Decision Trees

Decision trees is a supervised machine learning algorithm which can be used for both classification and regression tasks. Sometimes, we often refer to its CART trees. These are intuitive in nature and can be easily explained.

Advantages

* For using a decision tree we require very minimal data preparation and there is no such need of feature scaling.
** The cost of using the tree for inference is 'logarithmic' in the number of data points used to train the tree.

Disadvantages

→ Decision trees are mostly prone to overfitting.
→ Also these are prone to errors in case of imbalanced datasets.

Process of creating decision trees: For creating a decision tree, we need to know about three terminologies —

(a) **Entropy:** Entropy is the measure of randomness in a data. Conversely, we can say entropy is the measure of impurity in a data.
The more the data is distributed / random / impure the higher the entropy of that data will be.
The mathematical formula for entropy is given as —

$$E(s) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

[It is sum across all the classes]
[p_i is the probability of an element/class 'i' in our data]

Suppose, we have two classes (Yes & No), then Entropy of this data can be calculated as —

$$E(s) = -p_{yes} \log_2(p_{yes}) - p_{no} \log_2(p_{no})$$

For three classes, yes, no & not sure, Entropy will be —

$$E(s) = -p_{yes} \log_2(p_{yes}) - p_{no} \log_2(p_{no}) - p_{not\ sure} \log_2(p_{not\ sure})$$

⇒ Some key observations —

(a) More the uncertainity/impurity in a dataset more will be the entropy.

(b) For a binary-class problem, the entropy lies between 0 and 1.

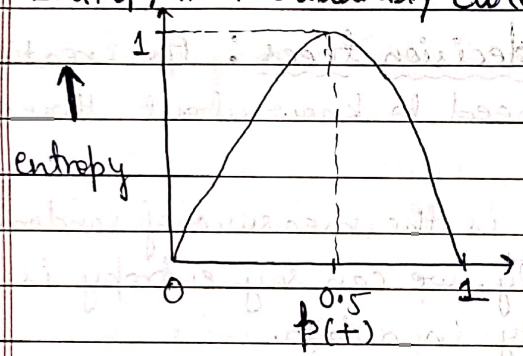
→ The entropy will be 0, when the data is completely pure i.e., it belongs to any one class completely.

→ The entropy will be 1, when the data belongs to both the classes equally.

(c) For more than two classes, minimum entropy can be 0 but maximum entropy can be greater than 1.

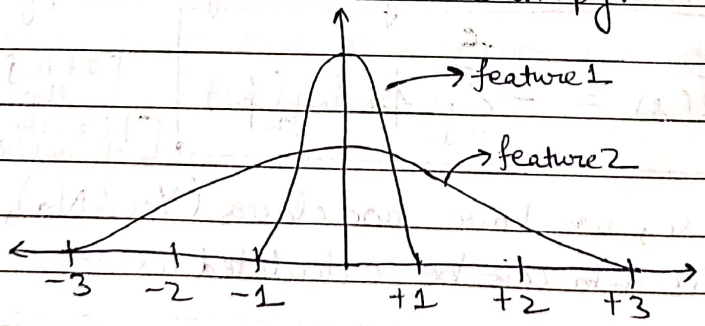
(d) We can use both $\log_2$ and $\log_e$ for calculating entropy.

Entropy vs Probability curve →



For a binary class, The entropy will be maximum when probability of positive and negative class is equal.

(e) For numerical/continuous variables, we can plot a KDE plot and the variable whose peakedness is less will have more entropy.



since, feature 2 has less peakedness its entropy is higher.

(b) Information Gain: Information gain measures the quality of a split. The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain.

The formula for information gain is given as —

$$\boxed{I.G. = E(\text{parent}) - \text{Weighted Avg.} * E(\text{Children})}$$

→ Decision tree applies a recursive greedy search algorithm in top bottom fashion to find information gain at every level of the tree. Once a leaf node is reached (Entropy = 0), no more splitting is done.

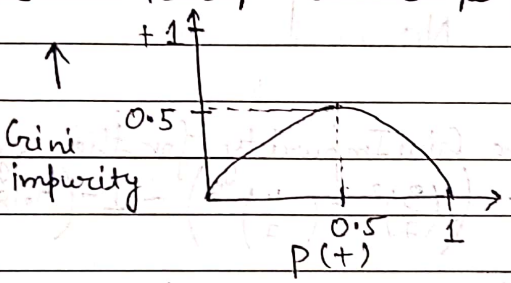
(c) Gini Impurity: It is also a measure of quality of split i.e., it also helps us find out the amount of impurity in our data. The formula for ~~entr~~ gini impurity is given as —

$$\boxed{Gi = 1 - (py^2 + pn^2)}$$

py → probability of positive class
pn → probability of negative class

$$\boxed{I.G. = G_i(\text{Parent}) - \text{weighted Avg. } G_i(\text{Child})}$$

Gini Impurity vs Probability Curve



For a binary class problem, the gini impurity will be maximum when positive and negative class is equal.

Imp

Ques. Why do we use gini impurity when we have entropy?

Ans→ Gini impurity and Entropy both are a measure of quality of split. But calculating gini impurity is less computationally expensive than entropy as it contains calculation of log as well, therefore computation of gini impurity is faster as well.

→ But in certain datasets it is proven that using entropy leads to a formation of more balanced trees, so sometimes it is also better to use entropy as the criterion of split.

Creation of decision trees for categorical columns and numerical columns

Using ID3 (Iterative Dichotomiser 3) to build decision trees: This algorithm is named such because it iteratively (repeatedly) dichomatizes (divides) features into two or more groups at each stage. ID3 uses a top-down greedy approach to build decision trees. Let us have a look at one dataset using which we can make a decision tree:-

Age	Gender	Likes Cartoon	Likes Batman
20	M	Yes	Yes
30	M	No	Yes
18	F	Yes	No
40	F	Yes	No
60	M	No	Yes
80	M	Yes	No
30	F	No	No

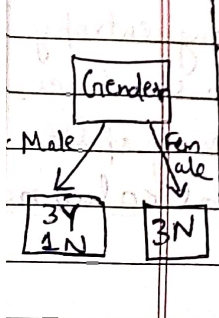
① Step 1 → Calculate the Gini Impurity for the system.

$$G_{system} = 1 - \left(\left(\frac{3}{7}\right)^2 + \left(\frac{4}{7}\right)^2\right) = 1 - (0.1836 + 0.326$$

$$= 1 - (0.51) = 0.49$$

② Step 2 → We need to decide which node should be selected as the root node (amongst present 3 columns). So, we calculate Gini impurity for all of them one by one —

(a) Taking Gender column —



Gender	Likes Batman
M	Y
M	Y
F	N
F	N
M	Y
M	N
F	N

$G_{Male} = 1 - \left(\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right)$

$= 1 - (0.5625 + 0.0625)$

$= 1 - 0.625$

$G_{male} = 0.375$

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

$= 1 - (0 + 1)$

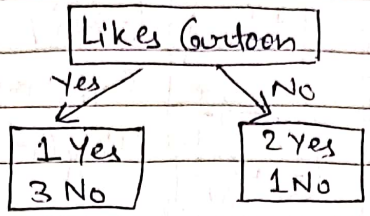
$G_{Female} = 0$

Weighted Avg. $G_{gender} = \frac{4}{7} * (0.375) + \frac{3}{7} * (0) = 0.214$

Total gini impurity for Gender = 0.214

(b) Taking Likes Cartoon column.

Likes Cartoon	Likes Batman
Yes	Yes
No	Yes
Yes	No
Yes	No
No	Yes
Yes	No
No	No



$G_{Yes} = 1 - \left[\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right] = 1 - [0.0625 + 0.5625]$

$G_{Yes} = 0.375$

$G_{No} = 1 - \left[\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right] = 1 - \frac{5}{9} = \frac{4}{9} = 0.444$

$W_{Avg}\ G_{likes\ cartoon} = \frac{4}{7} \times 0.375 + \frac{3}{7} * 0.414$

$G_{likes\ cartoon} = 0.214 + 0.19 = 0.404$

(c) Taking Age column —

Age	Likes Batman
20	Yes
30	Yes
18	No
40	No
60	Yes
80	No
30	No

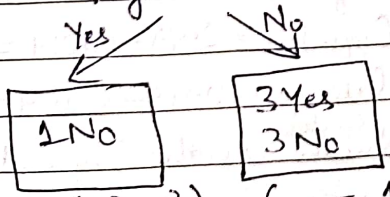
(i) for numerical columns, the first step is to sort them in ascending order.

(ii) second step is to calculate average of numerical values for consecutive numbers.

	Age	Likes Batman	
19	18	No	
	20	Yes	25
	30	Yes	
30	30	No	
	40	No	35
50	60	Yes	70
	80	No	

(iii) In the third step, we can create different stages and calculate gini impurity for each stage.

Age <= 19



$G_{Yes} = 1 - (0^2 + 1^2)$

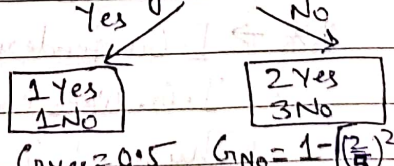
$G_{Yes} = 0$

$G_{No} = 1 - \left(\left(\frac{3}{6}\right)^2 + \left(\frac{3}{6}\right)^2\right)$

$G_{No} = 1 - \frac{1}{2} = 0.5$

$W_{avg}G = \frac{1}{7} \times 0 + \frac{6}{7} \times 0.5 = 0.4285$

Age <= 25



$G_{Yes} = 0.5$

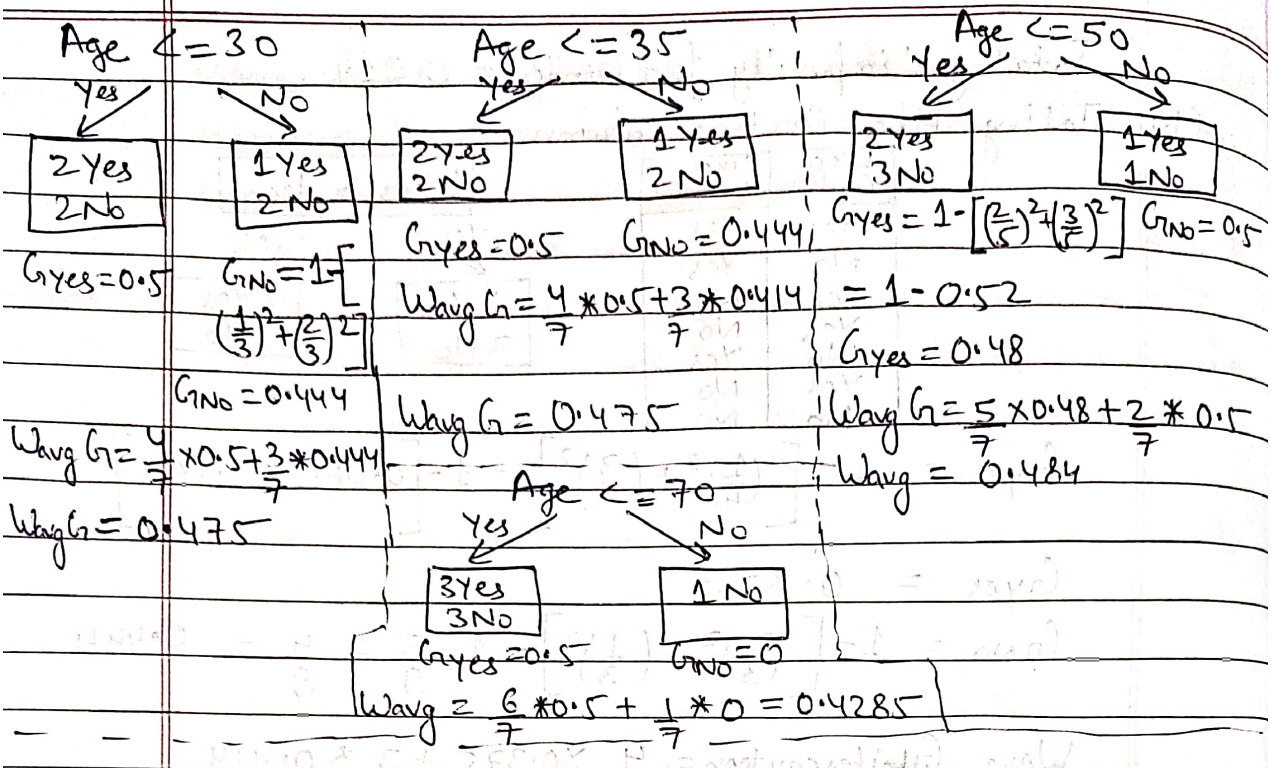
$G_{No} = 1 - \left[\left(\frac{2}{5}\right)^2 + \left(\frac{3}{5}\right)^2\right]$

$= 1 - 0.52$

$G_{No} = 0.48$

$W_{avg}\ G = \frac{2}{7} * 0.5 + \frac{5}{7} * 0.48$

$W_{avg}\ G = 0.4844$



(iv) In the fourth step, we have to select that age condition where Wavg Gr was minimum, in our case for condition Age <=19 and Age <= 70, Wavg Gr was the lowest i.e., 0.4285

(3) Now, we have Gini impurity for all the columns

$G_{rage} = 0.4285$, $G_{rgender} = 0.214$, $G_{rlikes\ cartoon} = 0.404$

Out of these three, we can select the node for which gini index was minimum or conversely for which Information gain will be maximum.

$I.G_{r.age} = G_{rsystem} - G_{rage} = 0.49 - 0.4285 = 0.0615$

** → $I.G_{r.gender} = G_{rsystem} - G_{rgender} = 0.49 - 0.214 = 0.276$

$I.G_{r.likes\ cartoon} = G_{rsystem} - G_{rlikes\ cartoon} = 0.49 - 0.404 = 0.086$

Thus, I Gr for Gender column is the highest, so we can select it for the root node splitting.

Now, we have to repeat this process iteratively till we get all the leaf nodes.

Now after deciding the root node as Gender, we are left with the following dataset

Age	Likes Cartoon	Likes Game
20	Yes	Yes
30	No	Yes
60	No	Yes
80	Yes	No

We find Gini impurity for Likes Cartoon column —

Likes Cartoon
Yes → 1 Yes, 1 No → $G_r = 0.5$
No → 2 Yes → $G_r = 0$

$$W_{avg} = \frac{2}{4} \times 0.5 + \frac{2}{4} \times 0 = 0.25$$

Grage ⇒

Age	LB
20	Yes
30	Yes
60	Yes
80	No

25 | 45 | 70

Age <= 25
Yes → 1 Yes → $G = 0$
No → 2 Yes, 1 No → $G = 0.444$

$$W_{avg} = \frac{1}{4} * 0 + \frac{3}{4} * 0.444$$

$$G_{Wavg} = 0.333$$

Age <= 45
2 Yes → $G = 0$
1 Yes, 1 No → $G = 0.5$

$$W_{avg} = \frac{2}{4} * 0 + \frac{2}{4} * 0.5$$

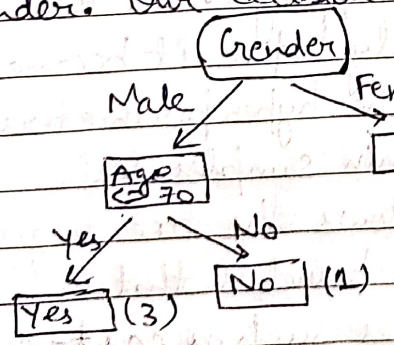
$$G_{Wavg} = 0.25$$

Age <= 70
Yes → 3 Yes → $G = 0$
No → 1 No → $G = 0$

$G_{Wavg} = 0$

Since for Age <= 70, we have least Gini impurity, we select it.

Now, we have Gini impurity for Likes cartoon = 0.25 and Age = 0, we know IG for Age will be higher so our decision tree can be split by Age after the Gender. Our decision tree will look like —



Problem of overfitting in decision trees:

Overfitting in decision trees occurs when the model learns not just the underlying patterns in the training data but also the noise or random fluctuations. This leads to a model that performs exceptionally well on a training data but poor performance on unseen (test) data, as it fails to generalize.

Ques. How overfitting happens in decision tree?

Ans→ ① Too deep trees: When a decision tree is allowed to grow without constraints, it may split until each node is converted into a leaf node containing a single observation, perfectly classifying the training data. This results in a highly complex tree which learns nois

② Irrelevant features: Including irrelevant or weakly correlated features can lead to splits that don't generalize well to new data.

Preventing Overfitting in Decision Trees

Ⓐ Pruning of trees: We can stop the tree to grow using certain constraints so that it does not learn the noise as well. We can perform both pre-pruning as well as post-pruning to stop or limit the depth of decision trees.

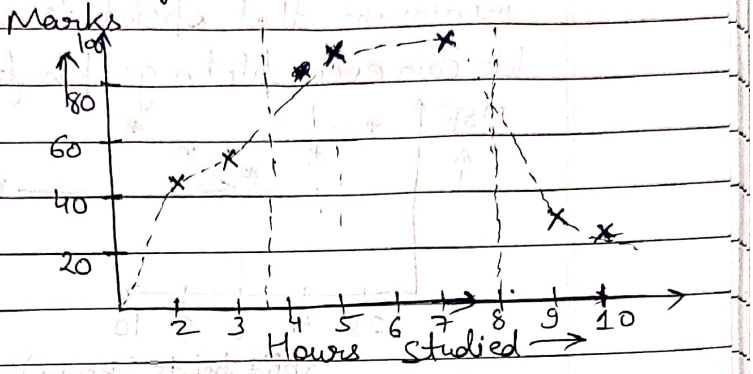
- Pre-Pruning (Early Stopping): In this we stop growing the tree early before it becomes too complex. This can be done by the hyperparameters like max_depth, min_samples_split, min_samples_leaf.
- Post-Pruning: It allows the tree to grow fully and then prune back the branches that have little importance. Post pruning techniques such as **cost-complexity pruning (ccp)**, remove branches that add minimal value

Ⓑ Ensemble methods Ⓒ Cross-validation

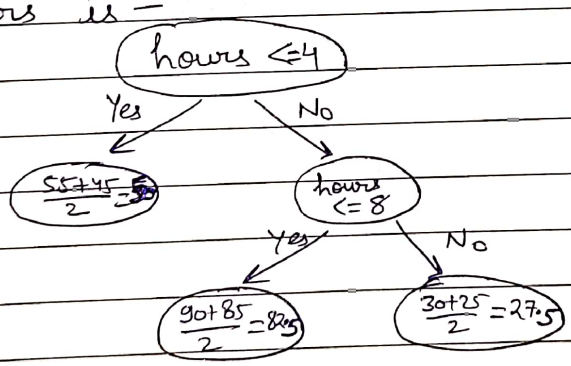
Ⓓ Feature Selection

Regression Trees : As we discussed already, decision trees can be used for regression tasks as well i.e., predicting continuous values. Suppose, we have a problem to detect marks obtained by students in a university exam given the following data.

Hours	Marks
2	45
3	55
5	85
7	90
9	30
10	25



In this type of data, we cannot simply fit a linear regression model as the data clearly is non-linear. In these type of cases, we can take help of regression trees, which draw multiple decision boundaries and help us in prediction. A sample decision tree for predicting suppose marks of a student who has studied for 6 hours is —



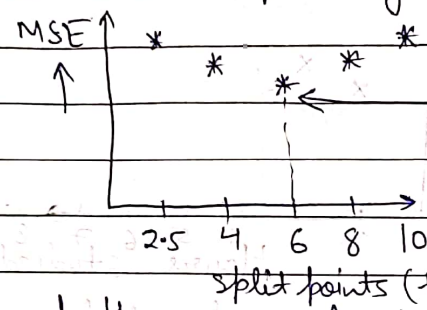
In regression trees, while drawing decision boundaries data points are sub-divided into clusters and then mean of those data points is taken as the value for prediction. But the question is how do we decide where to split for making these decision boundaries.

* In regression trees, we use a method known as **variance reduction** for getting the optimal split point.

For our dataset, we take consecutive values and then find their mean and make a condition and check what is the variance. and then repeat the process for all pairs. The pair at which the variance/ MSE is minimum that point is chosen for making a split.

Hours	Marks
2	45
3	55
5	85
7	90
9	30
10	25

We can even plot a graph for different mean values as—



hence we can choose, hours <= 6 as the first condition for making a split and then we repeat this process iteratively.

→ When there are more than one feature, we calculate these MSE for all features independently and then compare the MSE (least) of all features, then the feature who has least MSE is used for making a split.