Decision Tree Tutorial

Decision Trees

I am sure you are using Decision Trees in your day to day life without knowing it. For example, you go to your nearest super store and want to buy milk for your family, the very first question which comes to your mind is – How much milk should I buy today?

To answer the basic question, your un-conscious mind makes some calculations (based on the sample questions listed below) and you end up buying the required quantity of milk. Is it a normal weekday?

- On weekdays we require 1 Litre of Milk
- Is it a weekend? On weekends we require 1.5 Litre of Milk
- Are we expecting any guests today? We need to buy 250 ML extra milk for each guest, etc.

Formally speaking, "Decision tree is a binary (mostly) structure where each node best splits the data to classify a response variable. Tree starts with a Root which is the first node and ends with the final nodes which are known as leaves of the tree".

Assume that you are given a characteristic information of 10,000 people living in your town. You are asked to study them and come up with the algorithm which should be able to tell whether a new person coming to the town is male or a female.

Primarily you are given information about:

- Skin colour
- Hair length
- Weight
- Height

Based on the information you can divide the information in such a way that it somehow indicates the characteristics of Males vs. Females.

Below is a hypothetical tree designed out of this data:

The tree shown above divides the data in such a way that we gain the maximum information, to understand the tree – If a person's hair length is less than 5 Inches, weight greater than 55 KGs then there are 80% chances for that person being a Male.

If you are familiar with Predictive Modelling e.g., Logistic Regression, Random Forest etc. – You might be wondering what is the difference between a Logistic Model and Decision Tree!
Because in both the algorithms we are trying to predict a categorical variable.

There are a few fundamental differences between both but ideally both the approaches should give you the same results. The best use of Decision Trees is when your solution requires a representation. For example, you are working for a Telecom Operator and building a solution using which a call center agent can take a decision whether to pitch for an upsell or not!

There are very less chances that a call center executive will understand the Logistic Regression or the equations, but using a more visually appealing solution you might gain a better adoption from your call center team.

How does Decision Tree work?

There are multiple algorithms written to build a decision tree, which can be used according to the problem characteristics you are trying to solve. Few of the commonly used algorithms are listed below:

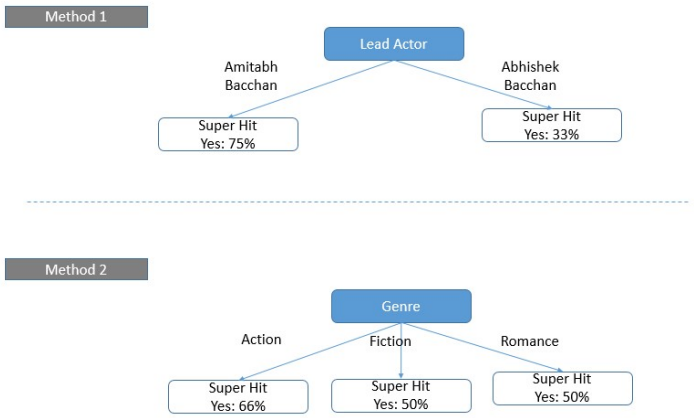
- ID3
- C4.5
- CART
- CHAID (CHi-squared Automatic Interaction Detector)
- MARS
- Conditional Inference Trees

Though the methods are different for different decision tree building algorithms but all of them works on the principle of Greediness. Algorithms try to search for a variable which give the maximum information gain or divides the data in the most homogenous way.

For an example, consider the following hypothetical dataset which contains Lead Actor and Genre of a movie along with the success on box office:

Lead Actor	Genre	Hit(Y/N)
Amitabh Bacchan	Action	Yes
Amitabh Bacchan	Fiction	Yes
Amitabh Bacchan	Romance	No
Amitabh Bacchan	Action	Yes
Abhishek Bacchan	*Action*	*No*
Abhishek Bacchan	*Fiction*	*No*
Abhishek Bacchan	*Romance*	*Yes*

Let say, you want to identify the success of the movie but you can use only one variable – There are the following two ways in which this can be done:



You can clearly observe that Method 1 (Based on lead actor) splits the data best while the second method (Based on Genre) have produced mixed results. Decision Tree algorithms do similar things when it comes to select variables.

There are various metrics which decision trees use in order to find out the best split variables. We'll go through them one by one and try to understand, what do they mean?

Entropy & Information Gain

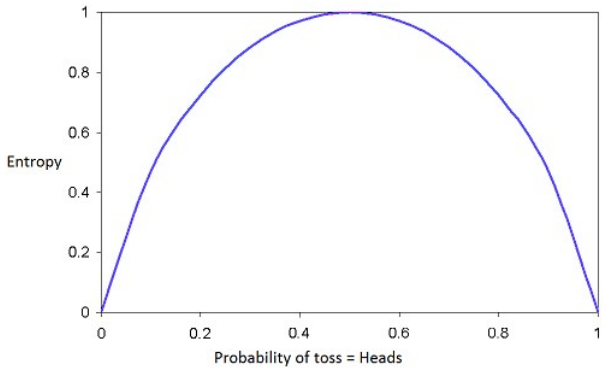
The word Entropy is borrowed from Thermodynamics which is a measure of variability or chaos or randomness. Shannon extended the thermodynamic entropy concept in 1948 and introduced it into statistical studies and suggested the following formula for statistical entropy:

$$H = -\sum_{j=1}^{n} P_j ln P_j$$

Where, H is the entropy in the system which is a measure of randomness.

Assuming you are rolling a fair coin and want to know the Entropy of the system. As per the formula given by Shann – Entropy would be equals to *-[0.5 ln(0.5) + 0.5 ln(0.5)].*

Which is equal to -0.69; which is the maximum entropy which can occur in the system. In other words, there will be maximum randomness in our dataset if the probable outcomes have same probability of occurrence.



Graph shown above shows the variation of Entropy with the probability of a class, we can clearly see that Entropy is maximum when probability of either of the classes is equal. Now, you can understand that when a decision algorithm tries to split the data, it selects the variable which will give us maximum reduction in system Entropy.

For the example of movie success rate – Initial Entropy in the system was:
$Entropy_{Parent} = -(0.57*ln(0.57) + 0.43*ln(0.43))$; Which is 0.68

Entropy after Method 1 Split

$Entropy_{left} = -(.75*ln(0.75) + 0.25*ln(0.25)) = 0.56$
$Entropy_{right} = -(.33*ln(0.33) + 0.67*ln(0.67)) = 0.63$

Captured impurity or entropy after splitting data using Method 1 can be calculated using the following formula: "*Entropy (Parent) – Weighted Average of Children Entropy*"

Which is,

*0.68 – (4*0.56 + 3*0.63)/7 = 0.09*

This number 0.09 is generally known as "Information Gain"

Entropy after Method 2 Split

$Entropy_{left} = -(.67*ln(0.67) + 0.33*ln(0.33)) = 0.63$
$Entropy_{middle} = -(.5*ln(0.5) + 0.5*ln(0.5)) = 0.69$
$Entropy_{right} = -(.5*ln(0.5) + 0.5*ln(0.5)) = 0.69$

Now using the method used above, we can calculate the Information Gain as:

*Information Gain = 0.68 – (3*0.63 + 2*0.69 + 2*0.69)/7 = 0.02*

Hence, we can clearly see that Method 1 gives us more than 4 times information gain compared to Method 2 and hence Method 1 is the best split variable.

Gain Ratio

Soon after the development of entropy mathematicians realized that Information gain is biased toward multi-valued attributes and to conquer this issue, "Gain Ratio" came into picture which is more reliable than Information gain. The gain ratio can be defined as:

$$Gain\ Ratio = \frac{Information\ Gain}{Split\ Info}$$

Where Split info can be defined as:

$$-\sum_{i=1}^{n} Di\ log_2 Di$$

Assuming we are dividing our variable into 'n' child nodes and Di represents the number of records going into various child nodes. Hence gain ratio takes care of distribution bias while building a decision tree.

For the example discussed above, for Method 1

Split Info = - ((4/7)log_2(4/7)) - ((3/7)*log_2(3/7)) = 0.98*

And Hence,

Gain Ratio = 0.09/0.98 = 0.092

Gini Index

There is one more metric which can be used while building a decision tree is Gini Index (Gini Index is mostly used in CART). Gini index measures the impurity of a data partition K, formula for Gini Index can be written down as:

$$Gini(K) = 1 - \sum_{i=1}^{n} P_i^{\,2}$$

Where m is the number of classes, and P_i is the probability that an observation in K belongs to the class. Gini Index assumes a binary split for each of the attribute in S, let say T_1 & T_2. The Gini index of K given this partitioning is given by:

$$Gini_S(K) = \frac{T_1}{T} Gini\ (T_1) + \frac{T_2}{T} Gini\ (T_2)$$

Which is nothing but a weighted sum of each of the impurities in split nodes. The reduction in impurity is given by:

$$Gini(K) - Gini_S(K)$$

Similar to Information Gain & Gain Ratio, split which gives us maximum reduction in impurity is considered for dividing our data.

Coming back to our movie example,

If we want to calculate *Gini(K)*-

$$Gini\ (K) = 1 - \left(\left[\frac{4}{7}\right]^2 + \left[\frac{3}{7}\right]^2\right)$$

$$= 0.49$$

Now as per our Method 1, we can get $Gini_s(K)$ as,

$$Gini_S(K) = \frac{T_1}{T} Gini\ (T_1) + \frac{T_2}{T} Gini\ (T_2)$$

$$= \frac{4}{7}\left\{1 - \left(\left[\frac{3}{4}\right]^2 + \left[\frac{1}{4}\right]^2\right)\right\} + \frac{3}{7}\left\{1 - \left(\left[\frac{1}{3}\right]^2 + \left[\frac{2}{3}\right]^2\right)\right\}$$

$$= 0.24 + 0.19$$

$$= 0.43$$

Now since we have understood all 3 of the commonly metrics, next question or confusion arises when we have to choose any one of them. There are a few drawbacks associated with all 3 of the metrics which is summarized in the table below:

Metrics	Drawback
Information Gain	Information Gain is biased towards multivariate attributes.
Gain Ratio	Gain Ratio generally prefers the unbalanced split of data where one of the child node has more number of entries compared to the others.
Gini Index	With more than 2 categories in the dataset, Gini Index gives unfavorable results. Apart from that it favors the split which results into equal sized children.

A brief about Random Forest

This tutorial is not targeted to talk about Random Forest but it will be injustice if we don't talk about it here. As the name suggests, Random Forest is a collection of multiple Decision Trees based on random sample of data (Both in terms of number of observations and variables), These Decision Trees also use the concept

of reducing Entropy in data and at the end of the algorithm, votes from different significant trees are ensemble to give a final response outcome for the observations.

So now, when you next get a chance to work on Random Forest, think it from a combination of decision trees and it will make your life easier.

Real Life Example for Decision Tree

Now we shall try to understand the working of a Decision Tree using an example.

Outlook	Temperature	Humidity	Windy	Play(0) or Don't Play(1)
Sunny	85	85	False	0
Sunny	80	90	True	0
Overcast	83	78	False	1
Rain	70	96	False	1
Rain	68	80	False	1
Rain	65	70	True	0
Overcast	64	65	True	1
Sunny	72	95	False	0
Sunny	69	70	False	1
Rain	75	80	False	1
Sunny	75	70	True	1
Overcast	72	90	True	1
Overcast	81	75	False	1
Rain	71	80	True	0

Our objective here is to use this data to build certain rules which can tell us whether we should play Golf on a given day or not.

We'll be using C5.0 algorithms which is widely used algorithm when it comes to decision trees. C5.0 is an advancement to C4.5 algorithms which is basically an extension to its predecessor ID3 algorithm. We'll be using C50 package which contains a function called C5.0 to build C5.0 Decision Tree using R.

Step 1: In order to use C50 package, we need to install the package and load it into R Session

```
install.packages("C50")
library(C50)
```

Step 2: Now we need to look the distribution of the data for any missing values outliers etc.
Which can be achieved by using summary() function in R. (We have read data into dataframe – tree_data)
Make sure that your dependent variable is a Factor variable and there are no Logical variables in dataset.

```
> summary(tree_data)
    Outlook    Temperature       Humidity        Windy
 overcast:4   Min.   :64.00   Min.   :65.00   Length:14
 rain    :5   1st Qu.:69.25   1st Qu.:71.25   Class :character
 sunny   :5   Median :72.00   Median :80.00   Mode  :character
              Mean   :73.57   Mean   :80.29
              3rd Qu.:78.75   3rd Qu.:88.75
              Max.   :85.00   Max.   :96.00
 Play
 0:5
 1:9
```

Step 3: Once we are sure and comfortable with data, we are good to go for tree building

```
built_tree<-C5.0(Play~.,data=tree_data)
```

We are storing the tree into built_tree R Object

Step 4: Let's check the summary of the built tree

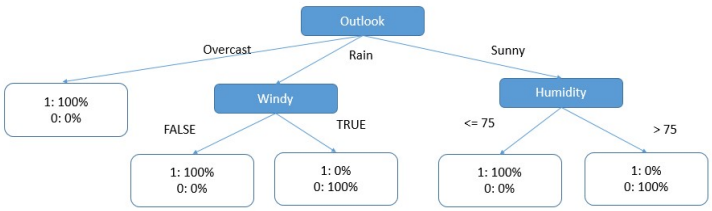
```
summary(built_tree)
```

Which looks like:

```
> summary(built_tree)

Call:
C5.0.formula(formula = Play ~ ., data = tree_data)


C5.0 [Release 2.07 GPL Edition]        Thu Apr 28 10:48:39 2016
-------------------------------

Class specified by attribute `outcome'

Read 14 cases (5 attributes) from undefined.data

Decision tree:

Outlook = overcast: 1 (4)
Outlook = rain:
:...Windy = FALSE: 1 (3)
:   Windy = TRUE: 0 (2)
Outlook = sunny:
:...Humidity <= 75: 1 (2)
    Humidity > 75: 0 (3)


Evaluation on training data (14 cases):

          Decision Tree
        ----------------
        Size      Errors

           5    0( 0.0%)   <<


         (a)   (b)    <-classified as
        ----  ----
           5          (a): class 0
                 9    (b): class 1
```

In summary we can see the structure of the tree and at the bottom accuracy and the confusion matrix based on the training data; We can see that for our example all of them are getting classified correctly.

C5.0 built a tree which looks like:



Now, you are ready to build your own tree and predict for the new data coming in.

Drawbacks of Using Decision Trees

Till now we have talked about various benefits of Decision Trees, algorithm behind building a tree but there are a few drawbacks or precautions which we should be aware of before going ahead with Decision trees:

- Decision trees are susceptible to change in your data; Even a small change in data can result into a completely new tree structure
- Decision trees tend to overfit but this can be overcome by pruning your trees
- You might face a problem when you are trying to do an out of sample testing or prediction