on top of this Data. Important do this for 1 example by yourself

(2) C4.5 algo it's update version of ID3

Also, explore C4.5

(1) → Humadity

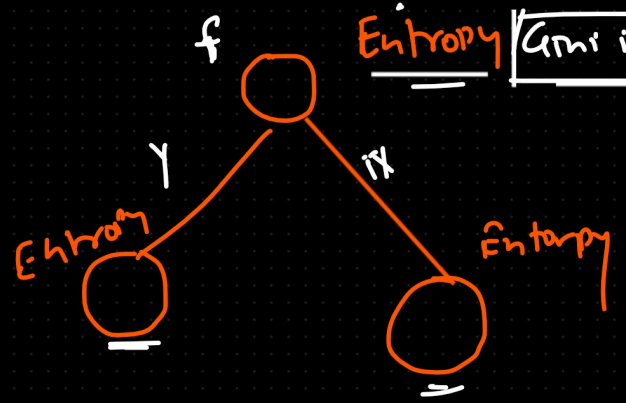
1st

- Outlook
- temp
- wind

IG

In above case we saw how to calculate information gain in case of ID3 approach using entropy.

Below, we will see how to calculate the information gain in case of CART approach using GINI impurity/coefficient.



Entropy | Gini impurity

ID3

Gain = Root entropy − [Child entropy]

The way we calculated info gain in ID3 approach (entropy) cannot be used in CART based approach (gini impurity or gini coefficient)

~~Gain = Root-Gini coff − [Gini coff child]~~

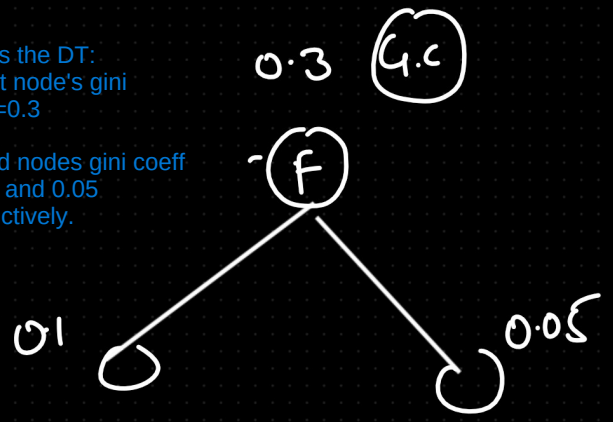
CART | Calculation of all Gini coff

Summation

Instead, in CART based approach info gain is calculated using by simply taking summation of root node and child node(s)

This is the DT:
- Root node's gini coeff=0.3

- Child nodes gini coeff is 0.1 and 0.05 respectively.



⇒ 0.3 + 0.1 + 0.05 = 0.45

0.4 + 0.05

Information gain for CART based DT taken on left

G.C

$f_1 = 0.45$

$f_2 = 0.35$

$f_3 = 0.0$ ⇐

less impurity

Note calculation for GINI Coeff is already discussed above pages.

To summarize:
--> We will ultimately select that node as root node that is providing the most Information gain.

--> Gini impurity/Coeff and entropy are methods of calculating impurity which is used for calculating the information gain.

--> This is how CART based approach (gini Coeff) and ID3 based approach (entropy) is used for determining the hierarchy of our decision tree.

*******DECISION TREE CLASIFICATION COMPLETED********

1. DTC ⇒ implementation
2. DTR –
3. PrePruning | Post-Pruning

1. DTC

1. ID3 ⇒ Entropy | IG
2. CART ⇒ Gini coeff

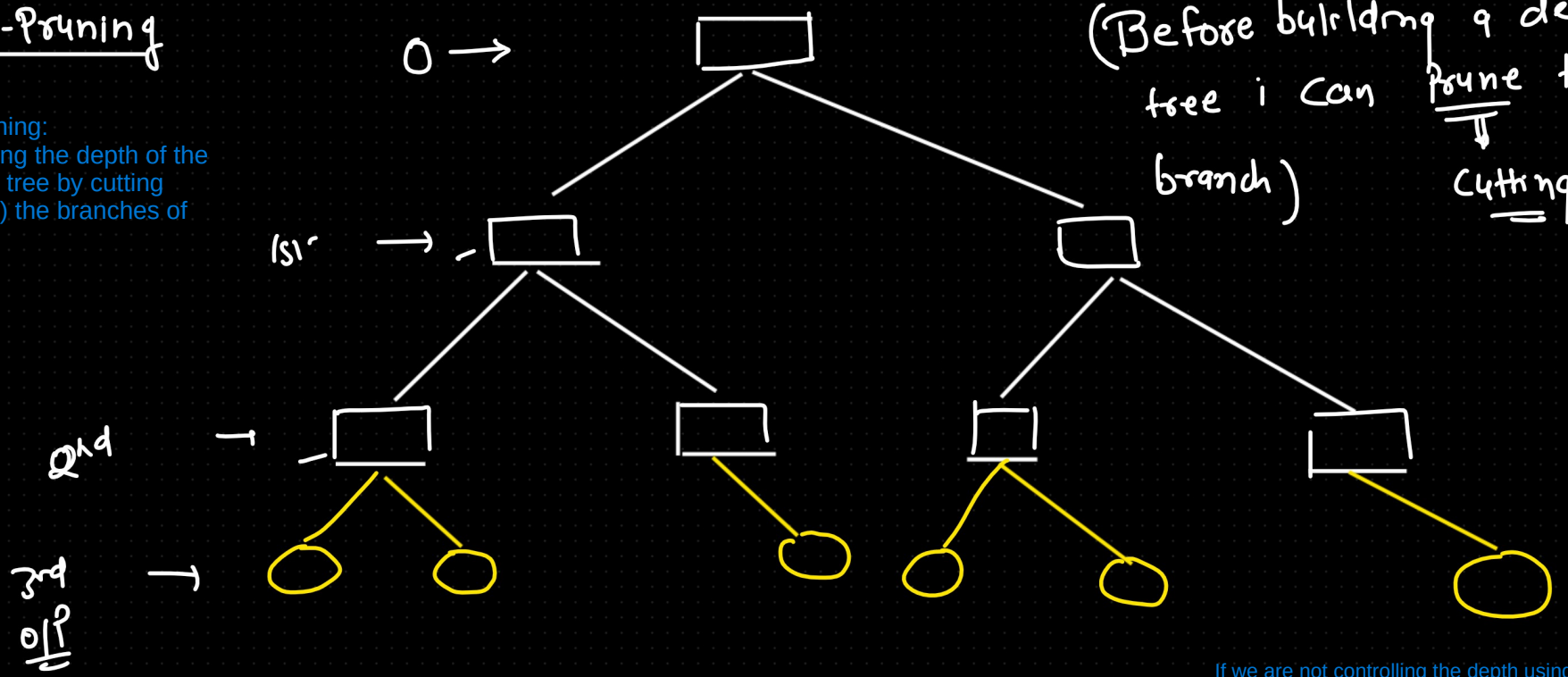
Entropy ⇒ formula
Gini formula
I.G. formula

Assignment. ⇒ Build a complete DT on the given Data

Explore C4.5 algo

Pre-Pruning

Pre-Pruning:
Controlling the depth of the decision tree by cutting (pruning) the branches of the tree.



(Before building a decision tree i can prune the branch) ↓ Cutting

If we are not controlling the depth using pre pruning then overfitting, computational expensive. That's why pre pruning with certain hyperparameter is used.

satisfying all the conditional based features

if we going to build DT till complete Depth = ?

1. Overfitting (train accuracy will be good test accuracy will be bad)
2. Computational expensive

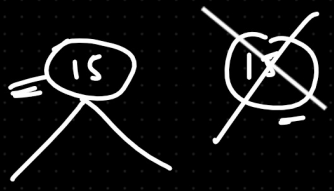
Preprune ⇒

1. Max_Depth
2. Minimum Sample leaf
3. minimum Sample split
4. max_feature

In order to counter the above stated issue we will need to be doing pruning using certain pre pruning hyperparameter tuning (max depth, min sample leaf, min sample split, max feature).

MSS = 10

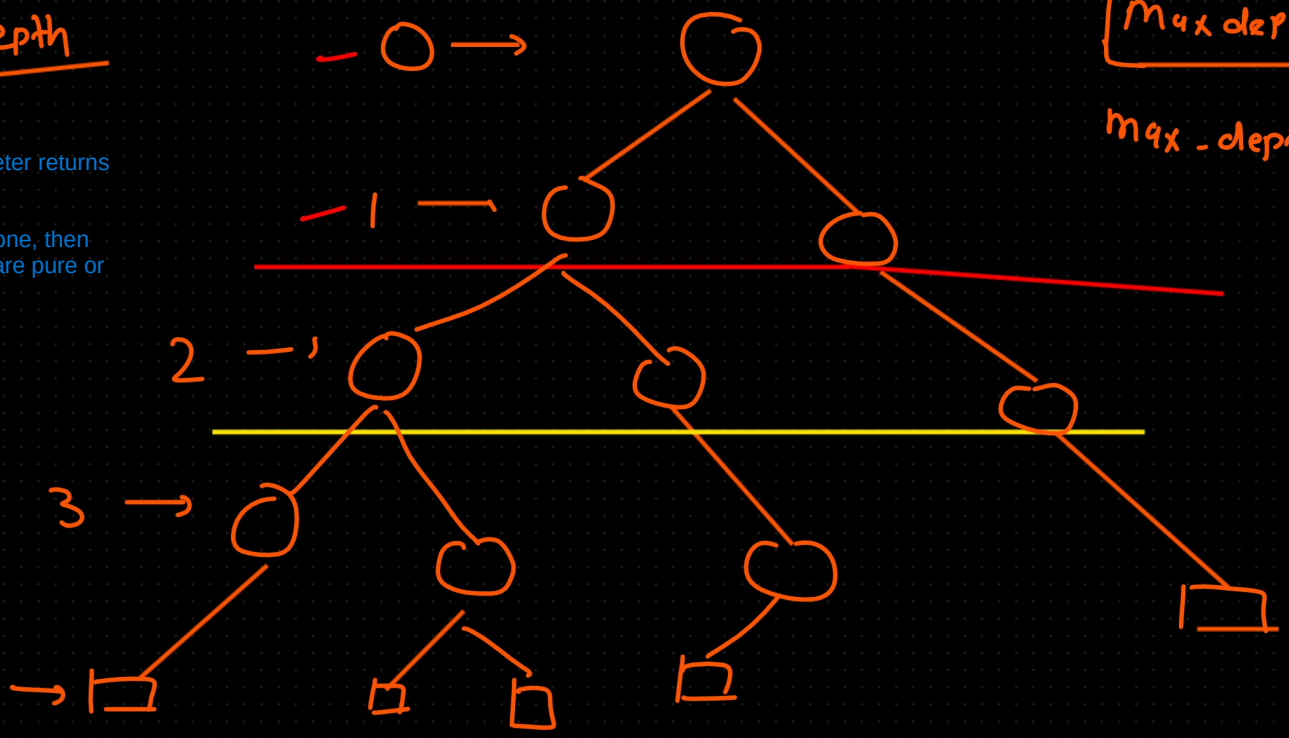
mss = 18



Can see this blog once from towards data science:

https://medium.com/analytics-vidhya/post-pruning-and-pre-pruning-in-decision-tree-561f3df73e65

① Max-Depth



max_depth: int (meaning this parameter returns integer), default=None

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

max depth = 2

max-depth = 3

Source for all the definition of hyperparameters mentioned here is:

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

② minimum sample-leaf

Constraint-

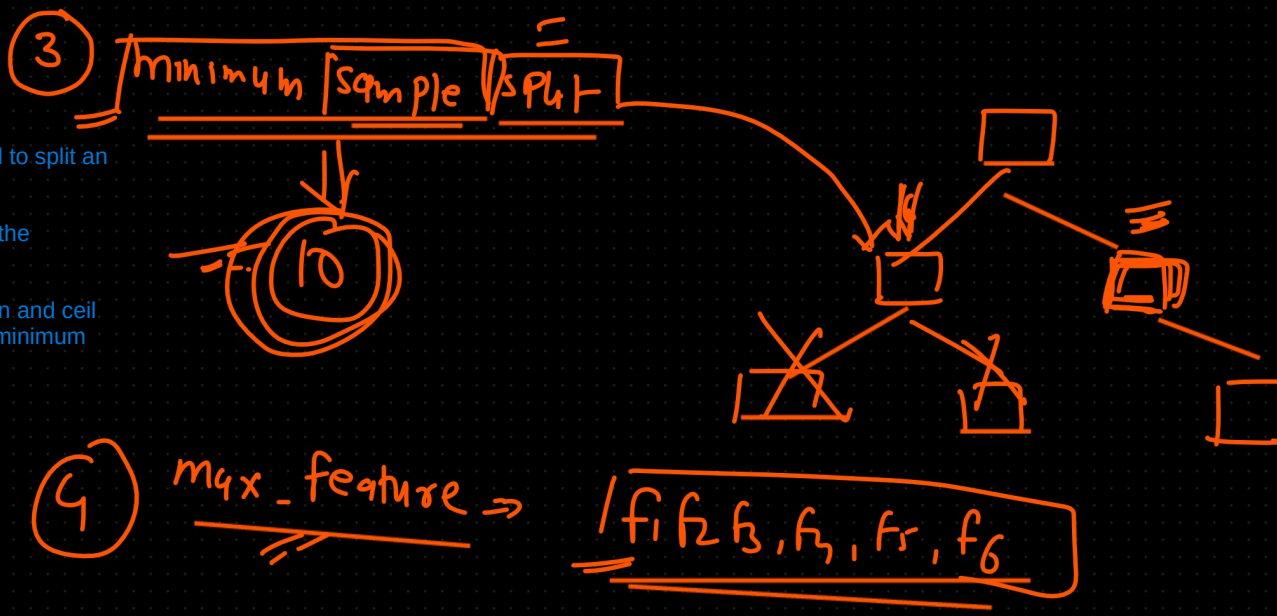
msl = 14

min_samples_leaf: int or float, default=1

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

If int, then consider min_samples_leaf as the minimum number.

If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.



min_samples_split: int or float, default=2
The minimum number of samples required to split an internal node:

If int, then consider min_samples_split as the minimum number.

If float, then min_samples_split is a fraction and ceil (min_samples_split * n_samples) are the minimum number of samples for each split.

NOTE that GridSearchCV is the optimization technique which can be used for pre-pruning hyper tuning for obtaining the best parameters using which one can build the best or optimized decision tree.

See Lecture-4 (62)/IG_and_DT_implementation/ decisio tree.ipynb for DTC implementation.

max_features: int, float or {“auto”, “sqrt”, “log2”}, default=None
The number of features to consider when looking for the best split:

If int, then consider max_features features at each split.

If float, then max_features is a fraction and max(1, int(max_features * n_features_in_)) features are considered at each split.

If “auto”, then max_features=sqrt(n_features).

If “sqrt”, then max_features=sqrt(n_features).

If “log2”, then max_features=log2(n_features).

If None, then max_features=n_features.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features.

Ques: Why we don't need to perform the feature scaling in case of decision tree?
OR
Why decision tree is not impacted by the outliers or feature scaling?

Ans: Decision trees and ensemble methods do not require feature scaling to be performed as they are not sensitive to the variance in the data.
Decision tree is condition based approach in which their splits don’t change with any monotonic transformation. Whereas, in other distance based approach like regression, k means clustering, neural network on applying monotonic transformations predicted values will change(or sensitive to variance) which in case of decision tree is not happening.

Source: https://towardsdatascience.com/do-decision-trees-need-feature-scaling-97809eaa60c6#:~:text=Decision%20trees%20and%20ensemble%20methods%20do%20not%20require%20feature%20scaling,the%20variance%20in%20the%20data.

AND

https://forecastegy.com/posts/do-decision-trees-need-feature-scaling-or-normalization/#:~:text=In%20general%2C%20no.,change%20with%20any%20monotonic%20transformation.