**Decision Tree : Meaning**  
  
A decision tree is a graphical representation of possible solutions to a decision based on certain conditions. It is called a decision tree because it starts with a single variable, which then branches off into a number of solutions, just like a tree.

A decision tree has three main components:

1. **Root Node :**The top most node is called Root Node. It implies the best predictor (independent variable).
2. **Decision / Internal Node :** The nodes in which predictors (independent variables) are tested and each branch represents an outcome of the test
3. **Leaf / Terminal Node :** It holds a class label (category) - Yes or No (Final Classification Outcome).

|  |
| --- |
|  |
| Decision Tree Explained |

**Advantages and Disadvantages of Decision Tree**  
  
**Advantages :**

1. Decision tree is easy to interpret.
2. Decision Tree works even if there are nonlinear relationships between variables. It does not require linearity assumption.
3. Decision Tree is not sensitive to outliers.

**Disadvantages :**

1. Decision tree model generally overfits. It means it does not perform well on validation sample.
2. It assumes all independent variables interact with each other, it is generally not the case every time.

**Terminologies related to decision tree**  
  
**1. Pruning : Correct Overfitting**  
  
It is a technique to correct overfitting problem. It reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. It is used to remove anomalies in the training data due to noise or outliers. The pruned trees are less complex trees.  
  
**Pre-Pruning Method : Significance Testing**

It refers to the process in which we stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node.   
  
**Post Pruning Method : Cost Complexity**  
  
Post pruning refers to the following process -

1. Build full tree
2. Prune it

The cost complexity is one of the most popular post-pruning method. It is measured by the following two parameters −

1. Number of leaves in the tree (i.e. size of the tree)
2. Error rate of the tree (i.e. misclassification rate or Sum of Squared Error)

The ‘CP’ stands for **Complexity Parameter** of the tree. We want the cp value of the smallest tree that has smallest cross validation error. In regression, this means that the overall R-squared must increase by cp at each step.  
  
In other words, it refers to trade-off between the size of a tree and the error rate to help prevent overfitting. **Thus, large trees with a low error rate are penalized in favor of smaller trees.**

CP nsplit rel error xerror xstd

1 0.046948 0 1.00000 1.00000 0.057151

2 0.023474 4 0.75587 0.81221 0.053580

3 0.015649 5 0.73239 0.83099 0.053989

4 0.011737 10 0.64789 0.87324 0.054867

5 0.010955 12 0.62441 0.89671 0.055328

6 0.010000 17 0.56808 0.89671 0.055328

In this case, we pick the tree having **CP = 0.023474** as it has least cross validation error (xerror). The rel error of each iteration of the tree is the fraction of misclassified cases in the iteration relative to the fraction of misclassified cases in the root node.

**Cost Complexity (cp)**is the tuning parameter in CART.

**2. Splitting**  
  
It is a process of dividing a node into two or more sub-nodes.  
  
**3. Branch**  
  
A sub section of entire tree is called branch.  
  
**4. Parent Node**  
  
A node which splits into sub-nodes.  
  
**5. Child Node**  
  
It is the sub-node of a parent node.  
 **6. Surrogate Split**  
  
When you have missing data, decision tree returns predictions when they include surrogate splits. If parameter value of surrogate is set 2, it means if the primary splitter is missing, we use the number one surrogate. If the number one surrogate is missing, then we use the number two surrogate.

**Classification and Regression Tree (CART)**  
 **Classification Tree**  
The outcome (dependent) variable is a categorical variable (binary) and predictor (independent) variables can be continuous or categorical variables (binary).  
**How Decision Tree works:**

1. Pick the variable that gives the best split (based on lowest Gini Index)
2. Partition the data based on the value of this variable
3. Repeat step 1 and step 2. Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met. (Alternatively, the data are split as much as possible and then the tree is later pruned.

**Algorithms of Classification Tree**  
  
**1. Split Method : Gini Index**

Gini Index measures impurity in node. It varies between 0 and (1-1/n) where n is the number of categories in a dependent variable.

|  |
| --- |
|  |
| Gini Index |

In this equation, p refers to probability of class. In layman's language, it can be read as -

1 – ( P(class1)^2 + P(class2)^2 + … + P(classN)^2)

**Gini Index favors larger partitions.**  
 **Important Points :**

1. Zero gini index implies perfect classification.
2. (1 - (1/ No. of classes) implies worst classification
3. We want a variable split having a low Gini Index.
4. For binary dependent variable, max gini index value can be 0.5. See the calculation below.

= 1 – ((1/2)2 + (1/2)2)  
= 1 - 2\*(1/2)2  
= 1- 2\*(1/4)  
= 1-0.5  
= 0.5  
  
**2. Entropy / Information Gain**  
  
Another splitting criteria method for classification tree is entropy. The formula of this technique is shown below -

|  |
| --- |
|  |
| Entropy |

It can be read as -

P(class1)\*log(P(class1),2) + P(class2)\*log(P(class2),2) + … + P(classN)\*log(P(classN),2)

**It favors partitions that have small counts but many distinct values.**

*Smaller value of Entropy signifies a good classification.*

**Information Gain** can be calculated by using the following formula -

= Entropy(parent) - Weighted Sum of Entropy(Children)

**Which is better - Entropy or Gini**

Both splitting criterias are approximately similar and produces similar result in 95% of the cases. Gini is comparatively faster than Entropy as it does not require calculation of log.

**Regression Tree**  
The outcome (dependent) variable is a continuous variable and predictor (independent) variables can be continuous or categorical variables (binary).  
 **Split Method:  Least-Squared Deviation or Least Absolute Deviation**

The impurity of a node is measured by the Least-Squared Deviation (LSD), which is simply the within variance for the node.  
  
**How to calculate best split manually**  
  
Suppose you have two independent variables that are binary and continuous (numeric) in nature and target variable is binary which has only two values - 0/1. Sample data is shown below -

|  |
| --- |
|  |
| Sample Data |

**Gini Index : Var1**  
 **Var1 = 1**

1. Var1 has 4 cases out of 10 where it is equal to 1.
2. For Var1 = 1 & Target = 1, 1/4 cases have target=1.
3. For Var1 = 1 & Target = 0,  3/4 cases have target=0.

**Gini Index** = 1-((1/4)^2 + (3/4)^2) = 0.375  
 **Var1 = 0**

1. Var1 has 6 cases out of 10 where it is equal to 0.
2. For Var1 = 0 and Target = 1, 4/6 cases have target = 1
3. For Var1 = 0 & Target = 0,  2/6 cases have target = 0.

**Gini Index** = 1-((4/6)^2 + (2/6)^2) = 0.4444  
  
By adding weight and sum each of the gini indices:

**Gini Index (Target, Var1) =** 4/10 \* 0.375 + 6/10 \* 0.444 = 0.41667

**Gini Index : Var2**   
  
For numeric variables, we first need to find the actual value as a threshold which gives the best split. There are k−1 possible splits on continuous variable, all of which should be used to determine an optimal split. Let's choose randomly 32 as a threshold.

**Var2 >= 32**

1. Var2 has 8 cases (8/10) where it is greater than or equal to 32.
2. For Var2 >= 32 and target = 1,  5/8 cases have target = 1.
3. For Var2 >= 32 & target = 0: 3 / 8 cases have target = 0.

**Gini Index =** 1-((5/8)^2 + (3/8)^2) = 0.46875  
  
**Var2 < 32**

1. Var2 has 2 cases out of 10 where it is less than 32
2. For Var2 < 32 and target = 1, 0 cases have target = 1
3. For Var2 < 32 and target = 0,  2/2 cases have target = 0.

**Gini Index =** 1-((0/2)^2 + (2/2)^2) = 0

**Gini Index(Target, Var2) =**8/10 \* 0.46875 + 2/10 \* 0 = 0.375

Since Var2 has lower Gini Index value, it should be chosen as a variable that gives best split. The next step would be to take the results from the split and further partition.  Let’s take the 8 / 10 cases and calculate Gini Index on the following 8 cases.

|  |  |  |
| --- | --- | --- |
| **Target** | **Var1** | **Var2** |
| 1 | 0 | 34 |
| 1 | 0 | 45 |
| 1 | 0 | 54 |
| 1 | 0 | 41 |
| 1 | 1 | 47 |
| 0 | 1 | 51 |
| 0 | 1 | 65 |
| 0 | 0 | 41 |

Try similar steps as what we performed earlier.  
  
**How to get predicted probability**  
  
Let' say an observation falls into lead node1 in which there are 6 class A’s and 4 class B’s, then probability P(ClassA | observation) = 6 / 10 = 0.6. Similarly, P(ClassB | observation) = 0.4.

**Analysis of German Credit Data**

We run decision tree on credit data which gives you background of the financial project and how predictive modeling is used in banking and finance domain.

The German Credit Data contains data on 20 variables and the classification whether an applicant is considered a Good or a Bad credit risk for 1000 loan applicants.

The objective of the model is whether to approve a loan to a prospective applicant based on his/her profiles.

1. Make sure all the categorical variables are converted into factors.
2. The function rpart will run a regression tree if the response variable is numeric, and a classification tree if it is a factor.
3. **rpart parameter -** **Method -**"class" for a classification tree ; "anova" for a regression tree
4. **minsplit :**minimum number of observations in a node before splitting. Default value - 20
5. **minbucket :**minimum number of observations in terminal node (leaf). Default value - 7 (i.e. minsplit/3)
6. **xval :**Number of cross validations
7. **Prediction (Scoring) :**If type = "prob": This is for a classification tree. It generates probabilities - Prob(Y=0) and Prob(Y=1).
8. **Prediction (Classification) :** If type = "class": This is for a classification tree. It returns 0/1.

**Decision Tree**

#read data file  
mydata= read.csv("E:\\Backup 14.03.2021\\Desktop\\Imarticus Post\\Online DSP\\german\_credit.csv")  
  
# Check attributes of data  
str(mydata)

# Check number of rows and columns  
dim(mydata)  
  
# Make dependent variable as a factor (categorical)  
mydata$Creditability = as.factor(mydata$Creditability)  
  
set.seed(123)

# Split data into training (70%) and validation (30%)  
dt = sort(sample(nrow(mydata), nrow(mydata)\*.7))  
train<-mydata[dt,]  
val<-mydata[-dt,]

# Check number of rows in training data set  
nrow(train)  
  
# To view dataset  
edit(train)  
# Decision Tree Model  
library(rpart)  
mtree <- rpart(Creditability~., data = train, method="class", control = rpart.control(minsplit = 20, minbucket = 7, maxdepth = 10, usesurrogate = 2, xval =10 ))

#maxdepth - **maxdepth** : This parameter is used to set the maximum depth of a tree. Depth is the length of the longest path from a Root node to a Leaf node. Setting this parameter will stop growing the tree when the depth is equal the value set for **maxdepth**

**#** usesurrogate - how to use surrogates in the splitting process. For value 2, if all surrogates are missing, then send the observation in the majority direction

mtree  
  
n= 700

Next, the structure of the tree is presented. First is provided a legend to be able to read the tree.

|  |
| --- |
| node), split, n, loss, yval, (yprob)  \* denotes terminal node |

This tells us that a node number will be provided, followed by a split, the number of entities at that node, how many entities are incorrectly classified (the loss), the default classification for the node (yval), and then the distribution of classes in that node (yprob). The distribution is ordered by the classes, and is the same order for all nodes. We are told that a ``\*'' denotes a terminal node of the tree (i.e., the tree is not split any further at that node).

|  |
| --- |
| 1) root 700 208 1 (0.29714286 0.70285714) |

The root node represents all 700 observations. Stopping at the root node, in building a model, represents a model that simply classifies every one with whatever class is the majority in the training dataset. Skipping the 700 for a moment, we see that the report tells us that the majority class for the root node (the yval) is 1 then it tells us how many of the 700 will be incorrectly classified as 1 technically called the loss.

#Plot tree  
plot(mtree)  
text(mtree)

#Beautify tree  
library(rattle)  
library(rpart.plot)  
library(RColorBrewer)  
  
#view1  
prp(mtree, faclen = 0, cex = 0.8, extra = 1)  
#extra=1 - Display the number of observations that fall in the node

# faclen - Length of factor level names in splits. Default 3, #meaning [abbreviate](https://www.rdocumentation.org/link/abbreviate?package=rpart.plot&version=3.0.9) to three characters.

#cex - the text size

#view2 - total count at each node  
tot\_count <- function(x, labs, digits, varlen)  
{paste(labs, "\n\nn =", x$frame$n)}

prp(mtree, faclen = 0, cex = 0.8, node.fun=tot\_count)

# node.fun - The function that generates the text at the node labels.

#Pruning#

printcp(mtree)  
# **printcp** ( x ) where x is the rpart object. This function provides the optimal prunings based on the cp value. We prune the tree to avoid any overfitting of the data. The convention is to have a small tree and the one with least cross validated error given by **printcp**() function i.e. 'xerror'.

bestcp <- mtree$cptable[which.min(mtree$cptable[,"xerror"]),"CP"]  
bestcp

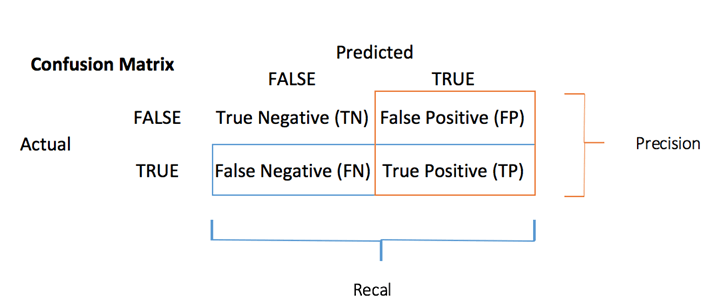
## # Prune the tree using the best cp. pruned <- prune(mtree, cp = bestcp)

## # Plot pruned tree prp(pruned, faclen = 0, cex = 0.8, extra = 1) # confusion matrix (training data) conf.matrix <- table(val$Creditability, predict(pruned,type="class")) rownames(conf.matrix) <- paste("Actual", rownames(conf.matrix), sep = ":") colnames(conf.matrix) <- paste("Pred", colnames(conf.matrix), sep = ":") print(conf.matrix)

## Measure performance

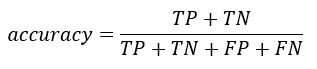
You can compute an accuracy measure for classification task with the [**confusion matrix**](https://www.guru99.com/confusion-matrix-machine-learning-example.html)**:**

The **confusion matrix** is a better choice to evaluate the classification performance. The general idea is to count the number of times True instances are classified are False.



Each row in a confusion matrix represents an actual target, while each column represents a predicted target.

You can compute the **accuracy test** from the confusion matrix:



It is the proportion of true positive and true negative over the sum of the matrix. With R, you can code as follow:

accuracy\_Test <- sum(diag(conf.matrix)) / sum(conf.matrix)

Code Explanation

* sum(diag(conf\_matrix)): Sum of the diagonal
* sum(conf\_matrix): Sum of the matrix.

You can print the accuracy of the test set:

print(paste('Accuracy for test', accuracy\_Test))

You have a score of 78 percent for the test set.

## #Scoring val1 = predict(pruned, val, type = "prob")

# What's a factor and why would you use it?

The term factor refers to a statistical data type used to store categorical variables. The difference between a categorical variable and a continuous variable is that a categorical variable can belong to a **limited number of categories**. A continuous variable, on the other hand, can correspond to an infinite number of values.

It is important that R knows whether it is dealing with a continuous or a categorical variable, as the statistical models you will develop in the future treat both types differently.

A good example of a categorical variable is gender. In many circumstances you can limit the gender categories to "Male" or "Female".

**Factor** variables are categorical variables that can be either numeric or string variables. There are a number of advantages to converting categorical variables to factor variables. Perhaps the most important advantage is that they can be used in statistical modeling where they will be implemented correctly, i.e., they will then be assigned the correct number of degrees of freedom. Factor variables are also very useful in many different types of graphics. Furthermore, storing string variables as factor variables is a more efficient use of memory. To create a factor variable, we use the factor function. The only required argument is a vector of values which can be either string or numeric. Optional arguments include the levels argument, which determines the categories of the factor variable, and the default is the sorted list of all the distinct values of the data vector.