Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions. It is mostly used in Machine Learning and Data Mining applications using R.

Examples of use of decision tress is − predicting an email as spam or not spam,

predicting of a tumor is cancerous or

predicting a loan as a good or bad

credit risk based on the factors in each of these. Generally, a model is created with observed data also called training data.

Then a set of validation data is used to verify and improve the model. R has packages which are used to create and visualize decision trees. For new set of predictor variable, we use this model to arrive at a decision on the category (yes/No, spam/not spam) of the data.

The R package **"party"** is used to create decision trees.

Install R Package

Use the below command in R console to install the package. You also have to install the dependent packages if any.

install.packages("party")

The package "party" has the function **ctree()** which is used to create and analyze decison tree.

Syntax

The basic syntax for creating a decision tree in R is −

ctree(formula, data)

Following is the description of the parameters used −

* **formula** is a formula describing the predictor and response variables.
* **data** is the name of the data set used.

Input Data

We will use the R in-built data set named **readingSkills** to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker or not.

Here is the sample data.

# Load the party package. It will automatically load other

# dependent packages.

library(party)

# Print some records from data set readingSkills.

print(head(readingSkills))

library(party)

# Create the input data frame.

input.dat <- readingSkills[c(1:105),]

# Give the chart file a name.

png(file = "decision\_tree.png")

# Create the tree.

output.tree <- ctree(

nativeSpeaker ~ age + shoeSize + score,

data = input.dat)

# Plot the tree.

plot(output.tree)

# Save the file.

dev.off()

# R - Random Forest

­­In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

An error estimate is made for the cases which were not used while building the tree. That is called an **OOB (Out-of-bag)**error estimate which is mentioned as a percentage.

The R package **"randomForest"** is used to create random forests.

Install R Package

Use the below command in R console to install the package. You also have to install the dependent packages if any.

install.packages("randomForest)

The package "randomForest" has the function **randomForest()**which is used to create and analyze random forests.

Syntax

The basic syntax for creating a random forest in R is −

randomForest(formula, data)

Following is the description of the parameters used −

* **formula** is a formula describing the predictor and response variables.
* **data** is the name of the data set used.

Input Data

We will use the R in-built data set named readingSkills to create a decision tree. It describes the score of someone's readingSkills if we know the variables "age","shoesize","score" and whether the person is a native speaker.

Here is the sample data.

# Load the party package. It will automatically load other

# required packages.

library(party)

# Print some records from data set readingSkills.

print(head(readingSkills))

We will use the **randomForest()** function to create the decision tree and see it's graph.

# Load the party package. It will automatically load other

# required packages.

library(party)

library(randomForest)

# Create the forest.

output.forest <- randomForest(nativeSpeaker ~ age + shoeSize + score,

data = readingSkills)

# View the forest results.

print(output.forest)

# Importance of each predictor.

print(importance(fit,type = 2))