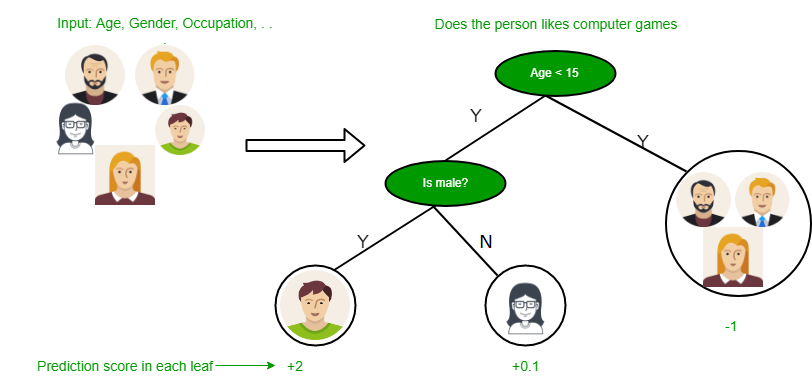
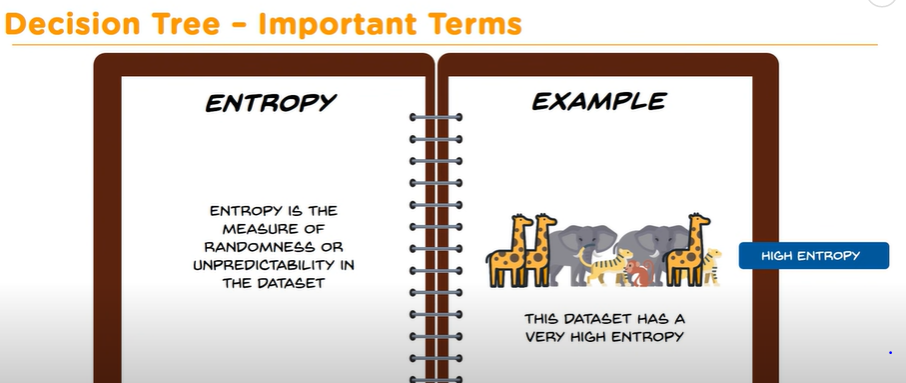
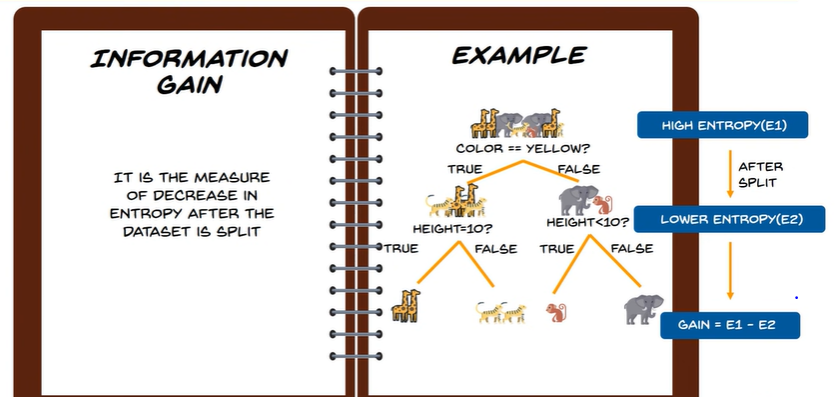
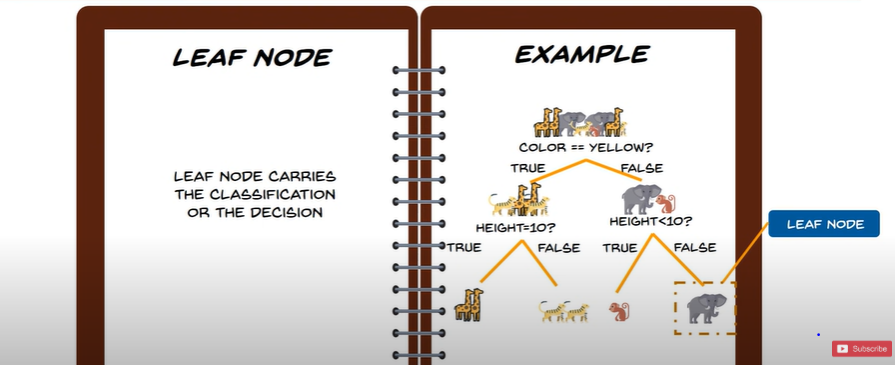


* Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.
* Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.
* We can represent any boolean function on discrete attributes using the decision tree.

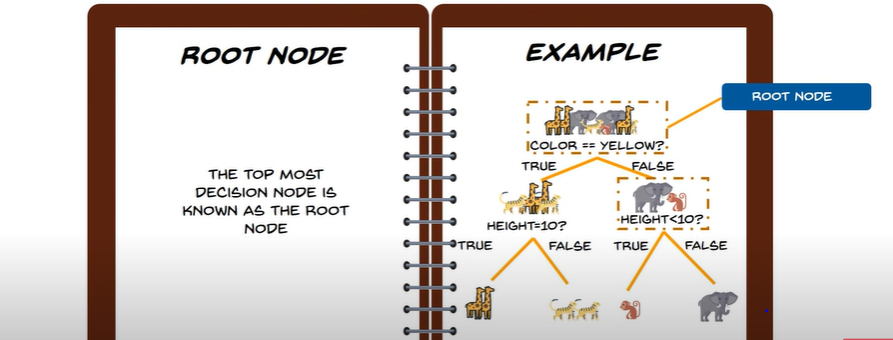








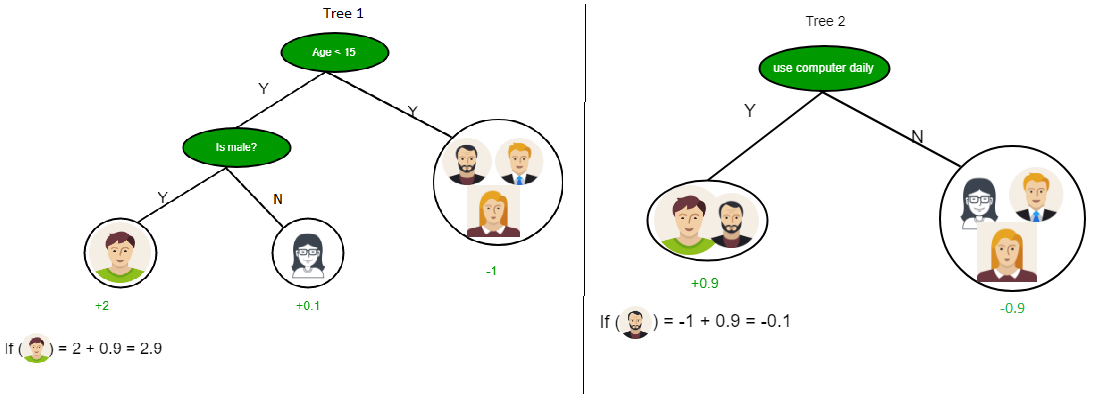
Thus, Leaf Node is a the final end or at bottom.



Also, There is Decision Node is one which has two or more Branches. This is where we break the group into two or more Parts.

**Below are some assumptions that we made while using decision tree:**

* At the beginning, we consider the whole training set as the root.
* Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
* On the basis of attribute values records are distributed recursively.
* We use statistical methods for ordering attributes as root or the internal node.



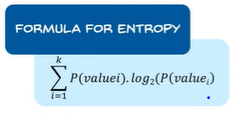
As you can see from the above image that Decision Tree works on the Sum of Product form which is also known as *Disjunctive Normal Form*. In the above image, we are predicting the use of computer in the daily life of the people.

In Decision Tree the major challenge **is to identification of the attribute** for the root node in each level, **such that after splitting the Entropy decreases. This process is known as attribute selection. We have two popular attribute selection measures:**

1. **Information Gain**
2. **Gini Index**

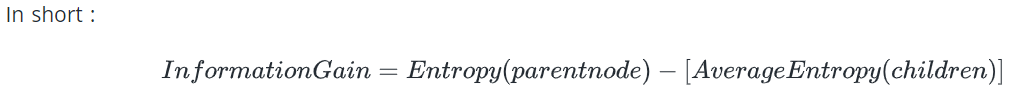
**1. Information Gain**  
When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy. Ideally, the splitting would be done based on such an Attribute which results in decrease in Entropy.

**Entropy**  
Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.



Where, P(value i) = percentage of ith group .

***Formula of Gain***: Suppose S is a set of instances, A is an attribute, Sv is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

**Example:**

**For the set X = {a,a,a,b,b,b,b,b}**

Total instances: 8

Instances of b: 5

Instances of a: 3



*Because,*

*P(value for attribute ‘instance’) = 3/8 , and*

*P(value for attribute ‘instance’) = 5/8*

= -[0.375 \* (-1.415) + 0.625 \* (-0.678)]

= -(-0.53-0.424)

= 0.954

**Building Decision Tree using Information Gain**  
**The essentials:**

* Start with all training instances associated with the root node
* Use info gain to choose which attribute to label each node with
* *Note:* No root-to-leaf path should contain the same discrete attribute twice
* Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.

**The border cases:**

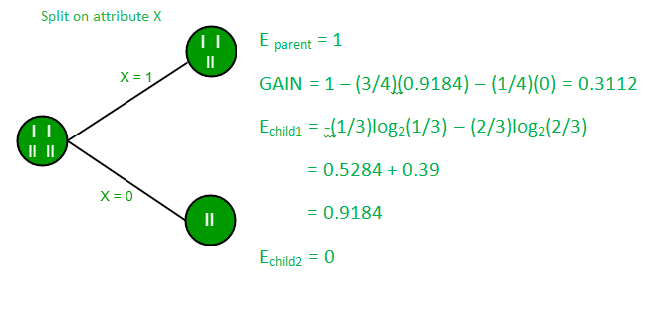
* If all positive or all negative training instances remain, label that node “yes” or “no” accordingly
* If no attributes remain, label with a majority vote of training instances left at that node
* If no instances remain, label with a majority vote of the parent’s training instances

**Example:**  
Now, lets draw a Decision Tree for the following data using Information gain.

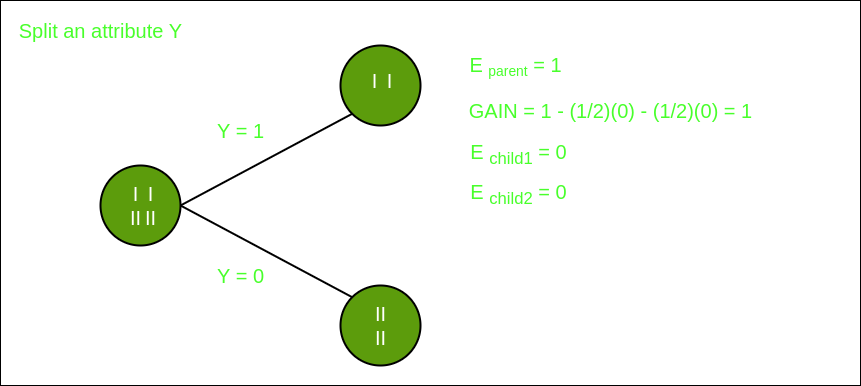
**Training set: 3 features and 2 classes**

| X | Y | Z | C |
| --- | --- | --- | --- |
| 1 | 1 | 1 | I |
| 1 | 1 | 0 | I |
| 0 | 0 | 1 | II |
| 1 | 0 | 0 | II |

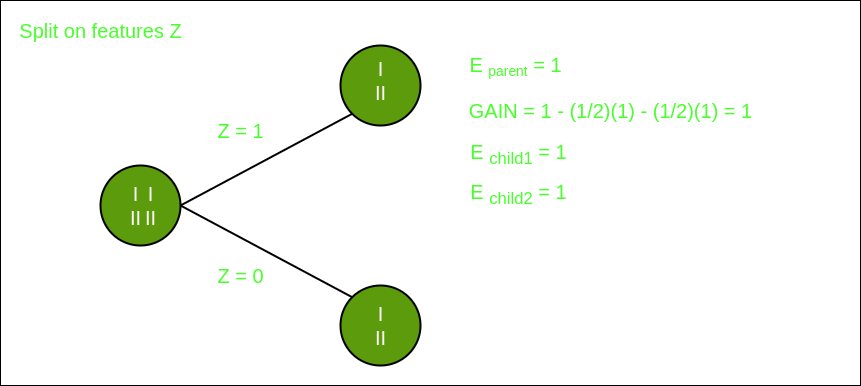
Here, we have 3 features and 2 output classes.

To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.  


**Split on feature X**

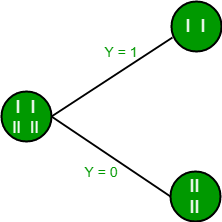


**Split on feature Y**



**Split on feature Z**

From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains pure subset of the target variable. So we don’t need to further split the dataset.

The final tree for the above dataset would be look like this:  
  
**2. Gini Index**

* Gini Index is a metric to measure how often a randomly chosen element be incorrectly identified.
* It means **an attribute with lower Gini index should be preferred.**
* Sklearn supports “Gini” criteria for Gini Index and by default, it takes “gini” value.

The Formula for the calculation of the Gini Index is given below.



**Example:**  
Lets consider the dataset in the image below and draw a decision tree using gini index.

| Index | A | B | C | D | E |
| --- | --- | --- | --- | --- | --- |
| 1 | 4.8 | 3.4 | 1.9 | 0.2 | positive |
| 2 | 5 | 3 | 1.6 | 1.2 | positive |
| 3 | 5 | 3.4 | 1.6 | 0.2 | positive |
| 4 | 5.2 | 3.5 | 1.5 | 0.2 | positive |
| 5 | 5.2 | 3.4 | 1.4 | 0.2 | positive |
| 6 | 4.7 | 3.2 | 1.6 | 0.2 | positive |
| 7 | 4.8 | 3.1 | 1.6 | 0.2 | positive |
| 8 | 5.4 | 3.4 | 1.5 | 0.4 | positive |
| 9 | 7 | 3.2 | 4.7 | 1.4 | negative |
| 10 | 6.4 | 3.2 | 4.7 | 1.5 | negative |
| 11 | 6.9 | 3.1 | 4.9 | 1.5 | negative |
| 12 | 5.5 | 2.3 | 4 | 1.3 | negative |
| 13 | 6.5 | 2.8 | 4.6 | 1.5 | negative |
| 14 | 5.7 | 2.8 | 4.5 | 1.3 | negative |
| 15 | 6.3 | 3.3 | 4.7 | 1.6 | negative |
| 16 | 4.9 | 2.4 | 3.3 | 1 | negative |

In the dataset above there are 5 attributes from which attribute E is the predicting feature which contains 2(Positive & Negative) classes. We have an equal proportion for both the classes.

Step 1 : First we should decide which of the Attribute A, B , C or D does a better Job at classifying the Predicting feature( ) E as Positive or Negative .

***There are several ways to do this like Information Gain or Gini Impurity/Gini Index***  .

Here, we Discuss Gini Impurity/Gini Index Method.  
In Gini Index, when we have Values (instead of attributes like Love/Hate, Big/Small etc.), we have to choose some random values to categorize each attribute.

Say,These values for this dataset are:

A B C D

>= 5 >= 3.0 >= 4.2 >= 1.4

< 5 < 3.0 < 4.2 < 1.4

**Calculating Gini Index for Attribute A:**

Based on >/< 5, we classify the dataset of A and E into two “leaves” :

Leaf 1: with Value >5 and 12 entries.

Leaf 2 : with Value < 5 and 4 entries**.**   
**Value >= 5: with 12 entries in this leaf 1.**  
Attribute A >= 5 & class = positive: 5/12  
Attribute A >= 5 & class = negative: 7/12

Gini(5, 7) = 1 –  

**Value < 5: with 4 entries in this leaf 2.**  
Attribute A < 5 & class = positive: 3/4  
Attribute A < 5 & class = negative: 1/4  
Gini(3, 1) = 1 – 

Now, since both Leaves have different number of entries in them (i.e. 12 and 4), So we take their **Weightage to get Total Gini Impurity or Total Gini Index** ::  
By Multiplying with weight and summing each of the gini indices, we get **Total Gini Impurity/Total Gini Index** for A , as shown below:   


**Calculating Gini Index for Attribute B:**

Again, Based on >/< 3, we classify the dataset of B and E into two “leaves” :

Leaf 1: with Value >=3 and with 12 entries.

Leaf 2 : with Value < 3 and with 4 entries**.**  
**Value >= 3: with 12 entries in this leaf**  
Attribute B >= 3 & class = positive: 8/12  
Attribute B >= 3 & class = negative: 4/12  
Gini(5, 7) = 1 – 

**Value < 3: with 4 entries in this leaf**  
Attribute A < 3 & class = positive: 0/4  
Attribute A < 3 & class = negative: 4/4  
Gini(3, 1) = 1 – 0

Again, since both Leaves have different number of entries in them (i.e. 12 and 4), By multiplying these values with their resp. weights and sum each of the them, we get **Total Gini Impurity/ Total Gini Index** for B : ---  


Using the same approach we can calculate the Gini index for C and D attributes.

Positive Negative

For A|>= 5.0 5 7

|<5 3 1

Gini Index of A = 0.45825

Positive Negative

For B|>= 3.0 8 4

|< 3.0 0 4

Gini Index of B= 0.3345

Positive Negative

For C|>= 4.2 0 6

|< 4.2 8 2

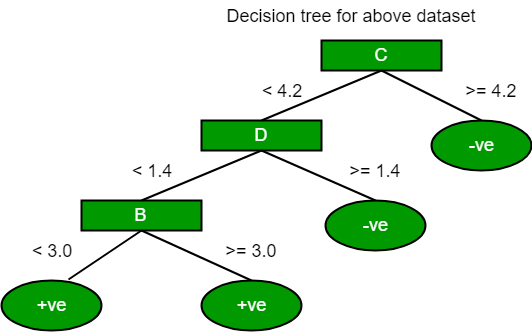
Gini Index of C= 0.2

Positive Negative

For D|>= 1.4 0 5

|< 1.4 8 3

Gini Index of D= 0.273



The most notable types of decision tree algorithms are:-

1. **Iterative Dichotomiser 3 (ID3):**

This algorithm uses Information Gain to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.

1. **C4.5:**

This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.

1. **Classification and Regression Tree(CART):**

It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

**Decision Tree**

**Decision Tree** is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



*A decision tree for the concept PlayTennis.*

**Construction of Decision Tree:** A tree can be *“learned”* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called*recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

**Decision Tree Representation:** Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.   
The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returns the classification associated with the particular leaf.(in this case Yes or No).

For example, the instance 

*(Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong )*

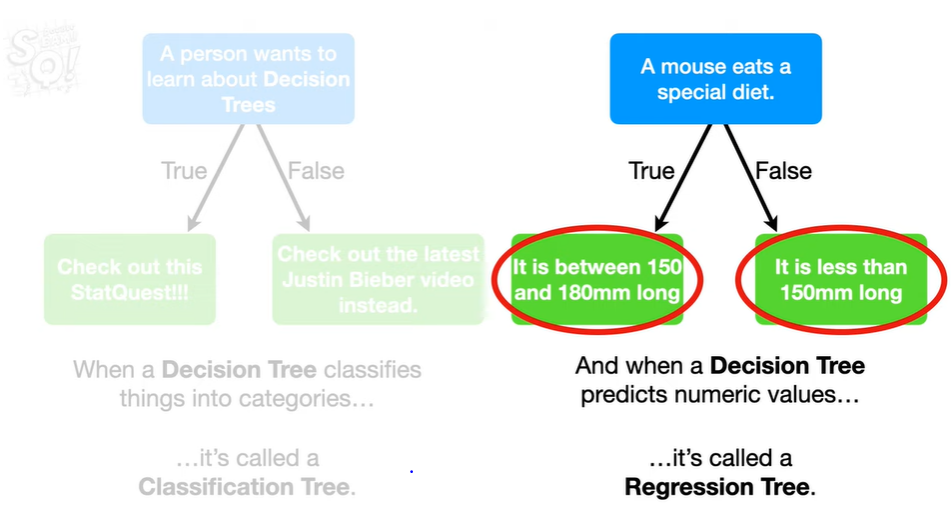
would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance.   
In other words, we can say that the decision tree represents a disjunction of conjunctions of constraints on the attribute values of instances.

*(Outlook = Sunny ^ Humidity = Normal) v (Outlook = Overcast) v (Outlook = Rain ^ Wind = Weak)*

**Gini Index:**

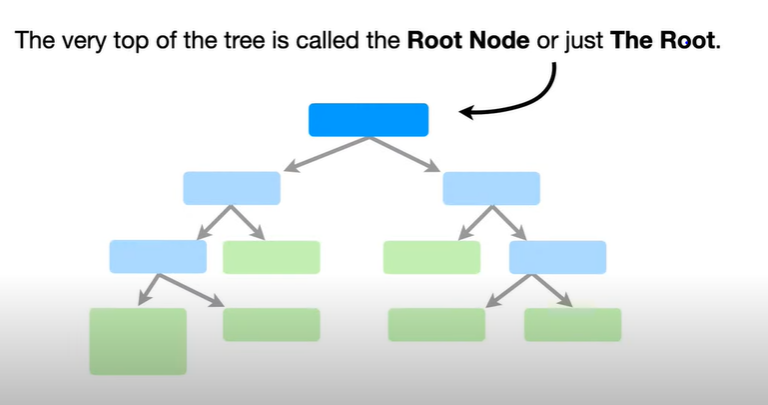
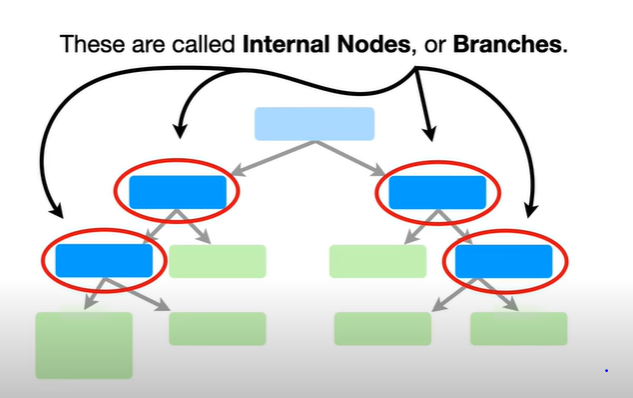
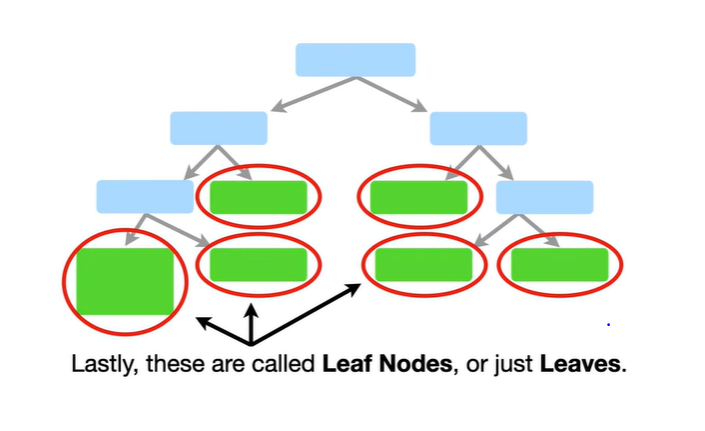
Gini Index is a score that evaluates how accurate a split is among the classified groups. Gini index evaluates a score in the range between 0 and 1, where 0 is when all observations belong to one class, and 1 is a random distribution of the elements within classes. In this case, we want to have a Gini index score as low as possible. Gini Index is the evaluation metrics we shall use to evaluate our Decision Tree Model.

**2 Types of Decision Tree Problems : Classification and Regression**



Terminology:

1. Root Node:

1. Internal Nodes : These have branches coming to them and going away from them.
2. Leave Nodes: They have branches coming to them only.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

A general algorithm for a decision tree can be described as follows:

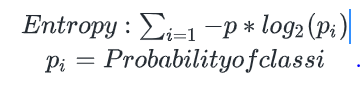
1. Pick the best attribute/feature. The best attribute is one which best splits or separates the data.
2. Ask the relevant question.
3. Follow the answer path.
4. Go to step 1 until you arrive to the answer.

The best split is one which separates two different labels into two sets. In Decision Tree the major challenge is to identification of the attribute for the root node in each level, such that **after splitting the Entropy decreases**. This process is known as attribute selection**. We have two popular attribute selection measures:**

**1. Information Gain**

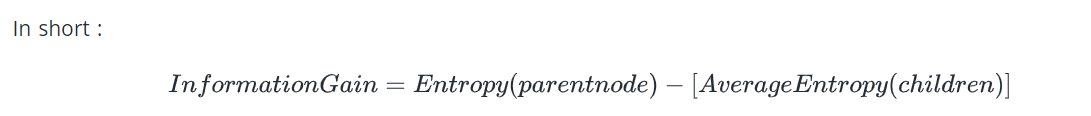
**2. Gini Index**

**Entropy that measures the level of impurity in a group of examples.** Mathematically, it is defined as:



***Information gain*** as a measure of the effectiveness of an attribute in classifying the training data. Information gain, *Gain (S, A)* of an attribute A, relative to a sample of examples *S*, is defined as:





Ideally**, Splitting is done so that Entropy after split decreases, or, we can say that, the Gain is Maximum.**

Pruning: Reduces the Nodes in order to get the Optimal Solution. Thus pruning reduces the complexity.

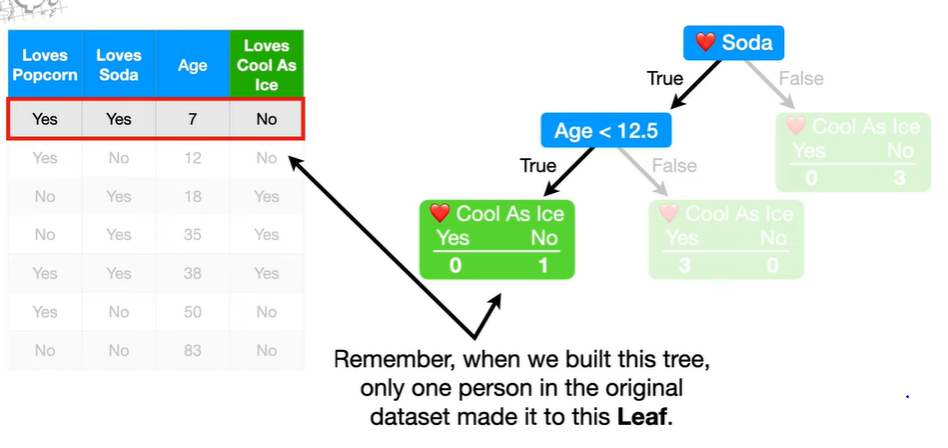
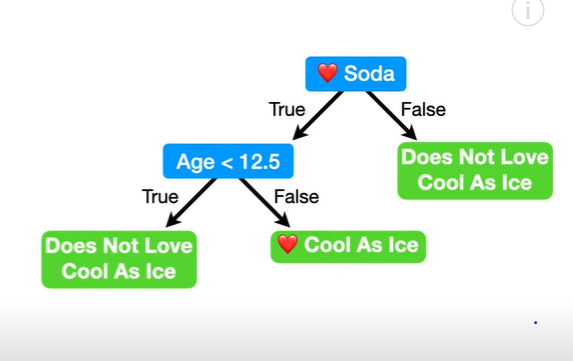
**WHEN TO USE TREE BASED MODEL ?**

**WHY NOT USE LOGISTIC REGRESSION FOR ALL CLASSIFICATION PROBLEMS ?**

1. If the relationship between dependent and Independent Variable can be well approximated (i.e. there exists a clear relationship ) , then Linear Model will out-perform Tree Based Algorithm.
2. If there is High Non-linearity and complex relation between dependent and Independent Variable, then, a Tree based model is preferred.
3. Decision tree models are used where we need to explain to other people.

**OVERFITTING AND HOW TO AVOID IT :**

CONSIDER THE FOLLOWING CASE:

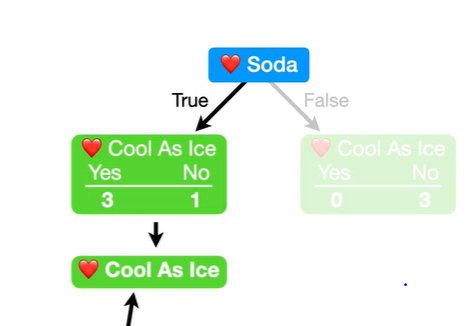
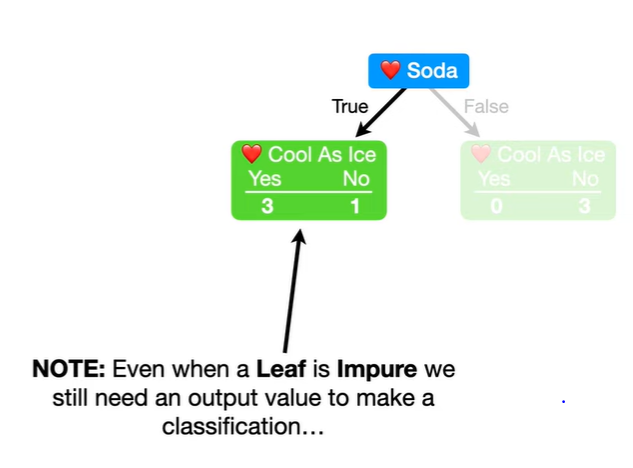
 

To Obtain the Above Fig, We kept going (i.e kept on splitting) until we had all the leaves as **Pure leaves** i.e. Either 100% of entry in each leaf either loved “Cool As Ice “ movie (see center Leaf), or, 100% of entry Did not love “Cool As Ice” movie ( see left and right Leaves).

This May lead to **Overfitting of Data** . (For example, Observe the left green Leave (Yes=0, No=1). We have used only 1 person while Training the model to get this leaf. This obviously may fail in Real life scenarios where we may need to classify thousands of data ).

This would not have happened , if we did not keep on going to get a Pure Leaf at the end i.e. if we could put a **Constraint on how many Minimum people(or entries) are needed in order to grow the Tree**.

Example : In this above example, we could put a limit of at Least 3 or more people per Leaf, in order to grow (i.e if a leaf has less than 3 people, it wont be created) Based on this criteria, the above fig. would NOT have the Left most Leaf and the New Updated Tree would look like : (see below) :



**New Updated Tree**

**In New Updated Tree,** in order to classify the 2 leaves as either Loves (or) Does Not Love *Cool as Ice* :

1. The left Leaf has 3 out of 4 values as Yes = Loves. So, it is classified as “Loves CAI”
2. The right Leaf has all 3 out of 3 as No = Does not Love. So, it is Classified as “Does Not Love CAI”

**Now , we have ended up with a Impure Leaf, However , this should gives us a better accuracy because we have Avoided Overfitting the data.**

**NOTE:** Another Method to Avoid Overfitting iscalled **Pruning .**

# What is Pruning?

**Pruning** is a technique that is used to reduce overfitting. Pruning also simplifies a decision tree by removing the weakest rules.

Pruning starts with an unpruned tree, takes a sequence of subtrees (pruned trees), and picks the best one through cross-validation.

Pruning should ensure the following:

* The subtree is optimal — meaning it has the highest accuracy on the cross-validated training set. (Trees can be optimized for whatever parameter is most important to the engineer — not always accuracy)
* The search for the optimal subtree should be computationally tractable.

In scikit-learnsDecisionTreeClassifier, ccp\_alphaIs the **cost-complexity parameter.**

Essentially, pruning recursively finds the node with the “weakest link.” The weakest link is characterized by an effective alpha, where the nodes with the smallest effective alpha are pruned first.

Mathematically, the **cost complexity measure** for a tree T is given by:



* **R(T)**— Total training error of leaf nodes or, SSR(sum of squared error)
* **|T|** — The number of leaf nodes
* **α**— complexity parameter(a whole number determined using GridSearch)

As alpha increases, more of the tree is pruned, which increases the total impurity of its leaves.

If we only try to reduce the training error **R(T)**, it will lead to relatively larger trees (more leaf nodes), resulting in overfitting.

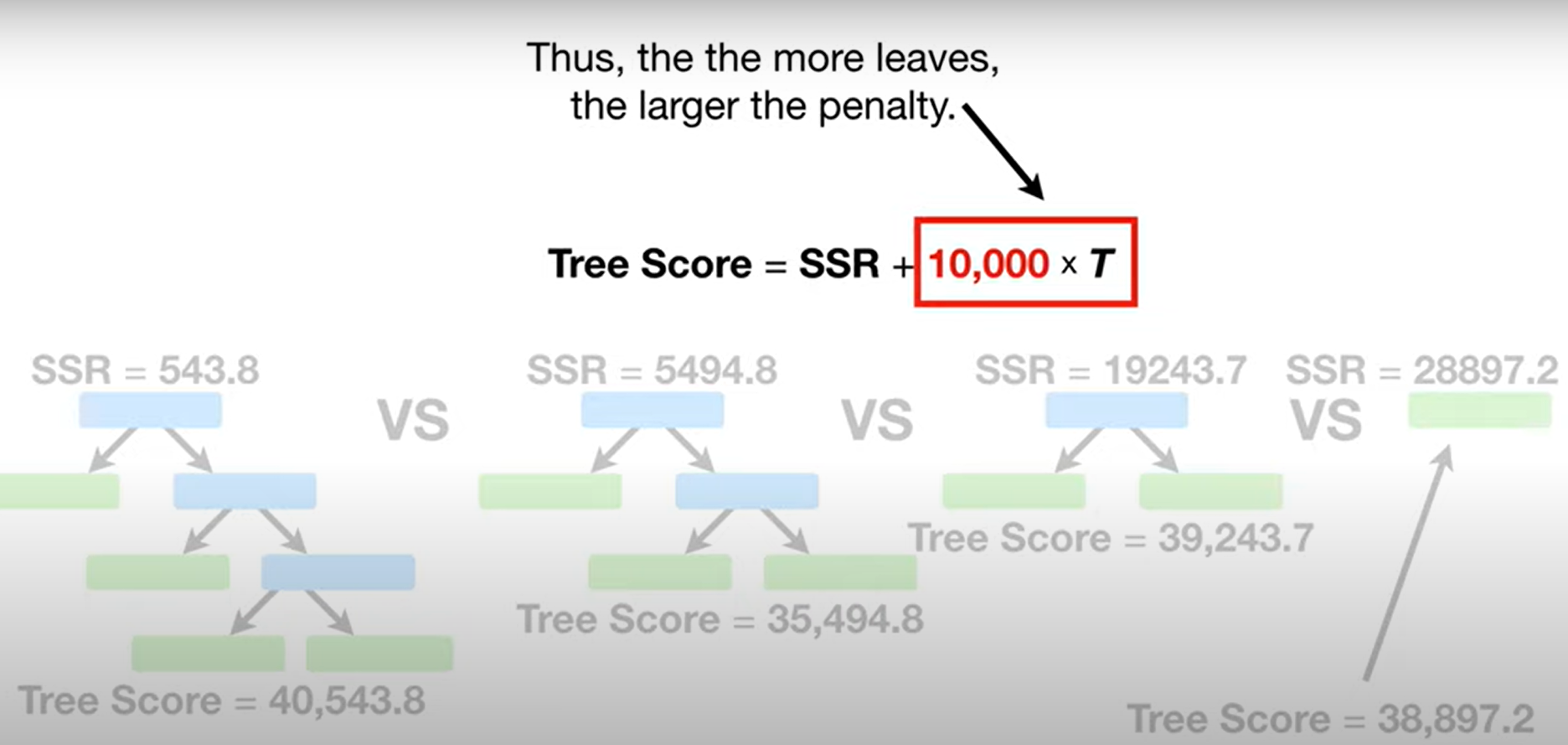
Cost complexity pruning generates a series of trees where cost complexity measure for sub-tree **Tₜ**is:



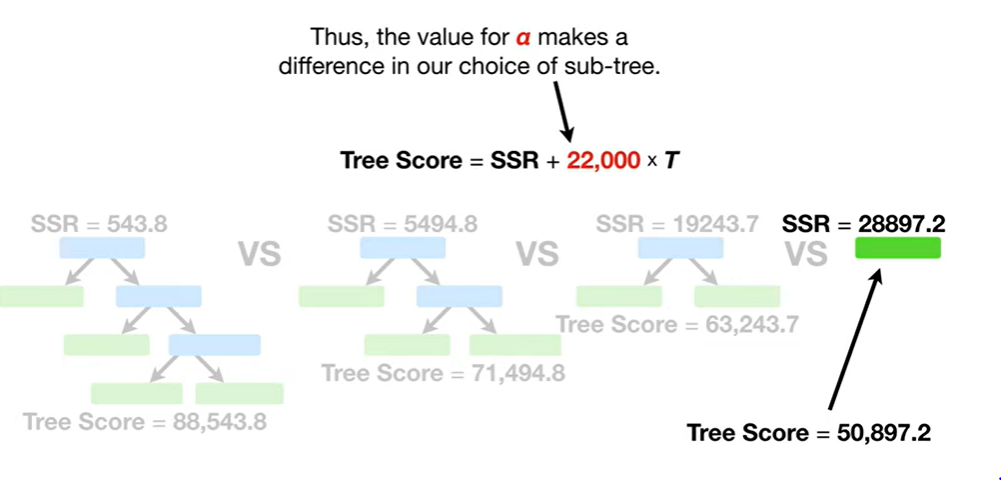
Example in Fig. Below:

We start with *a* alpha value of 10000, and get Ra(T) i.e. denoted as Tree Score for each of the Tree (original and pruned tree both).

Upon looking at the fig below , clearly the tree with the Least Ra(T) is chosen i.e. The Second pruned tree with 3 leaves in it.

The parameter **α**reduces the complexity of the tree by controlling the number of leaf nodes, which eventually reduces over-fitting.

NOTE :In above example , if a value changes to say 22000, then the last pruned tree with 1 leaf would become the best choice b/c it would then then have the least Tree Score or Ra(T) .



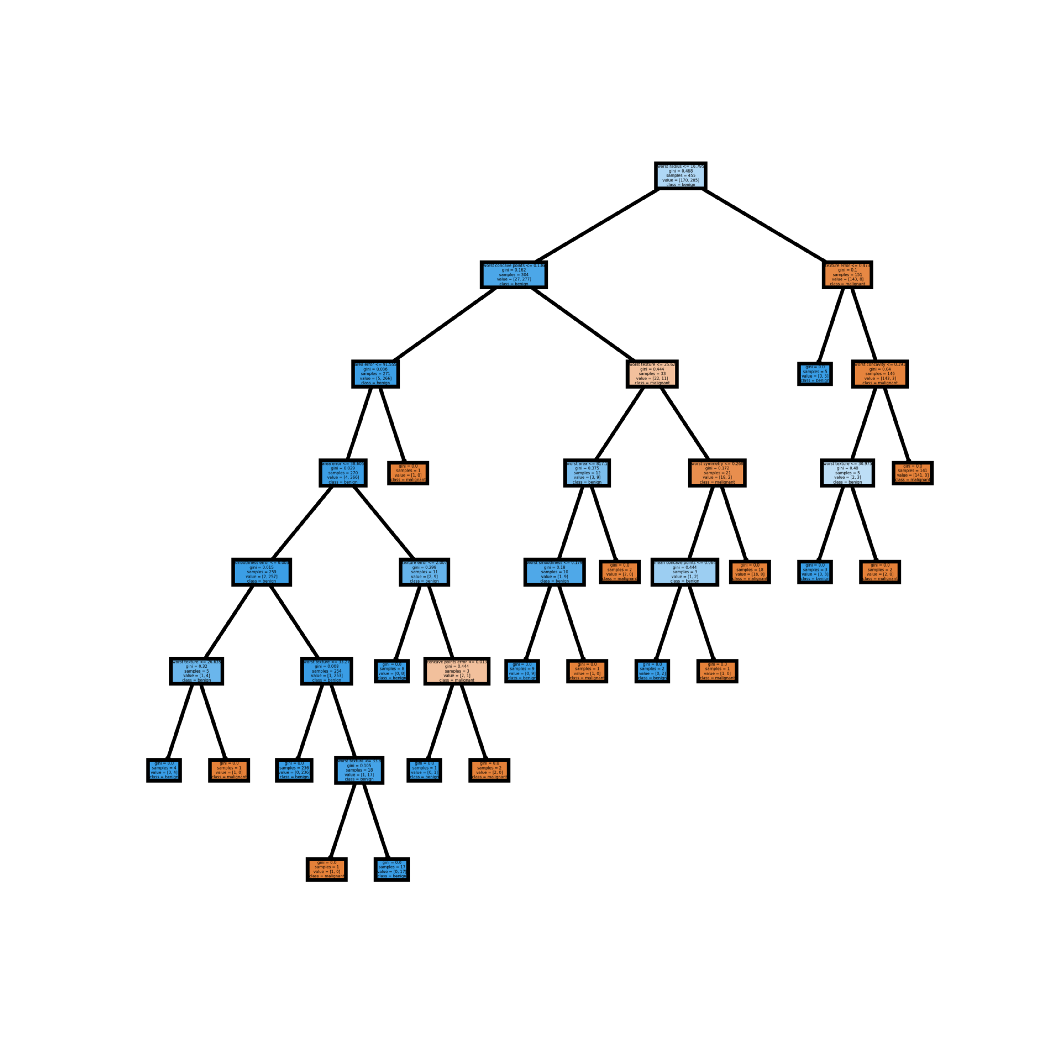
Thus, we use GridSearch CV to get the best choice of *a* which minimizes the Tree Score or Ra(T).

*Note: Which subtree is selected eventually depends on α . If****α=0****, then the biggest tree will be chosen because the complexity penalty term(i.e. a \* T ) is essentially dropped. As α approaches infinity, the tree of size 1, i.e., a single root node, will be selected.*

To get an idea of what values of ccp\_alpha will work to reduce the tree size, scikit-learn provides a functioncost\_complexity\_pruning\_path that returns the effective alphas and the corresponding total leaf impurities at each step of the pruning process.

Let’s build our final tree model and see how it performs.

**Original Tree:** It has 18 rules and a accuracy of 91%.



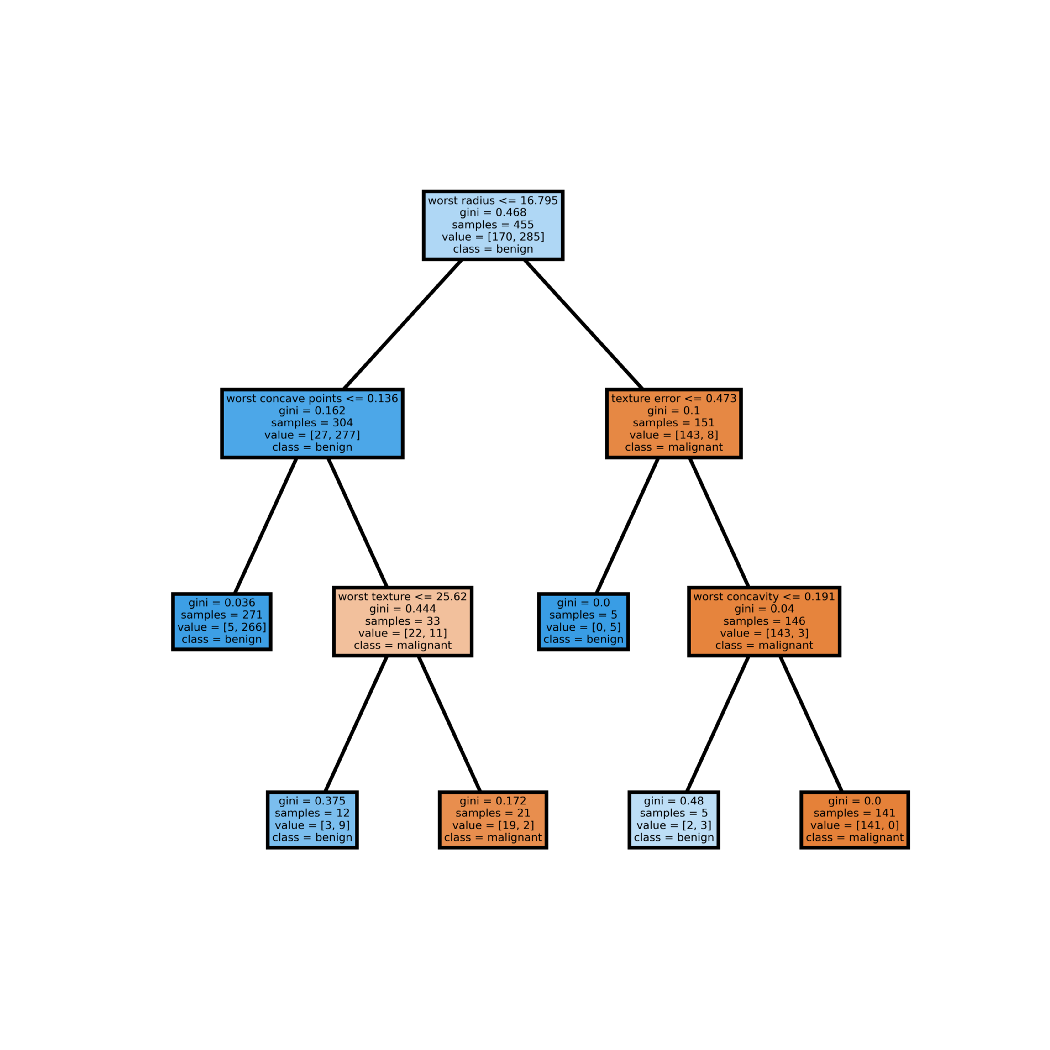


STEP 1&2 above: Each ccp\_alpha above represents an optimal subtree.

STEP 3&4 above: Once again, we build a grid search to compare the various trees. Here the grid search is comparing the training accuracy for each optimal subtree.

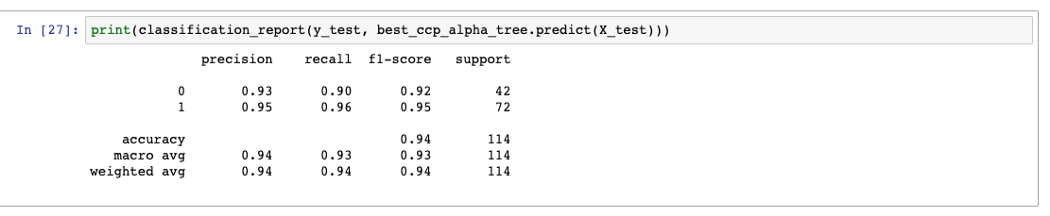
We see that the most accurate subtree is that generated by the ccp\_alpha, 0.0059340658…

Lets look at the Pruned tree structure.



This model only contains 5 rules, whereas original Tree contained 18 rules. Also, The pruned model is less complex, easier to explain and easier to understand than the previous decision tree plots.

Also, lets check the accuracy



The accuracy has also improved to 94% that also with much reduced Rules.T**his reduced complexity while giving better accuracy.**