# Decision Tree in Machine Learning

A decision tree is a flowchart-like structure in which  test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules. Below diagram illustrate the basic flow of decision tree for decision making with labels (Rain(Yes), No Rain(No)).

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning.

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a [non-parametric](https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/) **supervised learning** method used for both **classification**and **regression**tasks.

Tree models where the target variable can take a discrete set of values are called **classification trees**. Decision trees where the target variable can take continuous values (typically real numbers) are called **regression trees**. Classification And Regression Tree (CART) is general term for this.

Throughout this post i will try to explain using the examples.

## **Data Format**

Data comes in records of forms.

(x,Y)=(x1,x2,x3,....,xk,Y)

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector x is composed of the features, x1, x2, x3 etc., that are used for that task.

**Example**

training\_data = [

['Green', 3, 'Apple'],

['Yellow', 3, 'Apple'],

['Red', 1, 'Grape'],

['Red', 1, 'Grape'],

['Yellow', 3, 'Lemon'],

]

# Header = ["Color", "diameter", "Label"]

# The last column is the label.

# The first two columns are features.

# **Approach to make decision tree**

While making decision tree, at each node of tree we ask different type of questions. Based on the asked question we will calculate the information gain corresponding to it.

## **Information Gain**

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information. For each node of the tree, the information value **measures how much information** **a feature gives us about the class. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.**

Algorithm for constructing decision tree usually works top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrices for measuring best.

# **Gini Impurity**

First let’s understand the meaning of **Pure** and **Impure**.

## **Pure**

Pure means, in a selected sample of dataset all data belongs to same class (PURE).

## **Impure**

Impure means, data is mixture of different classes.

## **Definition of Gini Impurity**

Gini Impurity is a measurement of the likelihood of an incorrect classification of a new instance of a random variable, if that new instance were randomly classified according to the distribution of class labels from the data set.

If our dataset is Pure then likelihood of incorrect classification is 0. If our sample is mixture of different classes then likelihood of incorrect classification will be high.

# **Steps for Making decision tree**

* Get list of rows (dataset) which are taken into consideration for making decision tree (recursively at each nodes).
* Calculate uncertanity of our dataset or Gini impurity or how much our data is mixed up etc.
* Generate list of all question which needs to be asked at that node.
* Partition rows into True rows and False rows based on each question asked.
* Calculate information gain based on gini impurity and partition of data from previous step.
* Update highest information gain based on each question asked.
* Update best question based on information gain (higher information gain).
* Divide the node on best question. Repeat again from step 1 again until we get pure node (leaf nodes).

**Advantage of Decision Tree**

* Easy to use and understand.
* Can handle both categorical and numerical data.
* Resistant to outliers, hence require little data preprocessing.

**Disadvantage of Decision Tree**

* Prone to overfitting.
* Require some kind of measurement as to how well they are doing.
* Need to be careful with parameter tuning.
* Can create biased learned trees if some classes dominate.

## **How to avoid overfitting the Decision tree model**

Overfitting is one of the major problem for every model in machine learning. If model is overfitted it will poorly generalized to new samples. To avoid decision tree from overfitting **we remove the branches that make use of features having low importance.**This method is called as **Pruning or post-pruning.**This way we will reduce the complexity of tree, and hence imroves predictive accuracy by the reduction of overfitting.

Pruning should reduce the size of a learning tree without reducing predictive accuracy as measured by a [cross-validation](https://en.wikipedia.org/wiki/Cross-validation_(statistics)) set. There are 2 major Pruning techniques.

* *Minimum Error:*The tree is pruned back to the point where the cross-validated error is a minimum.
* *Smallest Tree:*The tree is pruned back slightly further than the minimum error. Technically the pruning creates a decision tree with cross-validation error within 1 standard error of the minimum error.

## **Early Stop or Pre-pruning**

An alternative method to prevent overfitting is to try and stop the tree-building process early, before it produces leaves with very small samples. This heuristic is known as *early stopping* but is also sometimes known as pre-pruning decision trees.

At each stage of splitting the tree, we check the cross-validation error. If the error does not decrease significantly enough then we stop. Early stopping may underfit by stopping too early. The current split may be of little benefit, but having made it, subsequent splits more significantly reduce the error.

Early stopping and pruning can be used together, separately, or not at all. Post pruning decision trees is more mathematically rigorous, finding a tree at least as good as early stopping. Early stopping is a quick fix heuristic. If used together with pruning, early stopping may save time. After all, why build a tree only to prune it back again?

## **Decision Tree in Real Life**

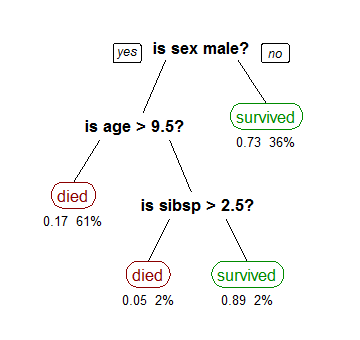
* **Selecting a flight to travel**

Suppose you need to select a flight for your next travel. How do we go about it? We check first if the flight is available on that day or not. If it is not available, we will look for some other date but if it is available then we look for may be the duration of the flight. If we want to have only direct flights then we look whether the price of that flight is in your pre-defined budget or not. If it is too expensive, we look at some other flights else we book it!

A tree has many analogies in real life, and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning, which will be the main focus of this article.

## **How can an algorithm be represented as a tree?**

For this let’s consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).

  
*A decision tree is drawn upside down with its root at the top.* In the image on the left, the bold text in black represents a condition/**internal node**, based on which the tree splits into branches/ **edges**. The end of the branch that doesn’t split anymore is the decision/**leaf**, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can’t ignore the simplicity of this algorithm. The **feature importance is clear** and relations can be viewed easily. This methodology is more commonly known as **learning decision tree from data** and above tree is called **Classification tree** as the target is to classify passenger as survived or died. **Regression trees** are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

**So, what is actually going on in the background?** Growing a tree involves deciding on **which features to choose** and **what conditions to use** for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, **you will need to trim it down** for it to look beautiful. Lets start with a common technique used for splitting.



*In this procedure all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected.*

Consider the earlier example of tree learned from titanic dataset. In the first split or the root, all attributes/features are considered and the training data is divided into groups based on this split. We have 3 features, so will have 3 candidate splits. Now we will ***calculate how much***[***accuracy***](https://medium.com/towards-data-science/balancing-bias-and-variance-to-control-errors-in-machine-learning-16ced95724db)***each split will cost us, using a function***. ***The split that costs least is chosen***, which in our example is sex of the passenger. This ***algorithm is recursive in nature*** as the groups formed can be sub-divided using same strategy. Due to this procedure, this algorithm is also known as the **greedy algorithm**, as we have an excessive desire of lowering the cost. **This makes the root node as best predictor/classifier.**

## **Cost of a split**

Lets take a closer look at **cost functions used for classification and regression**. In both cases the cost functions try to **find most homogeneous branches, or branches having groups with similar responses**. This makes sense we can be more sure that a test data input will follow a certain path.

Regression : sum(y — prediction)²

Lets say, we are predicting the price of houses. Now the decision tree will start splitting by considering each feature in training data. The mean of responses of the training data inputs of particular group is considered as prediction for that group. The above function is applied to all data points and cost is calculated for all candidate splits. *Again the split with lowest cost is chosen*. Another cost function involves reduction of standard deviation, more about it can be found [here](http://www.saedsayad.com/decision_tree_reg.htm).

Classification : G = sum(pk \* (1 — pk))

A Gini score gives an idea of how good a split is by how mixed the response classes are in the groups created by the split. Here, pk is proportion of same class inputs present in a particular group. A perfect class purity occurs when a group contains all inputs from the same class, in which case pk is either 1 or 0 and G = 0, where as a node having a 50–50 split of classes in a group has the worst purity, so for a binary classification it will have pk = 0.5 and G = 0.5.

## **When to stop splitting?**

You might ask ***when to stop growing a tree?*** As a problem usually has a large set of features, it results in large number of split, which in turn gives a huge tree. Such trees are *complex and can lead to overfitting.* So, we need to know when to stop? One way of doing this is to **set a minimum number of training inputs to use on each leaf.** For example we can use a minimum of 10 passengers to reach a decision(died or survived), and ignore any leaf that takes less than 10 passengers. Another way is to set **maximum depth** of your model. **Maximum depth refers to the the length of the longest path from a root to a leaf.**

## **Pruning**

The performance of a tree can be further increased by ***pruning***. *It involves****removing the branches that make use of features having low importance***. This way, we reduce the complexity of tree, and thus increasing its predictive power by reducing overfitting.

Pruning can start at either root or the leaves. The simplest method of pruning starts at leaves and removes each node with most popular class in that leaf, this change is kept if it doesn't deteriorate [accuracy](https://medium.com/towards-data-science/balancing-bias-and-variance-to-control-errors-in-machine-learning-16ced95724db). Its also called **reduced error pruning**. More sophisticated pruning methods can be used such as **cost complexity pruning** where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree. This is also known as **weakest link pruning.**

## **Advantages of CART**

* Simple to understand, interpret, visualize.
* Decision trees *implicitly perform variable screening or feature selection.*
* Can *handle both numerical and categorical data*. Can also *handle multi-output problems.*
* Decision trees require relatively *little effort from users for data preparation.*
* *Nonlinear relationships between parameters do not affect tree performance.*

## **Disadvantages of CART**

* Decision-tree learners *can create over-complex trees* that do not generalize the data well. This is called *overfitting*.
* Decision trees can be unstable because *small variations in the data might result in a completely different tree being generated.* This is called [*variance*](https://medium.com/towards-data-science/balancing-bias-and-variance-to-control-errors-in-machine-learning-16ced95724db), which needs to be *lowered by methods like* *bagging and*[***boosting***](https://towardsdatascience.com/boosting-the-accuracy-of-your-machine-learning-models-f878d6a2d185).
* Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.
* Decision tree learners create [*biased*](https://medium.com/towards-data-science/balancing-bias-and-variance-to-control-errors-in-machine-learning-16ced95724db) *trees if some classes dominate*. It is therefore recommended to balance the data set prior to fitting with the decision tree.

This is all the basic, to get you at par with decision tree learning. An improvement over decision tree learning is made using [technique of **boosting**](https://towardsdatascience.com/boosting-the-accuracy-of-your-machine-learning-models-f878d6a2d185). A popular library for implementing these algorithms is [**Scikit-Learn**](https://becominghuman.ai/implementing-decision-trees-using-scikit-learn-5057b27221ec). It has a wonderful api that can get your model up an running with **just a few lines of code in python**.

DECISION TREE:::::::::::::::::::::::::::::::::::::::::::

It’s time to begin the journey

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller subsets with increase in depth of tree. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor is called **root node**. Decision trees can handle both categorical and numerical data.

**Types of decision trees**

1. Categorical Variable Decision Tree: Decision Tree which has categorical target variable then it called as categorical variable decision tree.
2. Continuous Variable Decision Tree: Decision Tree which has continuous target variable then it is called as Continuous Variable Decision Tree.

# **Important Terminology related to Decision Trees**

Let’s look at the basic terminologies used with Decision trees:

1. **Root Node:**It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:**It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:**When a sub-node splits into further sub-nodes, then it is called decision node.
4. **Leaf/ Terminal Node:**Nodes with no children (no further split) is called Leaf or Terminal node.
5. **Pruning:**When we reduce the size of decision trees by removing nodes (opposite of Splitting), the process is called pruning.
6. **Branch / Sub-Tree:**A sub section of decision tree is called branch or sub-tree.
7. **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 node.

**Algorithm used in decision trees:**

1. ID3
2. Gini Index
3. Chi-Square
4. Reduction in Variance

## **ID3**

The core algorithm for building decision trees is called **ID3.**Developed by J. R. Quinlan, this algorithm employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses *Entropy* and *Information Gain* to construct a decision tree.

**Entropy :**A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogeneous). ID3 algorithm uses entropy to calculate the homogeneity of a sample.

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

b) Entropy using the frequency table of two attributes:

**Information Gain:**The information gain is based on the decrease in entropy after a data-set is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

*Step 1*: Calculate entropy of the target.

*Step 2*: The dataset is then split on the  proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

*Step 3*: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

*Step 4a*: A branch with entropy of 0 is a leaf node.

*Step 4b*: A branch with entropy more than 0 needs further splitting.

*Step 5*: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

# **Gini Index**

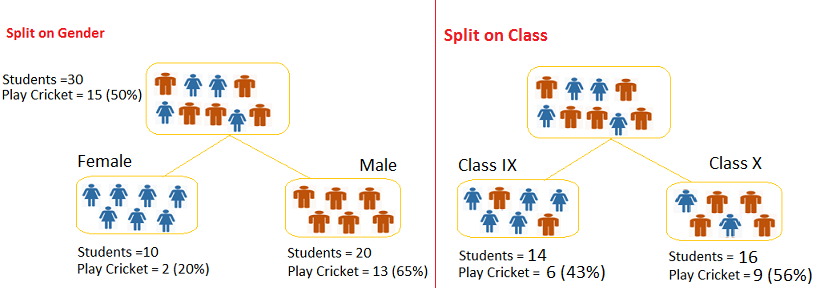
Gini index says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

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

**Steps to Calculate Gini for a split**

1. Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p²+q²).
2. Calculate Gini for split using weighted Gini score of each node of that split

**Example: —**Referring to example where we want to segregate the students based on target variable ( playing cricket or not ). In the snapshot below, we split the 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:**

1. Gini for sub-node Female = (0.2)\*(0.2)+(0.8)\*(0.8)=0.68
2. Gini for sub-node Male = (0.65)\*(0.65)+(0.35)\*(0.35)=0.55
3. Weighted Gini for Split Gender = (10/30)\*0.68+(20/30)\*0.55 = **0.59**

**Similar for Split on Class:**

1. Gini for sub-node Class IX = (0.43)\*(0.43)+(0.57)\*(0.57)=0.51
2. Gini for sub-node Class X = (0.56)\*(0.56)+(0.44)\*(0.44)=0.51
3. 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 by sum of squares of standardised differences between observed and expected frequencies of target variable.

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

**Steps to Calculate Chi-square for a split:**

1. Calculate Chi-square for individual node by calculating the deviation for Success and Failure both
2. 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:**

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

**Split on Class:**

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

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

# **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 (regression problems). 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 X-bar is mean of the values, X is actual and n is number of values.

**Steps to calculate Variance:**

1. Calculate variance for each node.
2. 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:

1. 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+….15 times+(0–0.5)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**
2. 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
3. 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
4. Variance for Split Gender = Weighted Variance of Sub-nodes = (10/30)\*0.16 + (20/30) \*0.23 = **0.21**
5. 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
6. 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
7. 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. As I said, decision tree can be applied both on regression and classification problems. Let’s understand these aspects in detail.

**C4.5 algoritham**

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy.

The training data is a set of already classified samples. Each sample consists of a p-dimensional vector , where the represent attribute values or features of the sample, as well as the class in which falls.

At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy).

The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

This algorithm has a few base cases.

All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.

None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.

Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

C4.5 made a number of improvements to ID3. Some of these are:

Handling both continuous and discrete attributes — In order to handle continuous attributes, C4.5 creates a threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it.

Handling training data with missing attribute values — C4.5 allows attribute values to be marked as ? for missing. Missing attribute values are simply not used in gain and entropy calculations. Handling attributes with differing costs.

Pruning trees after creation — C4.5 goes back through the tree once it’s been created and attempts to remove branches that do not help by replacing them with leaf nodes.

**Pruning:**

One of the questions that arises in a decision tree algorithm is the optimal size of the final tree. A tree that is too large risks overfitting the training data and poorly generalizing to new samples. A small tree might not capture important structural information about the sample space. However,

it is hard to tell when a tree algorithm should stop because it is impossible to tell if the addition of a single extra node will dramatically decrease error. This problem is known as the horizon effect.

A common strategy is to grow the tree until each node contains a small number of instances then use pruning to remove nodes that do not provide additional information.

Pruning should reduce the size of a learning tree without reducing predictive accuracy as measured by a cross-validation set. There are many techniques for tree pruning that differ in the measurement that is used to optimize performance.

**Reduced error pruning**

One of the simplest forms of pruning is reduced error pruning. Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected then the change is kept. While somewhat naive, reduced error pruning has the advantage of simplicity and speed.

**Cost complexity pruning**

Cost complexity pruning generates a series of trees T0 . . . Tm where T0 is the initial tree and Tm is the root alone. At step i the tree is created by removing a subtree from tree i-1and replacing it with a leaf node with value chosen as in the tree building algorithm. The subtree that is removed is chosen as follows. Define the error rate of tree T over data set S as err(T,S). The subtree that minimizes is chosen for removal. The function prune(T,t) defines the tree gotten by pruning the subtrees t from the tree T. Once the series of trees has been created, the best tree is chosen by generalized accuracy as measured by a training set or cross-validation.

**Advantages:**

Decision trees assist managers in evaluating upcoming choices. The tree creates a visual representation of all possible outcomes, rewards and follow-up decisions in one document. Each subsequent decision resulting from the original choice is also depicted on the tree, so you can see the overall effect of any one decision. As you go through the tree and make choices, you will see a specific path from one node to another and the impact a decision made now could have down the road.

1. Brainstorming Outcomes : Decision trees help you think of all possible outcomes for an upcoming choice. The consequences of each outcome must be fully explored, so no details are missed. Taking the time to brainstorm prevents overreactions to any one variable. The graphical depiction of various alternatives makes them easier to compare with each other. The decision tree also adds transparency to the process. An independent party can see exactly how a particular decision was made.
2. Decision Tree Versatility : Decision trees can be customized for a variety for situations. The logical form is good for programmers and engineers. Technicians can also use decision trees to diagnose mechanical failures in equipment or troubleshoot auto repairs. Decision trees are also helpful for evaluating business or investment alternatives. Managers can recreate the math used in a particular decision tree to analyzing the company’s decision-making process.
3. 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.
4. Data type is not a constraint: It can handle both numerical and categorical variables.
5. 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.