Business Analytics

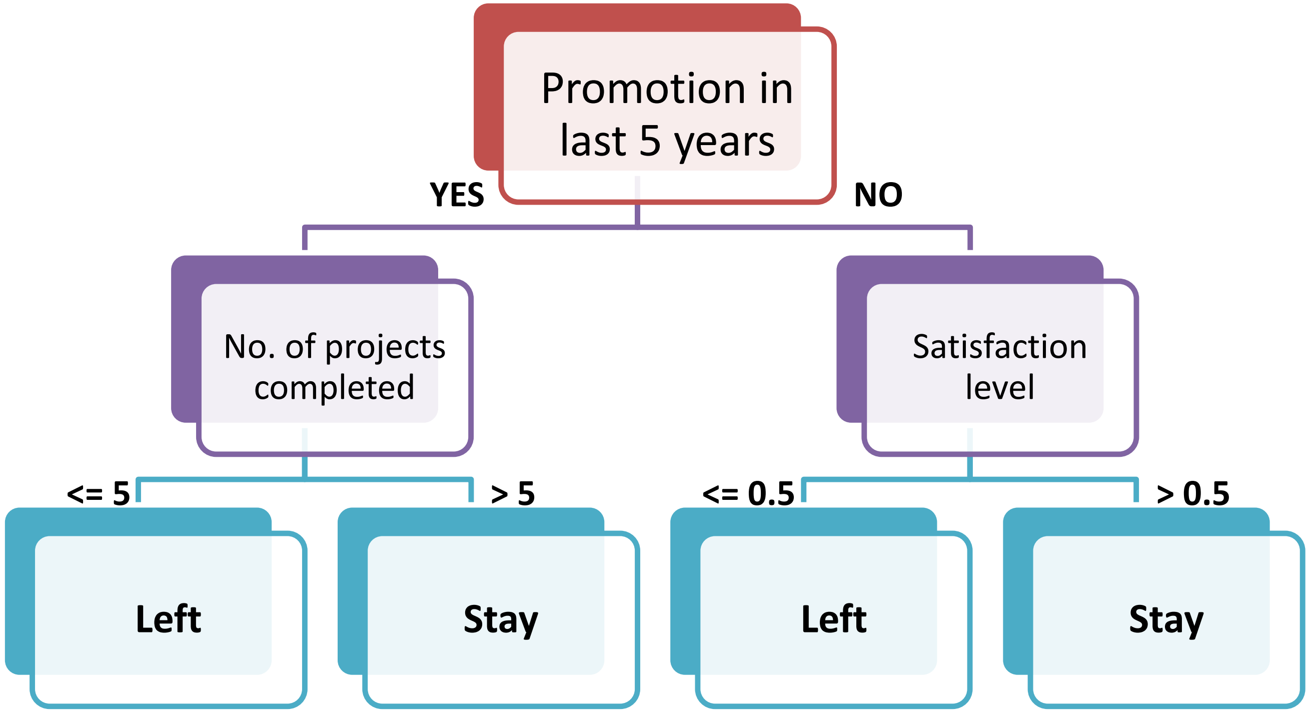
Data Classification

Proschool
An IMS Initiative

Caselet

The HR of a company wants to gauge who will stay and who will leave the company in next 1 year.

They have collected a sample data of the employees and they find out that:



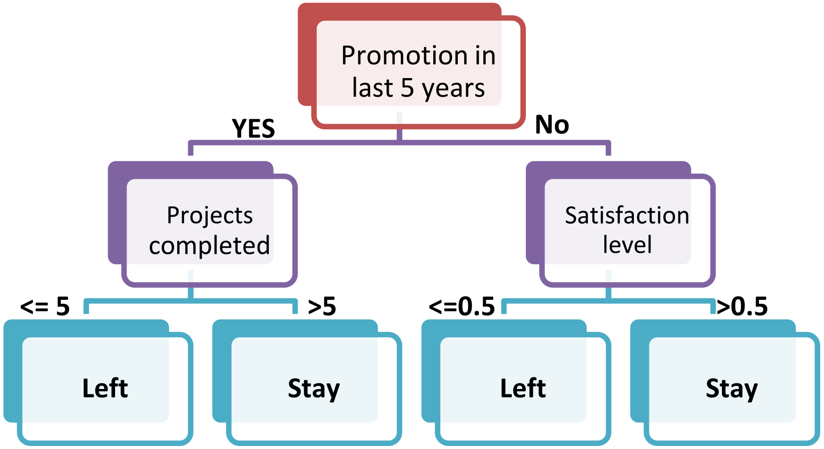
So here a set of rules get created for knowing Employees status

***R1**: If employee "Promotion in last 5 years" = YES and "Projects completed" > 5 then "**Stay**"*

***R2**: If employee "Promotion in last 5 years" = YES and "Projects completed" <= 5 then "**Left**"*

***R3**: If employee "Promotion in last 5 years" = NO and "Satisfaction level" > 0.5 then "**Stay**"*

***R4**: If employee "Promotion in last 5 years" = NO and "Satisfaction level" <= 0.5 then "**Left**"*



So, the employee who got promoted and completed more than 5 projects will stay in the company and similarly we can conclude the remaining. The HR will simply apply this set of rules to classify the remaining employees(not a part of this sample) to know if they will stay with the company or leave in the next 1 year.

- This graphical representation which classifies the data is called as the Decision Tree.

- The Decision tree is a type of Classification Algorithm, just like Logistic Regression.

- Logistic Regression is a go-to method for binary classification problems (problems with two class values).

- We can also use Logistic Regression to decide which employee will stay or leave using the predictor variables.

- Then why use Decision Trees over Logistic Regression?

Let’s understand Decision Tree and see some advantages of it…

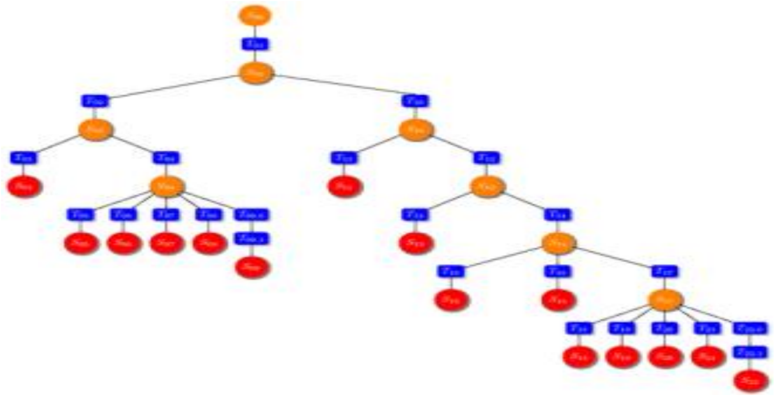
What is a Decision Tree? How does it work?

•Decision trees are a popular and powerful tool used for classification and prediction purposes.

• 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 differentiator in input variables.

•It is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems.



Advantages of Decision tree

Easy to Understand: It does not require any statistical knowledge to read and interpret them. Its graphical representation is very intuitive and users can easily relate their speculation.

Useful in Data exploration: Decision tree is one of the fastest way to identify most significant variables and relation between two or more variables. With the help of decision trees, we can create new variables / features that has better power to predict target variable. It can also be used in data exploration stage.

Less data cleaning required: It requires less data cleaning compared to some other modeling techniques. It is not influenced by outliers and missing values to a fair 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 assumptions about the space distribution and the classifier structure.

Are tree based models better than linear models?

“If we can use logistic regression for classification problems and linear regression for regression problems, why is there a need to use trees”? We can have this question. And, it is a valid one too.

Actually, we can use any algorithm. It is dependent on the type of problem we are solving. Let’s look at some key factors which will help us to decide which algorithm to use:

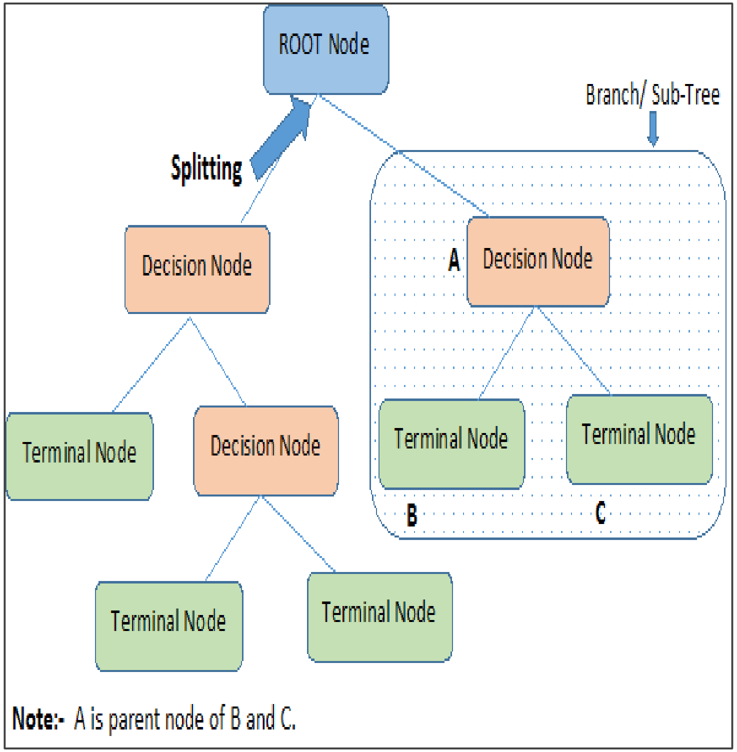
•If the relationship between dependent & independent variable is well approximated by a linear model, linear regression will outperform tree based model.

•If there is a high non-linearity & complex relationship between dependent & independent variables, a tree model will outperform a classical regression method.

•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 models are even simpler to interpret than linear regression!

Terminology of decision tree



Root Node: The starting point of the tree is the Root Node. It is also called as the Base node and contains the entire model population.

Splitting: The process of dividing a node into two smaller nodes is called splitting.

Decision Node: Any node, which is further split into smaller nodes is called a Decision node.

Leaf/ Terminal Node: The last nodes of the tree, i.e., the nodes that are not split any further are called the Terminal nodes. They are also denoted as Leaf nodes.

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

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 tree. Example:- In above scenario of HR problem, where the target variable was “Employee will stay or leave the company”.
It is also called as Classification Tree.

Continuous Variable Decision Tree: Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.
It is also called as Regression Tree.

Primary differences & similarity between classification and regression trees:

Classification Tree

1. Used when dependent variable is categorical.

2. 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.

Regression Tree

1. Used when dependent variable is continuous

2. 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.

•Both the trees follow a top-down greedy approach known as recursive binary splitting. We call it as 'top-down greedy' 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 and the algorithm considers 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.

Example

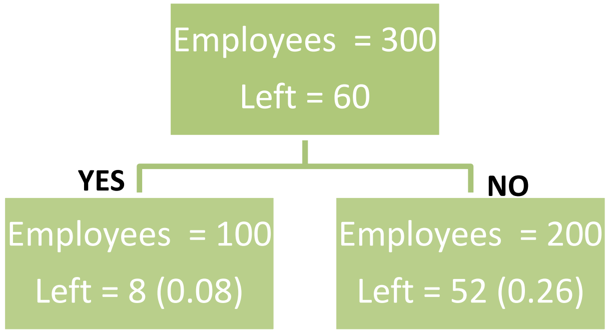
HR of a company has picked up a sample of 300 employees with three variables:
•Promotion in last 5 years (Yes/ No),
•Number of Projects completed (<=5/>5), and
•Satisfaction Level (<=0.5/>0.5).

60 out of these 300 employees, left the company.
Now, HR team will create a model to predict who will leave the company in next 1 year. In this problem, HR needs to segregate employees who will leave the company in next 1 year based on highly significant predictor variables among all three.

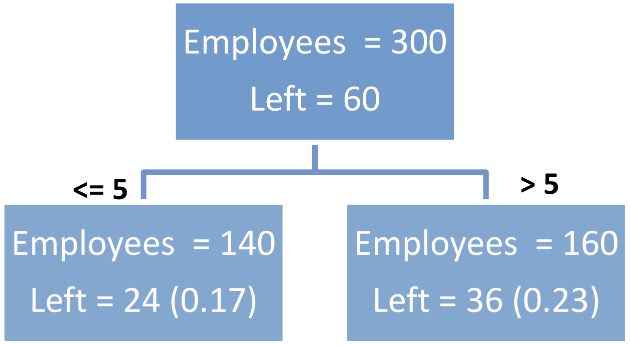
This is where decision tree will help, it will segregate the employees based on all values of three variable and identify the variable, which creates the best homogeneous sets of employees (which are heterogeneous to each other).

Decision tree will identify the most significant variable and it's value that gives best homogeneous sets of population. Now the question which arises is, how does it identify the variable and the split? To do this, decision tree uses various algorithms, which will be discussed in the following section.

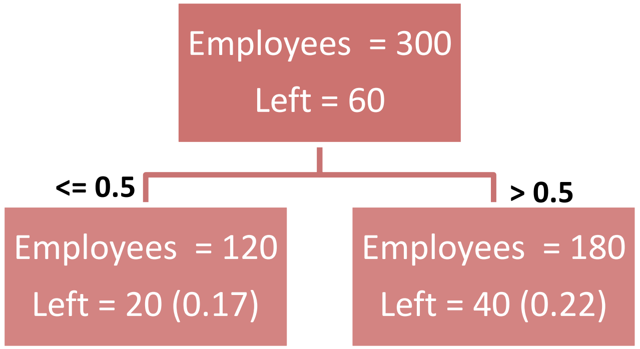
Split on 'Promotion in last 5 years'



Split on 'Projects completed'



Split on 'Satisfaction Level'



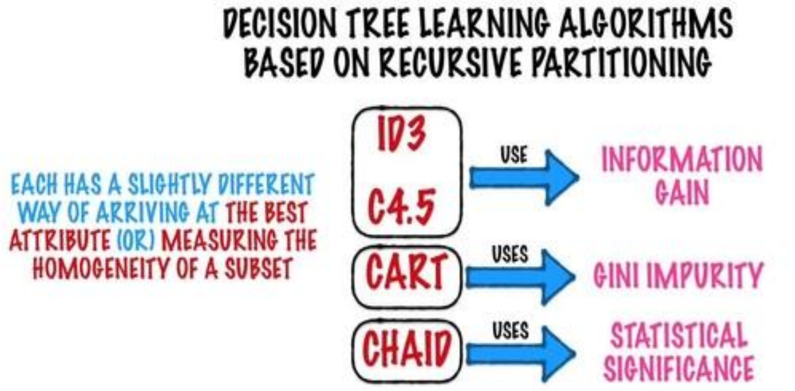
How does the tree decide where to split?

The primary challenge in the decision tree implementation is to identify which attributes need to be considered as the root node and each level.
Decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.
The decision of making strategic splits does affect a tree's accuracy immensely.

The decision criteria for attribute selection is different for classification and regression trees.

For Classification Tree
•GINI Index
•Chi Square (Statistical significance)
•Information Gain and Entropy

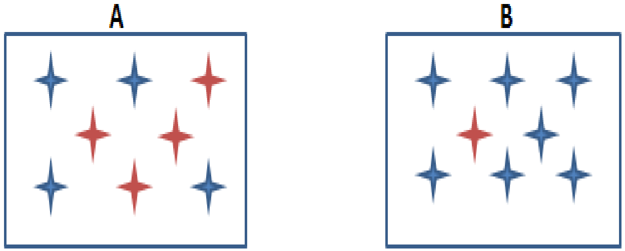
For Regression Tree
•Reduction in variance


DECISION TREE LEARNING ALGORITHMS BASED ON RECURSIVE PARTITIONING
EACH HAS A SLIGHTLY DIFFERENT WAY OF ARRIVING AT THE BEST ATTRIBUTE (OR) MEASURING THE HOMOGENEITY OF A SUBSET
ID3
C4.5
USE
INFORMATION GAIN
CART
USES
GINI IMPURITY
CHAID
USES
STATISTICAL SIGNIFICANCE

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".
•The impurity (or purity) measure used in building decision tree in CART is Gini Index. (Here, Node A is impure and Node B is pure)
•The decision tree built by CART algorithm is always a binary decision tree (each node will have only two child nodes)
•Higher the value of Gini higher the homogeneity



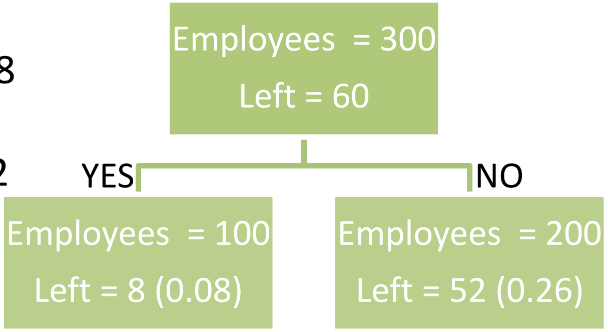
Steps to Calculate Gini for a split:
•Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).
•Calculate Gini for split using weighted Gini score of each node of that split

Actual working on GINI

- Split on **Promotion in last 5 years:**

Here, p = 8/100 = 0.08
Gini for sub-node "Yes" = (0.08)^2+(1-0.08)^2 = 0.8528
Here, p = 52/200 = 0.26
Gini for sub-node "No" = (0.26)^2+(1-0.26)^2 = 0.6152

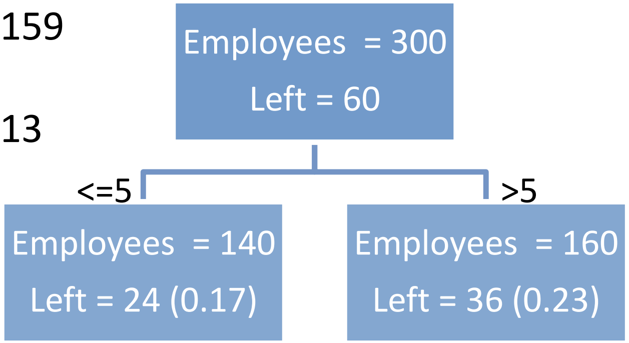
Weighted Gini for Promotion in last 5 years
= (100/300)* 0.8528 +(200/300)* 0.6152
= **0.6944**



- Split on **Projects completed:**

Here, p = 24/140 = 0.17
Gini for sub-node "<=5" = (0.17)^2+(1-0.17)^2 = 0.7159
Here, p = 36/160 = 0.23
Gini for sub-node ">5" = (0.23)^2+(1-0.23)^2 = 0.6513

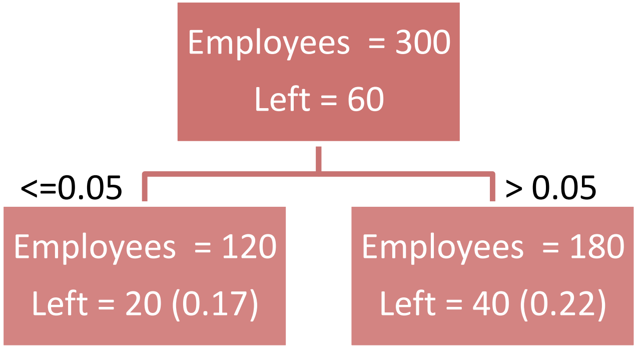
Weighted Gini for Promotion in last 5 years
= (140/300)* 0.7159 +(160/300)* 0.6513
= **0.6814**



Actual working on GINI

Split on **Satisfaction Level:**
Here, p = 20/120 = 0.17
Gini for sub-node “<=0.5” = (0.17)^2+(1-0.17)^2 = 0.7222
Here, p = 40/180 = 0.22
Gini for sub-node “>0.5” = (0.22)^2+(1-0.22)^2 = 0.6543

Weighted Gini for Promotion in last 5 years
= (120/300)* 0.7222 +(180/300)* 0.6543
= **0.6815**



We can see that Gini score for ***Promotion in last 5 years*** is higher than remaining variables, so the node split will take place on **Promotion in last 5 years**.

Chi Square

Chi square is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node. It is measured by sum of squares of 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.
- 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)
- Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.

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

Actual working on Chi Square

➢Split on '**Promotion in last 5 years**'

								Chi sqr	
Node	Left	Stay	Total	Exp Left	Exp Stay	Deviation Left	Deviation Stay	Left	Stay
Yes	8	92	100	50	50	-42	42	5.939697	5.939697
No	52	148	200	100	100	-48	48	4.8	4.8
							Total chi sqr	**21.47939392**	

➢Split on '**Projects completed**'

								Chi sqr	
Node	Left	Stay	Total	Exp Left	Exp Stay	Deviation Left	Deviation Stay	Left	Stay
<=5	24	116	140	70	70	-46	46	5.498052	5.498052
>5	36	124	160	80	80	-44	44	4.91935	4.91935
							Total chi sqr	**20.83480231**	

➢Split on '**Satisfaction Leve**l'

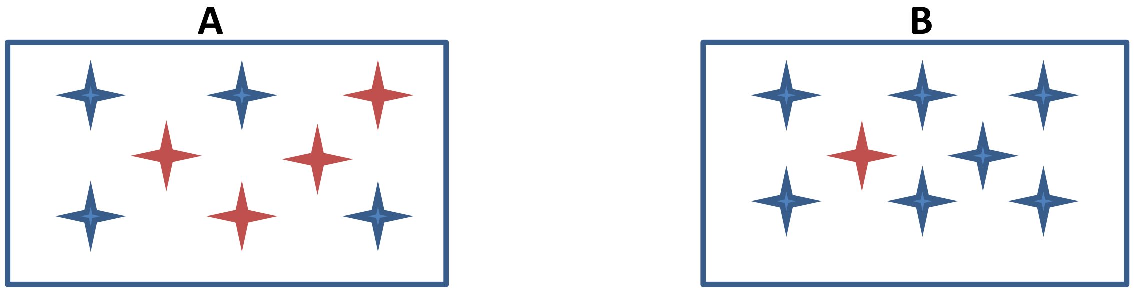
								Chi sqr	
Node	Left	Stay	Total	Exp Left	Exp Stay	Deviation Left	Deviation Stay	Left	Stay
<=0.5	20	100	120	60	60	-40	40	5.163978	5.163978
>0.5	40	140	180	90	90	-50	50	5.270463	5.270463
							Total chi sqr	**20.86888112**	

As observed, that Chi-square also identify the **Promotion in last 5 years** split is more significant compare to other variables as its chi square value is higher.

Information and Entropy

Information Gain

By using information gain as a criterion, we try to estimate the information contained by each attribute. Look at the image below,



These are 2 nodes A and B. We can easily explain node B as it requires less information because it has more of blue stars and can be classified as ‘Blue Star’ category, whereas in Node A it is equal mixture of both blue and red. Node B is called as a pure node/ homogenous node.

Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Entropy

To measure the randomness or uncertainty of a random variable X is defined by **Entropy**. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

Entropy is calculated as, Entropy = -p log2 p – q log2 q

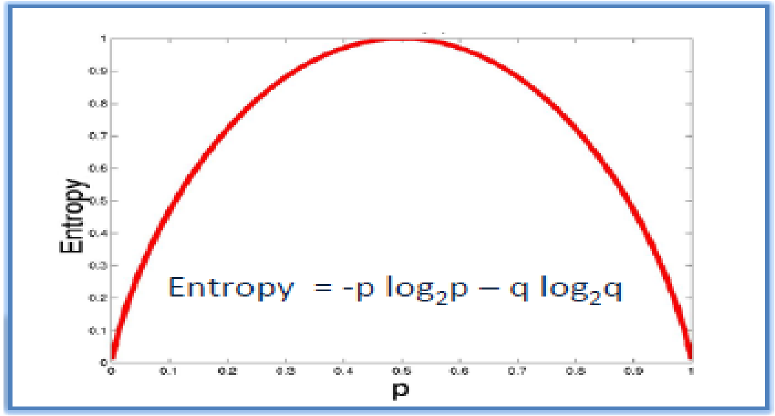
Here p and q is probability of success and failure respectively in that node.

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.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Actual working on Entropy

- Split on **Promotion in last 5 years:**

Entropy for 'YES' node = -(8/100) log2 (8/100) – (92/100) log2 (92/100) = 0.4022 and
For 'NO' node = -(52/200) log2 (52/200) – (148/200) log2 (148/200) = 0.8267

Weighted entropy of sub-nodes = (100/300)* 0.4022 + (200/300)* 0.8267 = **0.6852**

- Split on '**Projects completed**'

Entropy for '<= 5' node = -(24/140) log2 (24/140) – (116/140) log2 (116/140) = 0.6610 and
For '> 5' node = -(36/160) log2 (36/160) – (124/160) log2 (124/160) = 0.7692

Weighted entropy of sub-nodes = (140/300)* 0.6610 + (160/300)* 0.7692 = **0.7187**

Actual working on Entropy

➢Split on '**Satisfaction Leve**l'

Entropy for '<= 0.5' node = -(20/120) log2 (20/120) – (100/120) log2 (100/120) = 0.6500 and
For '> 0.5' node = -(40/180) log2 (40/180) – (140/180) log2 (140/180) = 0.7642

Weighted entropy of sub-nodes = (120/300)* 0.6500 + (180/300)* 0.7642
= **0.7185**

Above, we see that entropy for *Split on* ***Promotion in last 5 years*** is the lowest among all, so the tree will split on ***Promotion in last 5 years***.

We can derive information gain from entropy as **1- Entropy.**
Information Gain for Promotion in last 5 years = 1 – 0.6852 = 0.3148

Reduction in Variance

GINI, Chi Square and Entropy are the measures to decide the most significant variable to split first in classification trees.
But in regression trees, the measure used is Reduction in Variance.

Steps to calculate Variance:
•Calculate variance for each node.
•Calculate variance for each split as weighted average of each node variance.

➢‘Promotion in last 5 years’
Lets assign numerical value 1 for Left and 0 for Stay.
Mean of ‘Yes’ node = (8*1+92*0)/100 = 0.08 and
Variance = (8*(1-0.08)^2+92*(0-0.08)^2)/100 = 0.0736

Mean of ‘No’ Node = (52*1+148*0)/200 = 0.26 and
Variance =(52*(1-0.26)^2+148*(0-0.26)^2)/200 = 0.1924

Variance for Split ‘Promotion in last 5 years’ = Weighted Variance of Sub-nodes
= (100/300)*0.0736 + (200/300)*0.1924= **0.1528**

Similarly, we will calculate the variance for split at remaining other variables and compare.

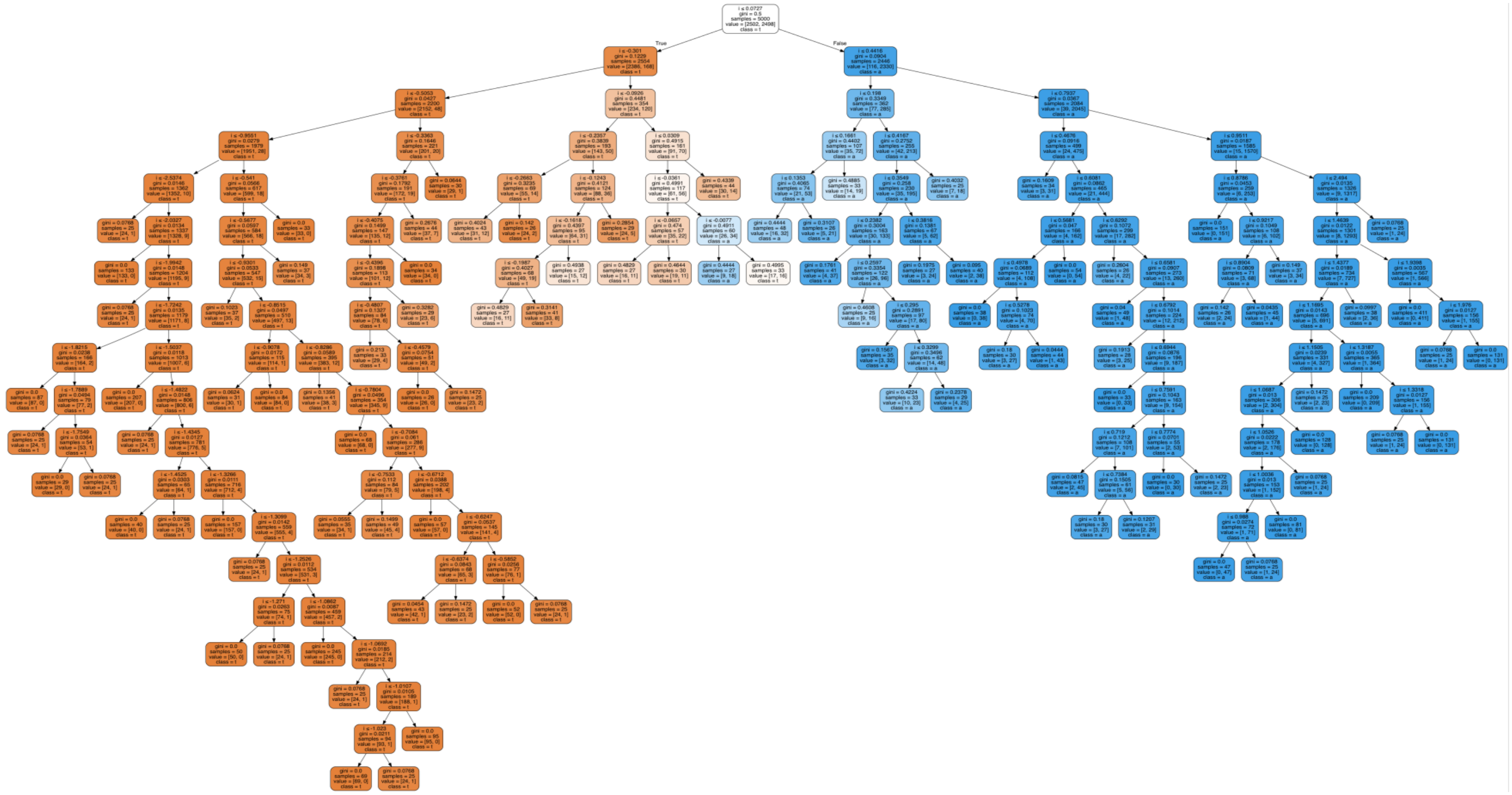
- Variance for Split ‘Projects completed’ = 0.1593

- Variance for Split ‘Satisfaction level’ = 0.1593

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 population.

Above, we can see that ‘Promotion in last 5 years’ split has lower variance compared to the other splits, so the split would first take place on ’Promotion in last 5 years’ variable.

Decision Tree - Overfitting



As we can see this is a over-fit tree which is perfectly classifying the data.

Decision Tree – Over-fitting

Over-fitting is a significant practical difficulty for decision tree models and many other predictive models. If there is no limit set of a decision tree, it will give you 100% accuracy on training set because in the worse case it will end up making 1 leaf for each observation as we see in the earlier diagram.

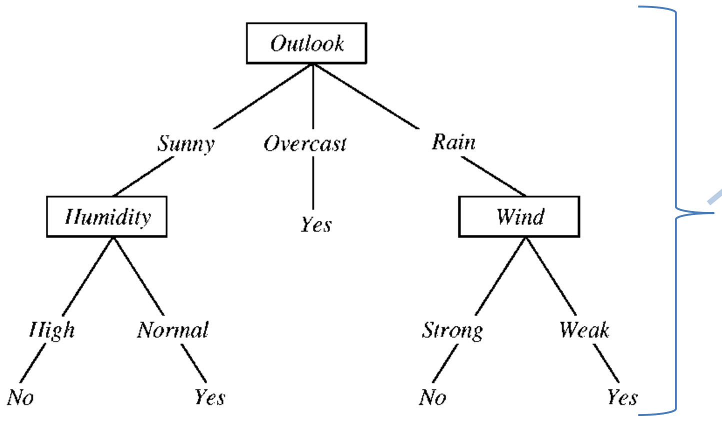
There are several approaches to avoiding over-fitting in building decision trees.
•Pre-pruning that stop growing the tree earlier, before it perfectly classifies the training set.
•Post-pruning that allows the tree to perfectly classify the training set, and then post prune the tree.

We will discuss both the approaches in details.

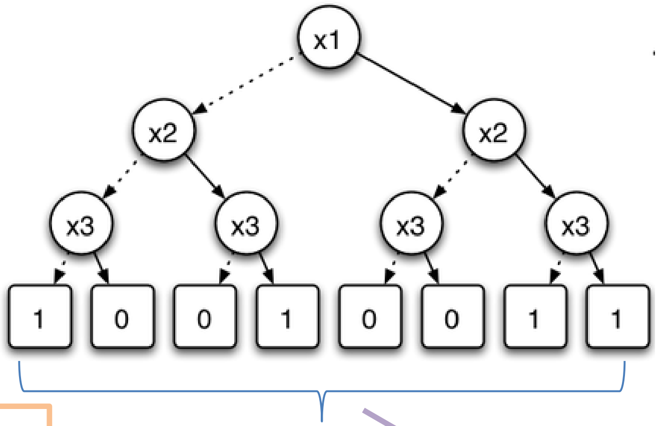
Setting constraints on Tree size : Pre pruning

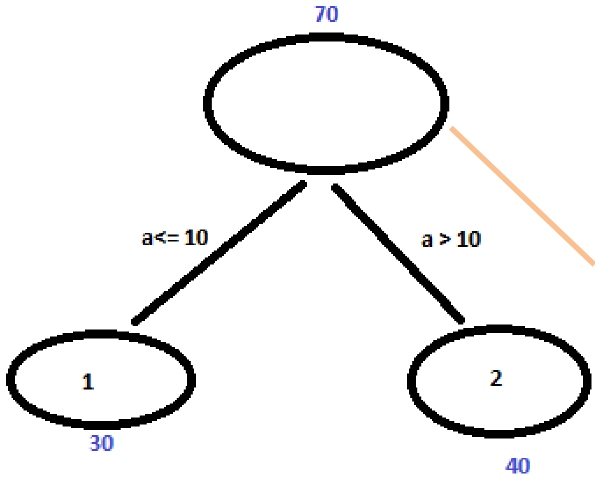
Minimum samples for a node split
Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.
Minimum samples for a terminal node (leaf)
Defines the minimum samples (or observations) required in a terminal node or leaf.
Maximum depth of tree (vertical depth)
The maximum depth of a tree can be specified to control over-fitting, as higher depth will allow model to learn relations very specific to a particular sample.
Maximum number of terminal nodes
The maximum number of terminal nodes or leaves in a tree.
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 features.

Pre pruning: Diagrammatic representation



The **max_depth** ensures non over-fitting of the tree. We can mention how deep a tree can grow .





The **min_split** of 70 would not allow any node with less than 70 samples to split further.

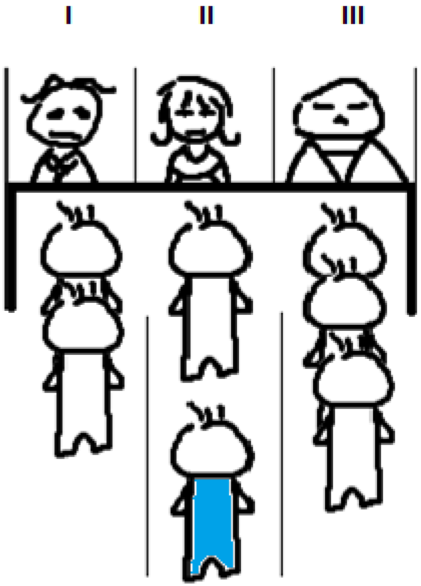
Deciding the maximum no.of terminal nodes

Pruning

Post-pruning is the method that allows the tree to perfectly classify the training set, and then post prune the tree.

Lets consider an example:

Here, we can see 3 counters and 3 queues. All the 3 counters have same operational time.
In queue I and III, there are 2 and 3 persons
Standing respectively, and in queue II, there is only 1 person standing.



If you are to enter the queue, the obvious choice would be II, since there is only one person standing.

But you figure out by enquiring that counter II, will close by the time its your turn. Hence, in this case you would optimally make a choice to enter in counter I.

This is exactly the difference between normal decision tree & pruning.
A decision tree with constraints won’t see the closing time ahead and adopt a greedy approach by taking counter II . On the other hand if we use pruning, we in effect look at a few steps ahead and make a choice.

Practically, the second approach of post-pruning over-fit trees is more successful because it is not easy to precisely estimate when to stop growing the tree.

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 -20 (loss of 20) and then the next split on that gives us a gain of 30. A simple decision tree will stop at step 1 but in pruning, we will see that the overall gain is +10 and keep both leaves.

We will learn it in detail with the case study.

Basics of Ensemble models

Let’s understand with an example to understand the basics of Ensemble learning

Suppose you want to buy a *XYZ* Car. You aren’t sure about its performance though. So, you will look for advice on whether it meets all your expectations? You decide to approach various experts having diverse domain experience:

1. Sales person of Company XYZ: This person knows the internal functionality of the car and have the insider information about the functionality and performance of the car. But he lacks a broader perspective on how are competitors innovating, how is the technology evolving and what will be the impact of this evolution on XYZ car. **In the past, he has been right 80% times**.

2. Online Auto review: They have a broader perspective on how the performance of cay will fair of in the competitive environment. However, it lacks a view on how the company’s internal functionality is faring. **In the past, it has been right 85% times**.

3. Friend who has the same car: This person has observed the car's performance over past 8 months. He is also a knowledgeable in this domain, due to his love for cars. He also has developed a strong intuition on how car might perform over time. **In the past, he has been right 80% times**.

Given the broad spectrum of access we have, we can probably combine all the information and make an informed decision.

Ensemble is the art of combining diverse set of learners (individual models) together to improvise on the stability and predictive power of the model.
In the above example, the way we combine all the predictions together will be termed as Ensemble Learning.

Why we need Ensemble models

•Decision trees are prone to over fitting, especially when a tree is particularly deep. One way to combat this issue is by setting a max depth. This will limit our risk of over fitting; but at the expense of error due to bias.

•Thus if we set a max depth to a lower number, it would be simpler model with less variance but ultimately will not be a strong predictive model.

•Ideally, we would like to minimize both error due to bias and error due to variance.

Here comes the role of ensemble models.

Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to **decrease variance**, **bias**, or **improve predictions**.

Lets us understand the terms Bias and Variance.

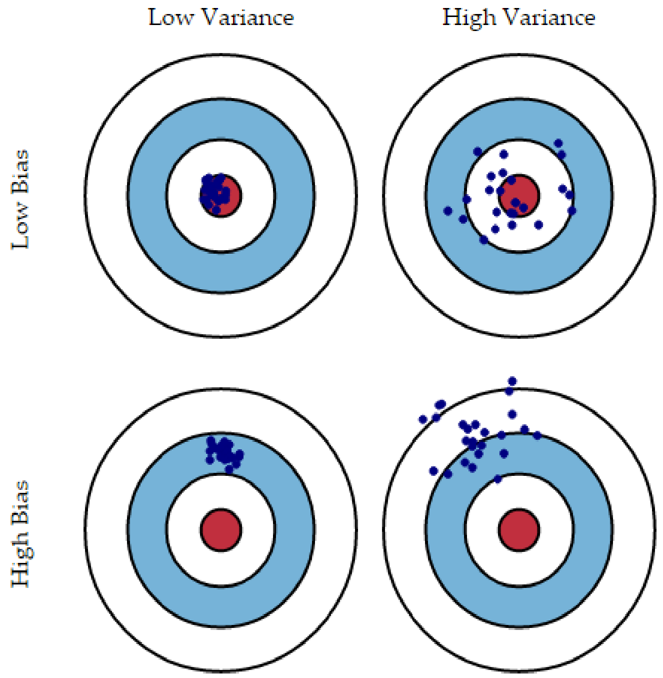
Introduction to Ensemble models

The error emerging from any model can be broken down into three components mathematically.

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

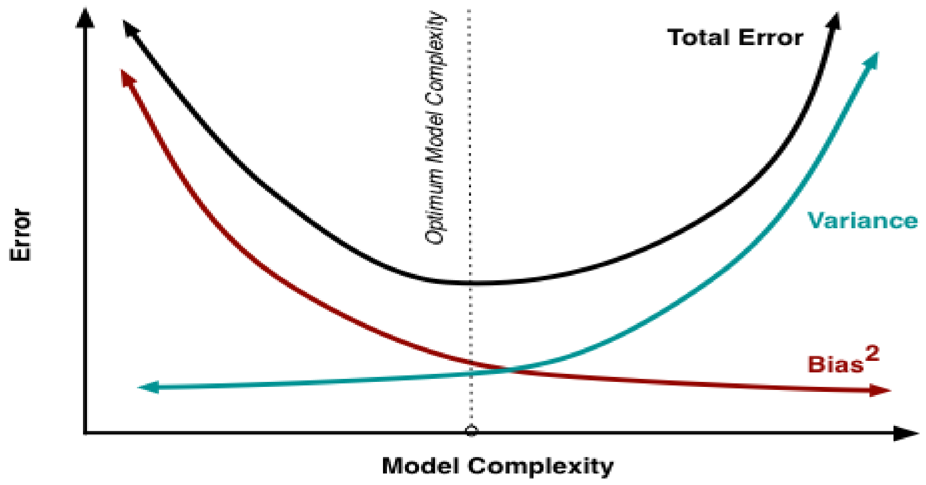
Bias error: is useful to quantify how much on an average are the predicted values different from the actual value. A high bias error means we have a under-performing model which keeps on missing important trends. **Variance** : quantifies how are the prediction made on same observation different from each other. A high variance model will over-fit on your training population and perform badly on any observation beyond training.

In diagram assume that red spot is the real value and blue dots are predictions.



Normally, as the complexity of the model increases, you will see a reduction in error due to lower bias in the model. However, this only happens till a particular point.
As we continue to make the model more complex, we end up over-fitting the model and hence the model will start suffering from high variance.

A champion model should maintain a balance between these two types of errors. This is known as the **trade-off management** of bias-variance errors. Ensemble learning is one way to execute this trade off analysis.



Thank You