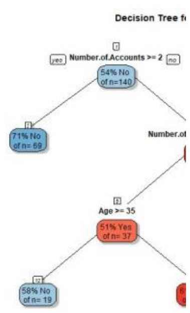
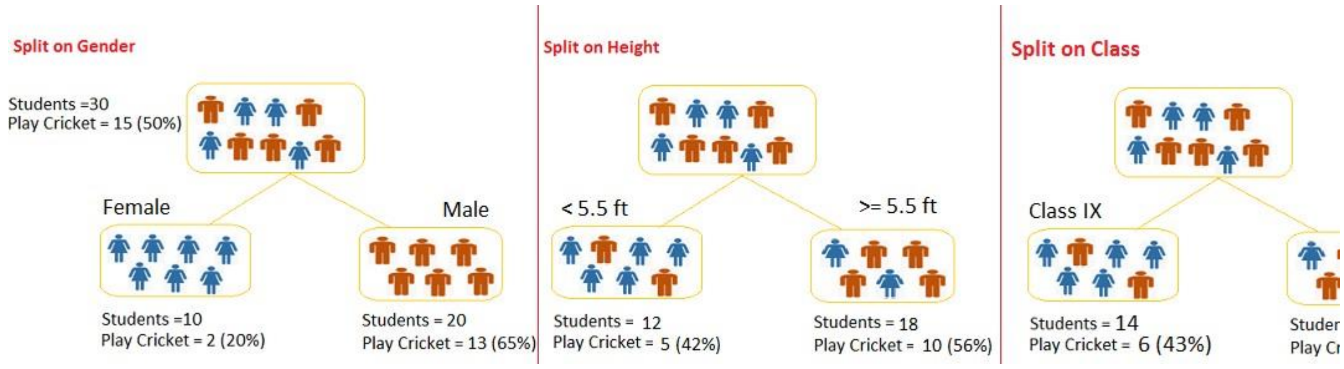
<

What is a Decision Tree? How does it work?

Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.



Example:
Let's say we have a sample of 30 students with three variables Gender (Boy/ Girl), Class(IX/ X) and Height (5 to 6 ft). 15 out of these 30 play cricket in leisure time. Now, I want to create a model to predict who will play cricket during leisure period? In this problem, we need to segregate students who play cricket in their leisure time based on highly significant input variable among all three.
This is where decision tree helps, it will segregate the students based on all values of three variable and identify the variable, which creates the best homogeneous sets of students (which are heterogeneous to each other). In the snapshot below, you can see that variable Gender is able to identify best homogeneous sets compared to the other two variables.



As mentioned above, decision tree identifies the most significant variable and it's value that gives best homogeneous sets of population. Now the qu how does it identify the variable and the split? To do this, decision tree uses various algorithms, which we will shall discuss in the following section.

Types of Decision Trees

Types of decision tree is based on the type of target variable we have. It can be of two types:

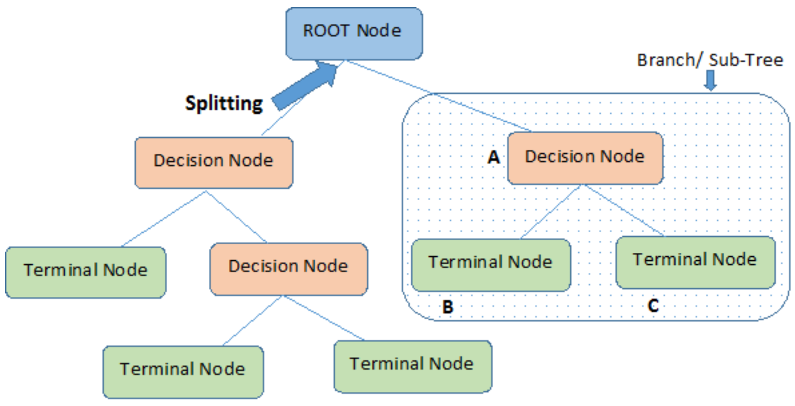
- **Categorical Variable Decision Tree:** Decision Tree which has categorical target variable then it called as categorical variable decision tre scenario of student problem, where the target variable was "Student will play cricket or not" i.e. YES or NO.
- **Continuous Variable Decision Tree:** Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

Example: Let's say we have a problem to predict whether a customer will pay his renewal premium with an insurance company (yes/ no). Here w customer is a significant variable but insurance company does not have income details for all customers. Now, as we know this is an important variabl decision tree to predict customer income based on occupation, product and various other variables. In this case, we are predicting values for continuou

Important Terminology related to Decision Trees

Let's look at the basic terminology used with Decision trees:

- **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
- **Leaf/Terminal Node:** Nodes do not split is called Leaf or Terminal node.



Note:- A is parent node of B and C.

Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.

- **Branch / Sub-Tree:** A sub section of entire tree is called branch or sub-tree.

- **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent

These are the terms commonly used for decision trees. As we know that every algorithm has advantages and disadvantages, below are the impor should know.

Advantages

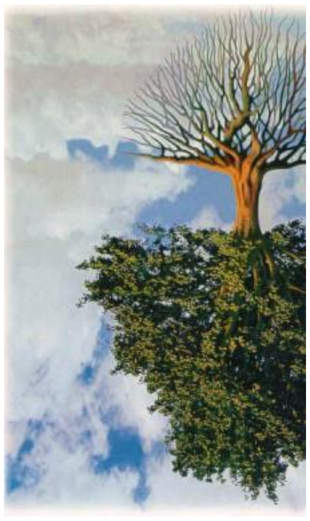
- **Easy to Understand**: Decision tree output is very easy to understand even for people from non-analytical background. It does not require any s read and interpret them. Its graphical representation is very intuitive and users can easily relate their hypothesis.
- **Useful in Data exploration:** Decision tree is one of the fastest way to identify most significant variables and relation between two or more vari decision trees, we can create new variables / features that has better power to predict target variable. You can refer article (Trick to enhance pow (https://www.analyticsvidhya.com/blog/2013/10/trick-enhance-power-regression-model-2/)) for one such trick. It can also be used in data example, we are working on a problem where we have information available in hundreds of variables, there decision tree will help to identify mos
- **Less data cleaning required:** It requires less data cleaning compared to some other modeling techniques. It is not influenced by outliers and degree.
- **Data type is not a constraint:** It can handle both numerical and categorical variables.
- **Non Parametric Method:** Decision tree is considered to be a non-parametric method. This means that decision trees have no assumpt distribution and the classifier structure.

Disadvantages

- **Over fitting:** Over fitting is one of the most practical difficulty for decision tree models. This problem gets solved by setting constraints on model pruning (discussed in detailed below).
- **Not fit for continuous variables**: While working with continuous numerical variables, decision tree looses information when it categorizes variab categories.

Regression Trees vs Classification Trees

We all know that the terminal nodes (or leaves) lies at the bottom of the decision tree. This means that decision trees are typically drawn upside dow the the bottom & roots are the tops (shown below).



Both the trees work almost similar to each other, let's look at the primary differences & similarity between classification and regression trees:

- Regression trees are used when dependent variable is continuous. Classification trees are used when dependent variable is categorical.
- In case of regression tree, the value obtained by terminal nodes in the training data is the mean response of observation falling in that region. Thus, if an unseen data observation falls in that region, we'll make its prediction with mean value.
- In case of classification tree, the value (class) obtained by terminal node in the training data is the mode of observations falling in that region. Thus, if an unseen data observation falls in that region, we'll make its prediction with mode value.
- Both the trees divide the predictor space (independent variables) into distinct and non-overlapping regions. For the sake of simplicity, you can think of these regions as high dimensional boxes or boxes.
- Both the trees follow a top-down greedy approach known as recursive binary splitting. We call it as 'top-down' because it begins from the top of tree when all the observations are available in a single region and successively splits the predictor space into two new branches down the tree. It is known as 'greedy' because, the algorithm cares (looks for best variable available) about only the current split, and not about future splits which will lead to a better tree.
- This splitting process is continued until a user defined stopping criteria is reached. For example: we can tell the the algorithm to stop once the number of observations per node becomes less than 50.
- In both the cases, the splitting process results in fully grown trees until the stopping criteria is reached. But, the fully grown tree is likely to overfit data, leading to poor accuracy on unseen data. This bring 'pruning'. Pruning is one of the technique used tackle overfitting. We'll learn more about it in following section.

How does a tree decide where to split?

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria is different for classification and regression trees.
Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes. The creation of sub-nodes increases the homogeneity of r other words, we can say that purity of the node increases with respect to the target variable. Decision tree splits the nodes on all available variables an which results in most homogeneous sub-nodes.
The algorithm selection is also based on type of target variables. Let's look at the four most commonly used algorithms in decision tree:

Gini Index

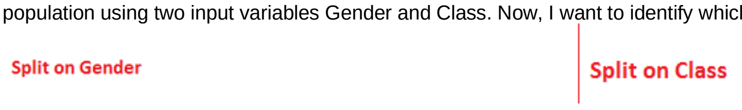
Gini index says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

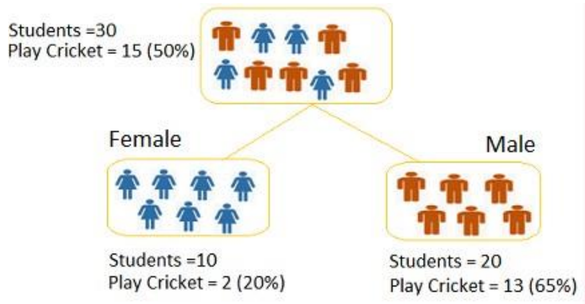
- It works with categorical target variable "Success" or "Failure".
- It performs only Binary splits
- Higher the value of Gini higher the homogeneity.
- CART (Classification and Regression Tree) uses Gini method to create binary splits.

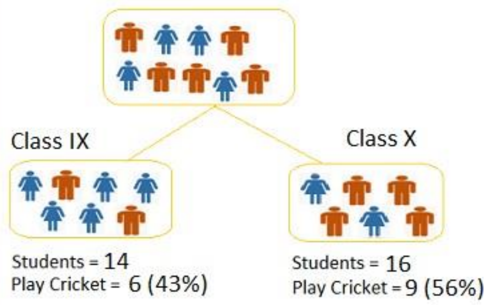
Steps to Calculate Gini for a split

- Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).
- Calculate Gini for split using weighted Gini score of each node of that split

Example: – Referring to example used above, where we want to segregate the students based on target variable (playing cricket or not). In the snaps population using two input variables Gender and Class. Now, I want to identify which split is producing more homogeneous sub-nodes using Gini index







Split on Gender:

- Calculate, Gini for sub-node Female = (0.2)*(0.2)+(0.8)*(0.8)=0.68
- Gini for sub-node Male = (0.65)*(0.65)+(0.35)*(0.35)=0.55
- Calculate weighted Gini for Split Gender = (10/30)*0.68+(20/30)*0.55 = **0.59**

Similar for Split on Class:

- Gini for sub-node Class IX = (0.43)*(0.43)+(0.57)*(0.57)=0.51
- Gini for sub-node Class X = (0.56)*(0.56)+(0.44)*(0.44)=0.51
- Calculate weighted Gini for Split Class = (14/30)*0.51+(16/30)*0.51 = **0.51**

Above, you can see that Gini score for *Split on Gender* is higher than *Split on Class,* hence, the node split will take place on Gender.

Chi-Square

It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node. We measure it b standardized differences between observed and expected frequencies of target variable.

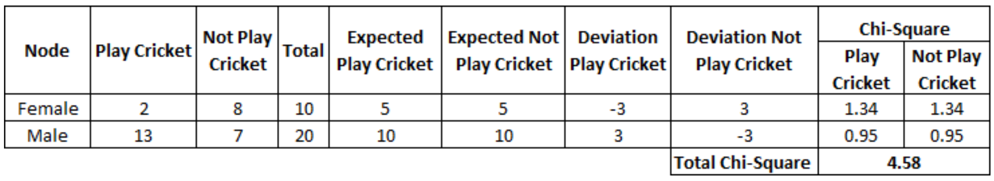
- It works with categorical target variable “Success” or “Failure”.
- It can perform two or more splits.
- Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.
- Chi-Square of each node is calculated using formula,
- Chi-square = ((Actual – Expected)^2 / Expected)^1/2
- It generates tree called CHAID (Chi-square Automatic Interaction Detector)

Steps to Calculate Chi-square for a split:

- Calculate Chi-square for individual node by calculating the deviation for Success and Failure both
- Calculated Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split

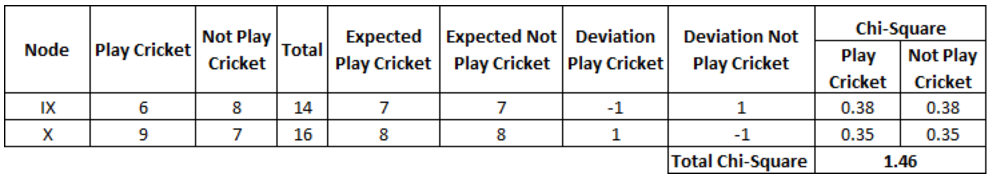
Example: Let’s work with above example that we have used to calculate Gini.
Split on Gender:

- First we are populating for node Female, Populate the actual value for “**Play Cricket”** and **“Not Play Cricket”**, here these are 2 and 8 respective
- Calculate expected value for “**Play Cricket”** and “**Not Play Cricket”**, here it would be 5 for both because parent node has probability of 50% same probability on Female count(10).
- Calculate deviations by using formula, Actual – Expected. It is for “**Play Cricket”** (2 – 5 = -3) and for “**Not play cricket”** (8 – 5 = 3).
- Calculate Chi-square of node for “**Play Cricket**” and “**Not Play Cricket**” using formula with formula, **= ((Actual – Expected)^2 / Expected)^1/** table for calculation.
- Follow similar steps for calculating Chi-square value for Male node.
- Now add all Chi-square values to calculate Chi-square for split Gender.



Node	Play Cricket	Not Play Cricket	Total	Expected Play Cricket	Expected Not Play Cricket	Deviation Play Cricket	Deviation Not Play Cricket	Chi-Square Play Cricket	Chi-Square Not Play Cricket
Female	2	8	10	5	5	-3	3	1.34	1.34
Male	13	7	20	10	10	3	-3	0.95	0.95
							Total Chi-Square	4.58	

Split on Class:
Perform similar steps of calculation for split on Class and you will come up with below table.

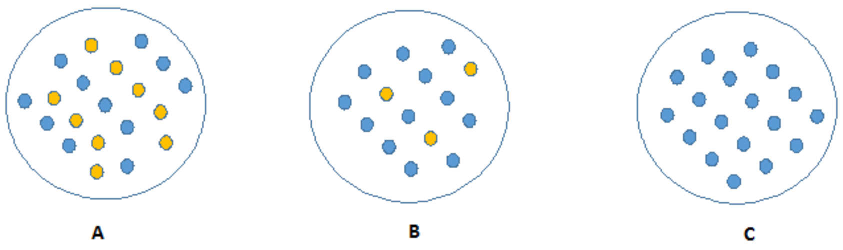


Node	Play Cricket	Not Play Cricket	Total	Expected Play Cricket	Expected Not Play Cricket	Deviation Play Cricket	Deviation Not Play Cricket	Chi-Square Play Cricket	Chi-Square Not Play Cricket
IX	6	8	14	7	7	-1	1	0.38	0.38
X	9	7	16	8	8	1	-1	0.35	0.35
							Total Chi-Square	1.46	

Above, you can see that Chi-square Gender split is more significant compare to Class.

Information Gain:

Look at the image below and think which node can be described easily. I am sure, your answer is C because it requires less information as all values a hand, B requires more information to describe it and A requires the maximum information. In other words, we can say that C is a Pure node, B is less impure.



Now, we can build a conclusion that less impure node requires less information to describe it. And, more impure node requires more information. I measure to define this degree of disorganization in a system known as Entropy. If the sample is completely homogeneous, then the entropy is zero a equally divided (50% – 50%), it has entropy of one.

Entropy can be calculated using formula:- $\text{Entropy} = -p \log_2 p - q \log_2 q$

Here p and q is probability of success and failure respectively in that node. Entropy is also used with categorical target variable. It chooses the split wh compared to parent node and other splits. The lesser the entropy, the better it is.

Steps to calculate entropy for a split:

- Calculate entropy of parent node
- Calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split.

Example: Let's use this method to identify best split for student example.

- Entropy for parent node = -(15/30) log2 (15/30) – (15/30) log2 (15/30) = **1**. Here 1 shows that it is a impure node.
- Entropy for Female node = -(2/10) log2 (2/10) – (8/10) log2 (8/10) = 0.72 and for male node, -(13/20) log2 (13/20) – (7/20) log2 (7/20) = **0.93**
- Entropy for split Gender = Weighted entropy of sub-nodes = (10/30)*0.72 + (20/30)*0.93 = **0.86**
- Entropy for Class IX node, -(6/14) log2 (6/14) – (8/14) log2 (8/14) = 0.99 and for Class X node, -(9/16) log2 (9/16) – (7/16) log2 (7/16) = 0.99.
- Entropy for split Class = (14/30)*0.99 + (16/30)*0.99 = **0.99**

Above, you can see that entropy for *Split on Gender* is the lowest among all, so the tree will split on *Gender*. We can derive information gain from entro

Reduction in Variance

Till now, we have discussed the algorithms for categorical target variable. Reduction in variance is an algorithm used for continuous target variables This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the populat

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

Above X-bar is mean of the values, X is actual and n is number of values.

Steps to calculate Variance:

- Calculate variance for each node.
- Calculate variance for each split as weighted average of each node variance.

Example: Let's assign numerical value 1 for play cricket and 0 for not playing cricket. Now follow the steps to identify the right split:

- Variance for Root node, here mean value is (15*1 + 15*0)/30 = 0.5 and we have 15 one and 15 zero. Now variance would be ((1-0.5)^2+(1-0.5)^2+(0-0.5)^2+….15 times) / 30, this can be written as (15*(1-0.5)^2+15*(0-0.5)^2) / 30 = **0.25**
- Mean of Female node = (2*1+8*0)/10=0.2 and Variance = (2*(1-0.2)^2+8*(0-0.2)^2) / 10 = 0.16
- Mean of Male Node = (13*1+7*0)/20=0.65 and Variance = (13*(1-0.65)^2+7*(0-0.65)^2) / 20 = 0.23
- Variance for Split Gender = Weighted Variance of Sub-nodes = (10/30)*0.16 + (20/30) *0.23 = **0.21**
- Mean of Class IX node = (6*1+8*0)/14=0.43 and Variance = (6*(1-0.43)^2+8*(0-0.43)^2) / 14= 0.24
- Mean of Class X node = (9*1+7*0)/16=0.56 and Variance = (9*(1-0.56)^2+7*(0-0.56)^2) / 16 = 0.25
- Variance for Split Gender = (14/30)*0.24 + (16/30) *0.25 = **0.25**

Above, you can see that Gender split has lower variance compare to parent node, so the split would take place on *Gender* variable.
Until here, we learnt about the basics of decision trees and the decision making process involved to choose the best splits in building a tree model. can be applied both on regression and classification problems. Let's understand these aspects in detail.

What are the key parameters of tree modeling and how can we avoid over-fitting in decision trees?

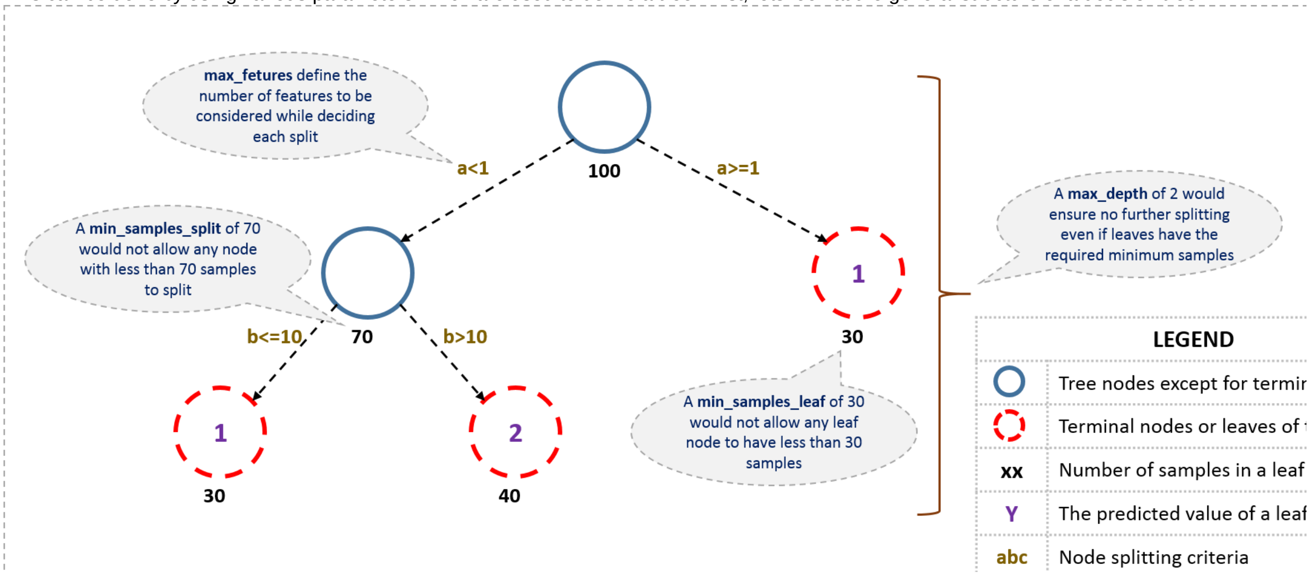
Overfitting is one of the key challenges faced while modeling decision trees. If there is no limit set of a decision tree, it will give you 100% accuracy on the worse case it will end up making 1 leaf for each observation. Thus, preventing overfitting is pivotal while modeling a decision tree and it can be don

- Setting constraints on tree size
- Tree pruning

Lets discuss both of these briefly.

Setting Constraints on Tree Size

This can be done by using various parameters which are used to define a tree. First, lets look at the general structure of a decision tree:

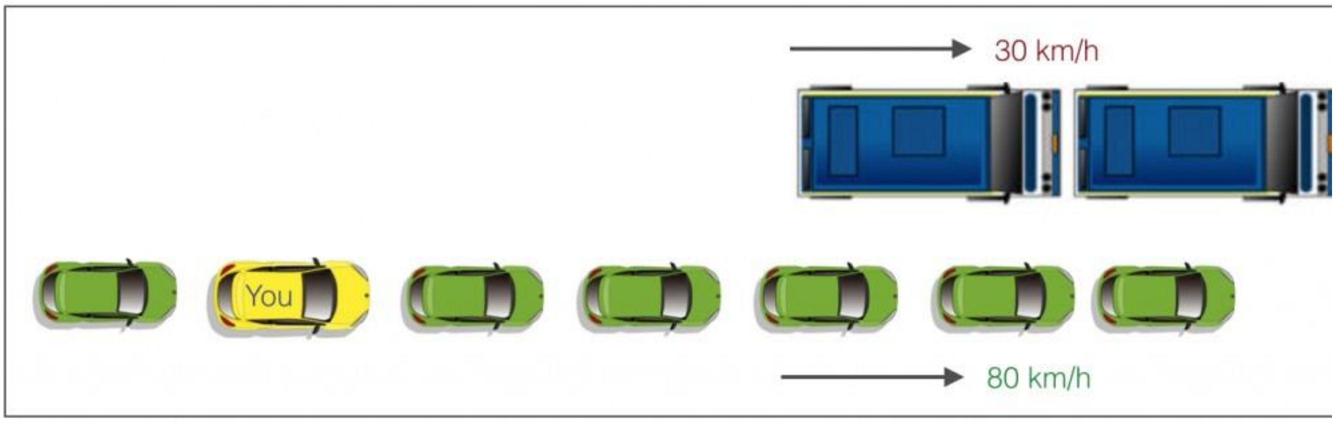

max_fetures define the number of features to be considered while deciding each split
a<1
100
a>=1
A min_samples_split of 70 would not allow any node with less than 70 samples to split
70
b<=10
b>10
1
30
2
40
1
30
A min_samples_leaf of 30 would not allow any leaf node to have less than 30 samples
A max_depth of 2 would ensure no further splitting even if leaves have the required minimum samples
LEGEND
Tree nodes except for termir
Terminal nodes or leaves of
xx Number of samples in a leaf
Y The predicted value of a leaf
abc Node splitting criteria

The parameters used for defining a tree are further explained below. The parameters described below are irrespective of tool. It is important to
parameters used in tree modeling. These parameters are available in R & Python.

- **Minimum samples for a node split**
  - Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.
  - Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample
  - Too high values can lead to under-fitting hence, it should be tuned using CV.
- **Minimum samples for a terminal node (leaf)**
  - Defines the minimum samples (or observations) required in a terminal node or leaf.
  - Used to control over-fitting similar to min_samples_split.
  - Generally lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in major
- **Maximum depth of tree (vertical depth)**
  - The maximum depth of a tree.
  - Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
  - Should be tuned using CV.
- **Maximum number of terminal nodes**
  - The maximum number of terminal nodes or leaves in a tree.
  - Can be defined in place of max_depth. Since binary trees are created, a depth of 'n' would produce a maximum of 2^n leaves.
- **Maximum features to consider for split**
  - The number of features to consider while searching for a best split. These will be randomly selected.
  - As a thumb-rule, square root of the total number of features works great but we should check upto 30-40% of the total number of feature
  - Higher values can lead to over-fitting but depends on case to case.

Tree Pruning

As discussed earlier, the technique of setting constraint is a greedy-approach. In other words, it will check for the best split instantaneously and move f
specified stopping condition is reached. Let's consider the following case when you're driving:



There are 2 lanes:

- A lane with cars moving at 80km/h
- A lane with trucks moving at 30km/h

At this instant, you are the yellow car and you have 2 choices:

- Take a left and overtake the other 2 cars quickly
- Keep moving in the present lane

Lets analyze these choice. In the former choice, you'll immediately overtake the car ahead and reach behind the truck and start moving at 30 km/h, loo
to move back right. All cars originally behind you move ahead in the meanwhile. This would be the optimum choice if your objective is to maximize t
next say 10 seconds. In the later choice, you sale through at same speed, cross trucks and then overtake maybe depending on situation ahead. Greed
This is exactly the difference between normal decision tree & pruning. A decision tree with constraints won't see the truck ahead and adopt a greedy
left. On the other hand if we use pruning, we in effect look at a few steps ahead and make a choice.
So we know pruning is better. But how to implement it in decision tree? The idea is simple.

- We first make the decision tree to a large depth.
- Then we start at the bottom and start removing leaves which are giving us negative returns when compared from the top.
- Suppose a split is giving us a gain of say -10 (loss of 10) and then the next split on that gives us a gain of 20. A simple decision tree will stop a
  we will see that the overall gain is +10 and keep both leaves.

Are tree based models better than linear models?

"If I can use logistic regression for classification problems and linear regression for regression problems, why is there a need to use trees"? Many of us
And, this is a valid one too.
Actually, you can use any algorithm. It is dependent on the type of problem you are solving. Let's look at some key factors which will help you to dec
use:

- If the relationship between dependent & independent variable is well approximated by a linear model, linear regression will outperform tree based
- If there is a high non-linearity & complex relationship between dependent & independent variables, a tree model will outperform a classical regres
- If you need to build a model which is easy to explain to people, a decision tree model will always do better than a linear model. Decision tree m
  to interpret than linear regression!