Decision Trees and Random Forest classifiers

R Mofidi

24/06/2020

Decision tree Analysis

Decision Trees are a class of tree like graph algorithms. In addition to being a predictive model decision trees use this tree like structure to illustrate the possible decisions and their consequences. It is one way of combining an algorithm and a decision support tool as well as a machine learning paradigm. A decision tree consists of 3 constituents (no pun intended), a node which represents an attribute and a branch which represents the consequences of classifying the data set based on that attribute. Attributes can occur as a result of an active decision (a decision node, usually marked as a square), a chance node which is the consequences of an event outside of the decision makers' control and an end node (leaf) which denotes one of the possible final outcomes of the decision tree algorithm. In this way, the findings of the decision tree can be illustrated in a compact and easy to follow format using an influence diagram which describes the relationship between actions and consequences.

In this way decision tree classifiers utilize the topography of a decision tree to map observations about a target item to conclusions about the value of the target item. Where the target value has a finite set of values (ordinal or nominal), the process is known as a classification tree. If the Target value is a continuous variable the resultant model is called regression tree. The umbrella term Classification and Regression Tree (CART) is used to group both processes. This class of classifiers were devised by the distinguished American statistician Leo Breiman who was also involved in their eventual evolution into random forest classifiers. The CARTs work from top down by choosing the variables which best splits a set of items into the intended classes.This is performed by calculating the Gini coefficient for the putative input variables. The Gini coefficient denotes the determinant ability of a variable in correctly classifying the data according to the output variable (which we are trying to predict). It is named after the Italian economist, sociologist and statistician Corrado Gini. The algorithms created are simple to understand and require little preparation, It is possible to use categorical as well as continuous variables and create a white box model which is transparent to the user. It is possible to validate using statistical models such as a confusion matrix or receiver operator characteristic test.

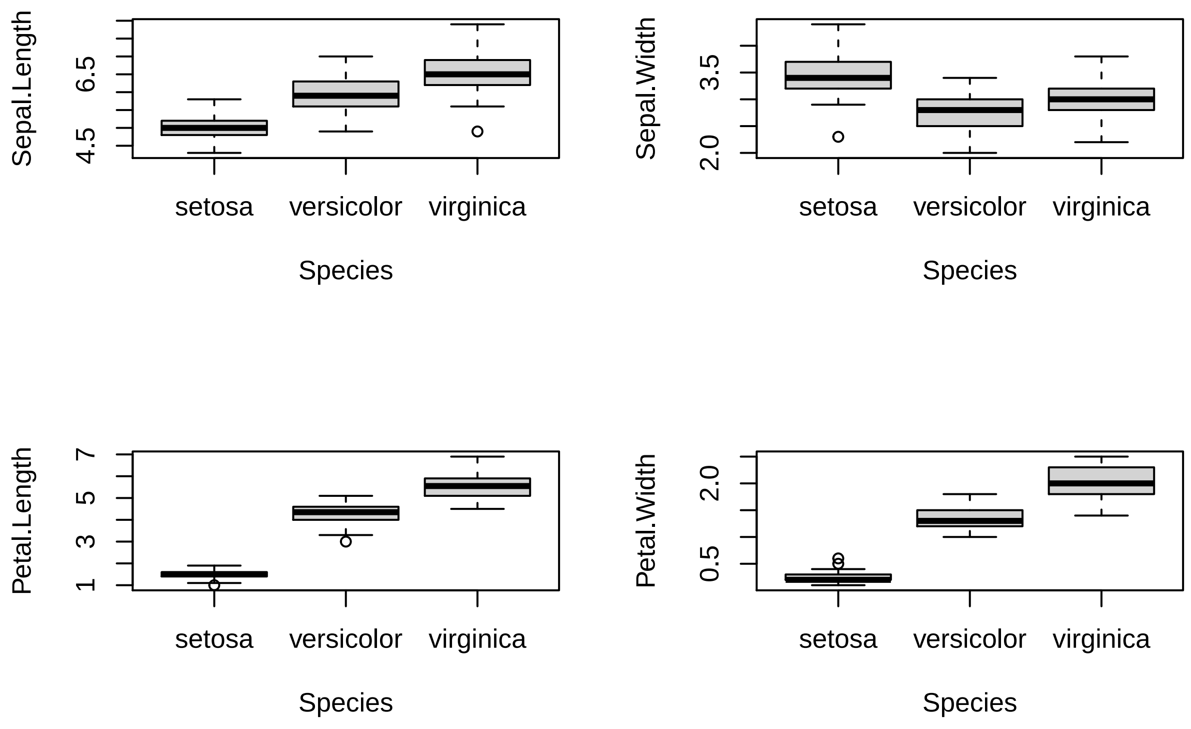
Decision tree analysis involves a series of simple steps the first of which involves identifying and using the variable which best separates the dataset in accordance with the outcome of interest. This creates 2 leaves or children nodes. the remaining data set in each leaves is divided in a similar manner until the groups are either too small or pure i.e. contain only a single outcome (pure). Some software programs create competing Trees and eliminate the ones which have lower predictive ability based on in sample testing.

The Fisher's or Anderson's iris dataset

The following example is created using a commonly encountered database in R known as the Iris database. This database uses the width and length of the petal and sepal of iris flowers to classify the iris flowers into the 3 different species: 1- Setosa 2- Versicolor 3- Virginica

The following boxplot illustrates how these variables differ in the 3 species of iris:

```r
par(mfrow=c(2,2))
boxplot(Sepal.Length~Species, data = iris)
boxplot(Sepal.Width~Species, data = iris)
boxplot(Petal.Length~Species, data = iris)
boxplot(Petal.Width~Species, data = iris)
```



The following code describes how decision trees (classification and Regression Trees) can be developed in R.

Install the appropriate libraries and datasets

Clearly the first step in R involves downloading the appropriate packages required:

```r
data(iris)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2

## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(ggplot2)
library(lattice)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Separate training and test datasets

The next step involves partitioning the dataset into training and test datasets. This creates a training dataset for developing the tree and a testing dataset for cross validation:

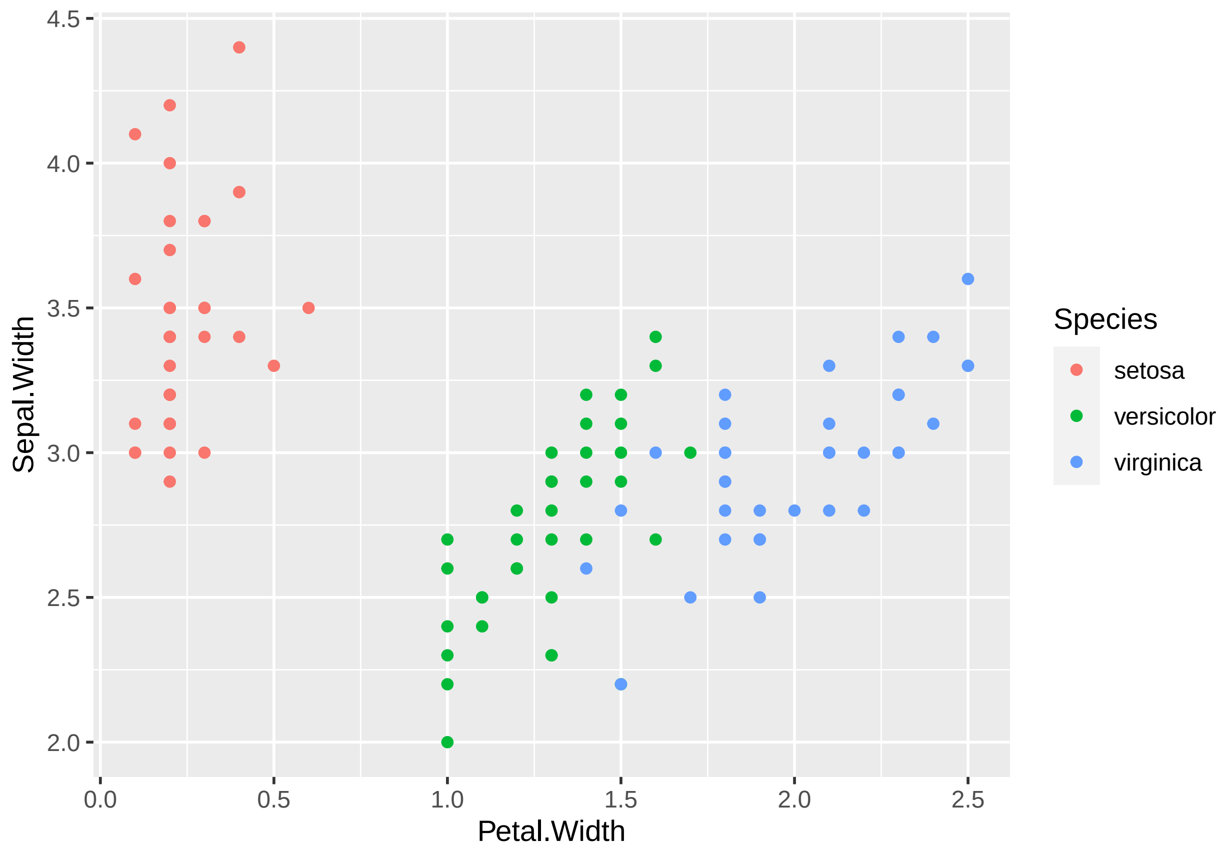
```
inTrain3<- createDataPartition(y=iris$Species, p=0.7, list=FALSE)
trainingDS<- iris[inTrain3,]
testingDS<- iris[-inTrain3,]
dim (trainingDS); dim(testingDS)
```

```
## [1] 105   5
```

```
## [1] 45  5
```

Make a plot to view data separation

It is very useful to see the separation between the data clusters as this can be used to predict whether the correct classifier has been selected for the task



Developing the decision tree "rpart" model.

The following r code creates the decision tree classifier using the training dataset (trainingDS). This is saved as "modfit". Using the *rattle* package it is possible to draw an aesthetically pleasing tree.

```r
library(caret)
modfit<- train(Species~.,method="rpart", data=trainingDS)
print(modfit$finalModel)
```

```
## n= 105
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 105 70 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.35 35  0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.35 70 35 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Length< 4.75 33  1 versicolor (0.00000000 0.96969697 0.03030303) *
##     7) Petal.Length>=4.75 37  3 virginica (0.00000000 0.08108108 0.91891892) *
```

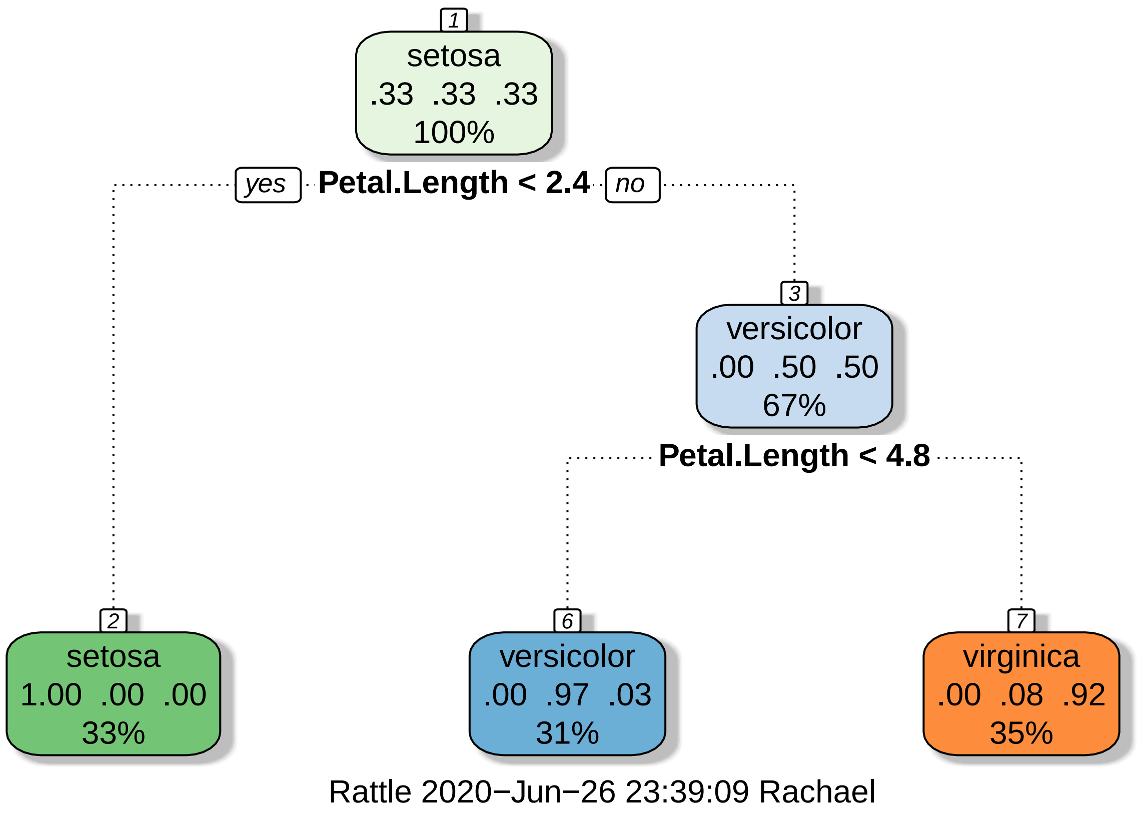
Plot the "modfit" classification Tree

```
## Warning: package 'rattle' was built under R version 4.0.2
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```



Cross validation by performing the prediction on the testing sample

Cross validation using testing dataset is the final step in forming a classifier. Strictly speaking it is not out of sample testing as the datset comes from the same dataset as the training sample. However as the training dataset has not been used to develop the model, cross validation is a valid replacement for out of sample testing. The following code performs the cross validation process:

```
predict(modfit, newdata=testingDS)
```

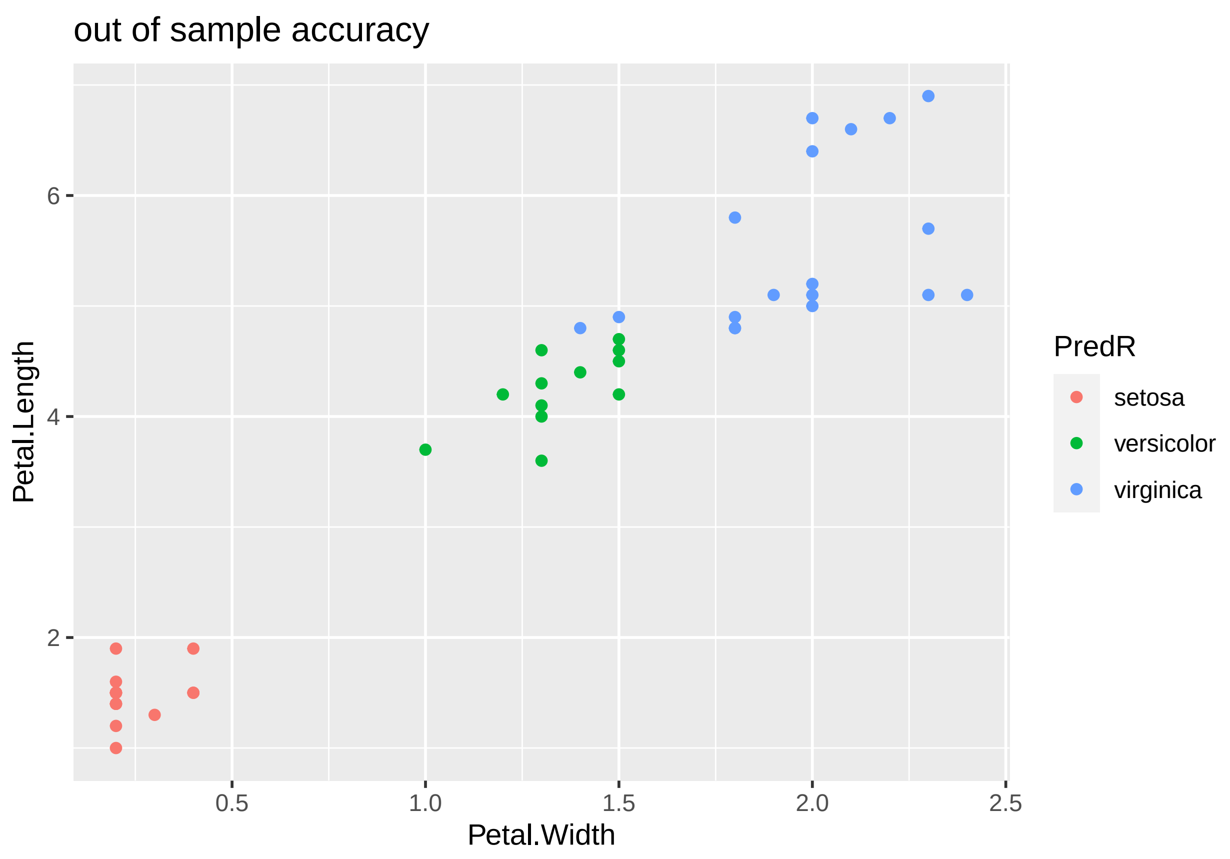
```
##  [1] setosa     setosa     setosa     setosa     setosa     setosa    
##  [7] setosa     setosa     setosa     setosa     setosa     setosa    
## [13] setosa     setosa     setosa     versicolor versicolor versicolor
## [19] versicolor versicolor virginica  versicolor virginica  versicolor
## [25] versicolor virginica  versicolor versicolor versicolor versicolor
## [31] virginica  virginica  virginica  virginica  virginica  virginica 
## [37] virginica  virginica  virginica  virginica  virginica  virginica 
## [43] virginica  virginica  virginica 
## Levels: setosa versicolor virginica
```

```
PredR<- predict(modfit, newdata=testingDS)
summary(PredR)
```

```
##     setosa versicolor  virginica 
##         15         12         18
```

The Cross validation

```
library(ggplot2)
qplot(Petal.Width, Petal.Length, colour=PredR,data=testingDS, main = "out of sample accuracy")
```



As you can see a simple CART classifier can accurately classify the iris database into the 3 different species based on the features included. In fact it is hard to see how the classifier can be improved upon without introducing new variables.

Decision trees are used extensively as classifiers and decision aids in many areas including healthcare. They provide a mixture of simplicity, utility and functionality. For many uses classification and regression trees are more than sufficient including many healthcare decision support models (2, 3). Random Forest classifiers which are discussed below represent the future of CART algorithms in the world of deep learning as data becomes cheaper and more freely available.

Random Forests and Ensemble Classifiers

Ensemble Classifiers consist of multiple classifiers which by themselves may be weak (i.e. have low predictive ability) but combining them into an ensemble improves their predictive ability significantly. A similar concept exists in statistics with the difference that machine learning the ensembles contain a finite set of constituent algorithms (and are used for practical machine learning), whilst a statistical ensemble is infinite and more of a theoretical construct. The design of this class of algorithms is based on the *ensemble theory* which states that a trained ensemble of algorithms can represent a single supervised trained algorithm. In general an ensemble algorithm functions better if it contains a diverse set of constituent algorithms trined on random selection of the data.

In fact the individual constituents of the ensemble do not necessarily have to be weak classifiers, being part of the ensemble means that they do not need to be complex in structure which in turn helps protect against over-fitting (over-training). They key to a successful ensemble is "stochastic discrimination" which is discussed below. There are a few types of ensemble classifiers in existence including: • Bayes optimal classifiers • Bootstrap aggregating methods such as Random forest classifiers • Boosting: such as Adaboost

In this paper we discuss random forest classifiers as they represent an extension of decision tree algorithms.

Random Forest classifiers

An example of ensemble classifiers are random forests. Random forests were first designed by the American data scientist Tin Kam Ho in the 1990s. They are made up of an ensemble made up of (often a large) number of tree classifiers which are added together in order to improve their classification ability. It is a bootstrap aggregating method which means each of these decision tree classifiers contributes to the eventual decision with an equal weight. Each tree acts as a weak classifier and together a large number of trees form a random forest. The diversity of classifiers within the ensemble is the key to its performance.

If you remember in order to develop an ensemble classifier two assumptions will need to be fulfilled. The second of these two assumptions was that each weak classifier should make its choice independently of the other classifiers. This concept is called "Stochastic discrimination". In order to fulfill this assumption each decision tree needs to undergo a different training process using a random selection of the training data set. This is why large, yet discrete datasets are ideal for random forests. This is a dilemma as often the training data is scarce. One way of getting around this is randomization (random selection) of training data he used to train each tree classifier. In order to develop a random forest during training process a number of hyperplanes are selected at random and are trained using a random sub set of they are available training data. The final classification is performed using a majority vote system to perform the classification task.

The following is an example of a small random forest classifier created using the iris database described above:

Loading the required packages and database

```
data(iris); library(ggplot2); library(randomForest);library(caret)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Partitioning the data into training and testing sets

```
inTrainRF<-createDataPartition(y=iris$Species, p=0.7, list=FALSE)
training4<-iris[inTrainRF,]
testing4<-iris[-inTrainRF,]
```

Training the random forest classifier:

This involves setting the output variable as the Species and the rest of the variables as input variables.

```
modFit<- train(Species~.,data=training4, method="rf", prox=TRUE)
modFit
```

```
## Random Forest
##
## 105 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9459176  0.9175041
##   3     0.9468223  0.9188593
##   4     0.9448223  0.9156632
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

As you can see the in sample accuracy of the data is excellent and it over performs most linear data models as well as classification and regression trees (only marginally). Clearly concerns regarding over-training (over fitting) the data exists. This is why cross validation and data visualization is important. This is what the testing sample “testing4” is used for.

Random forests can be complicated in order to understand their underlying anatomy, it is possible to view the constituent trees making up the random forest classifier:

```
getTree(modFit$finalModel,k=2)
```

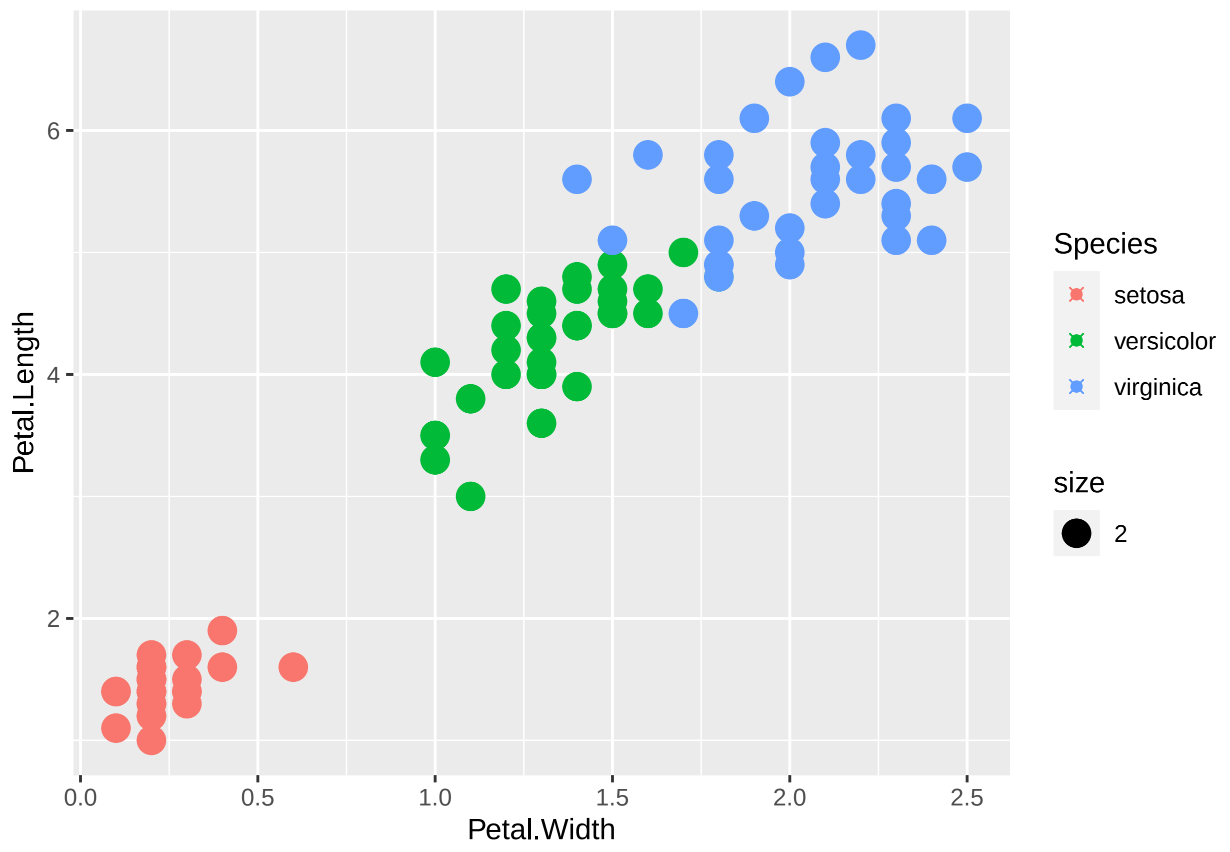
```
##    left daughter right daughter split var split point status prediction
## 1              2              3         4        0.80      1          0
## 2              0              0         0        0.00     -1          1
## 3              4              5         3        5.05      1          0
## 4              6              7         4        1.90      1          0
## 5              0              0         0        0.00     -1          3
## 6              8              9         4        1.75      1          0
## 7              0              0         0        0.00     -1          3
## 8              0              0         0        0.00     -1          2
## 9          10          11       2       3.00     1          0
## 10          0           0       0       0.00    -1          3
## 11          0           0       0       0.00    -1          2
```

```
irisP<- classCenter(training4[,c(3,4)], training4$Species, modFit$finalModel$prox)
irisP<-as.data.frame(irisP);irisP$Species<- rownames(irisP)
P<- qplot(Petal.Width, Petal.Length, col=Species, size=2,Shape=4,data=training4)
```

Visualizing the classifier

```
## Warning: Ignoring unknown parameters: Shape
```

```
P+geom_point(aes(x=Petal.Width, y=Petal.Length, col=Species), size=2, shape=4, data=irisP)
```



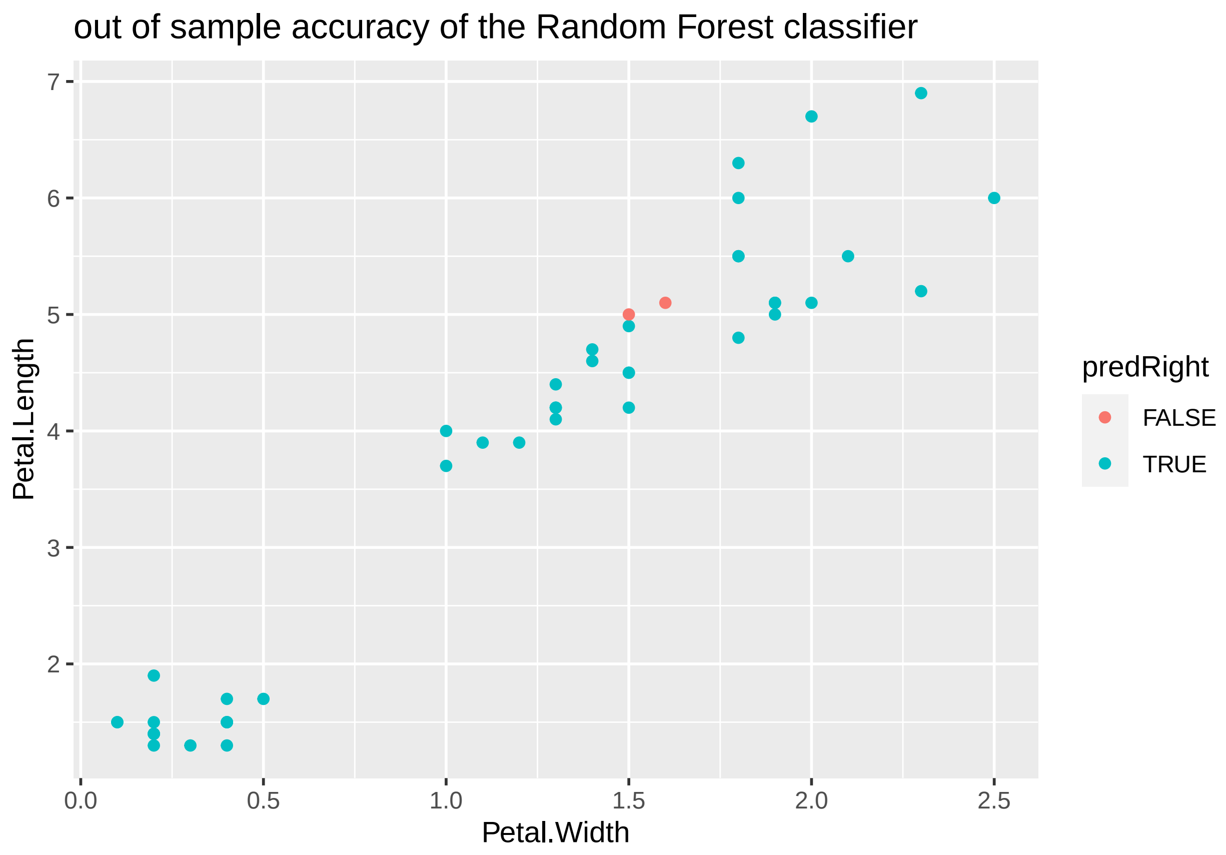
Making the prediction in the testing sample

The following R script codes for the process of cross validation of the classifier.

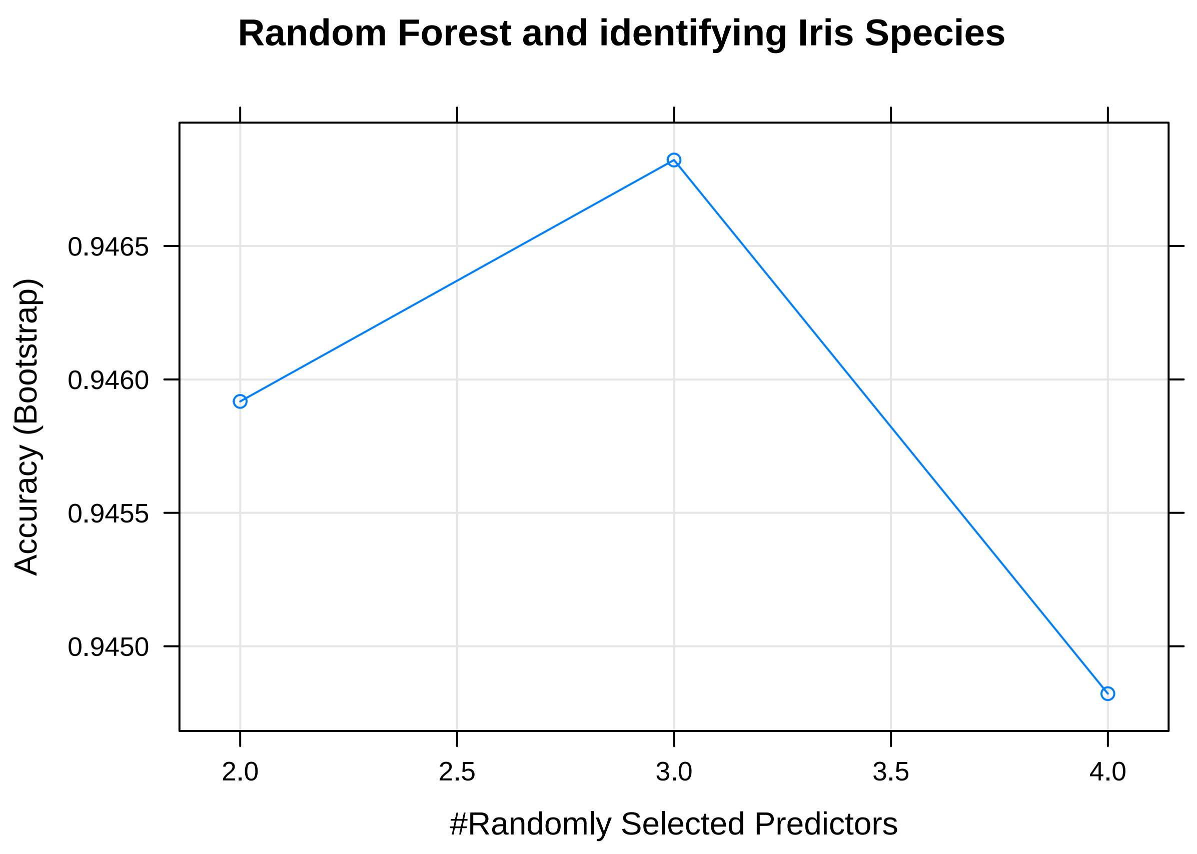
```
pred<-predict(modFit, testing4)
testing4$predRight<-pred == testing4$Species
table(pred,testing4$Species)
```

```
##
## pred         setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         14         1
##   virginica       0          1        14
```

The following plot illustrates the correct and incorrect classifications made by the random forest.

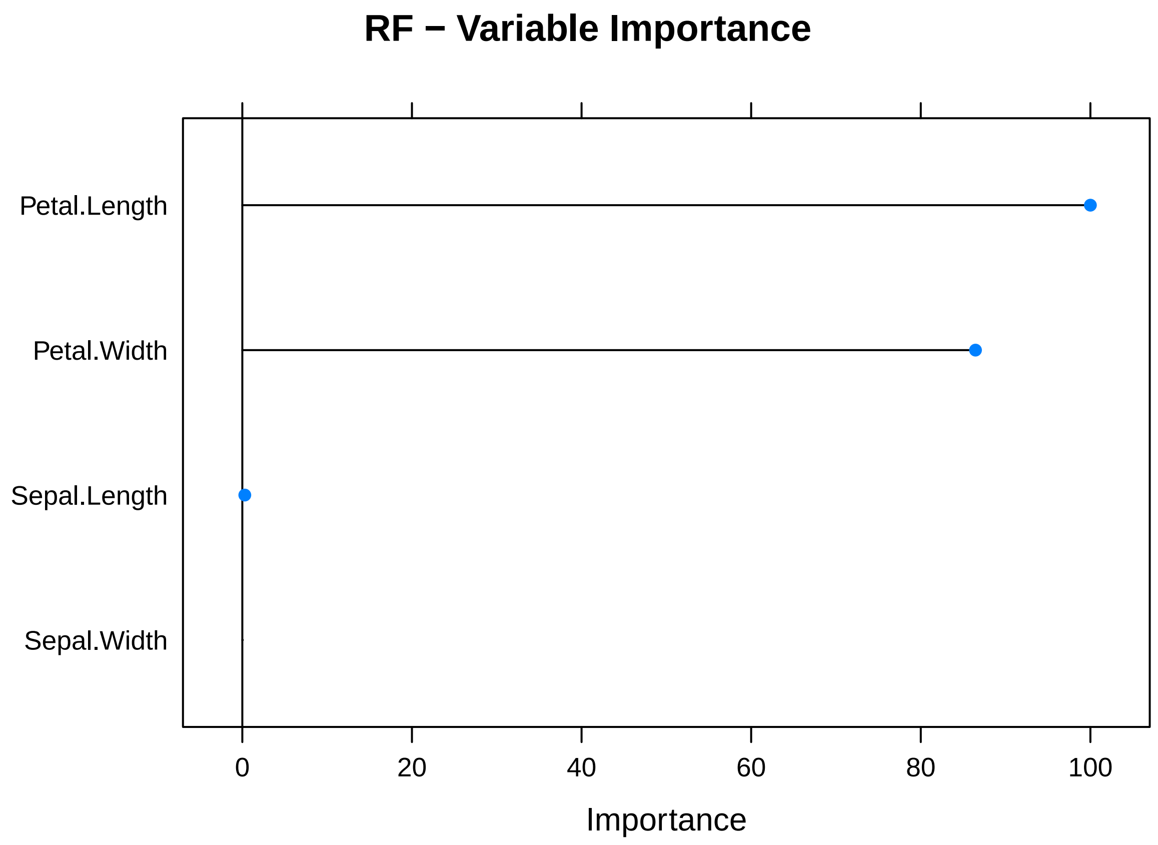


```r
plot(modFit, main="Random Forest and identifying Iris Species")
```



It is possible to examine the relative importance of each variable in constructing the eventual model. Using the Caret package it is possible to plot and visualise this

```
plot(varImp(object=modFit),main="RF - Variable Importance")
```



cc Random forests are among the most powerful classifiers available and as you can see at a very basic level they are not too difficult to develop in R. Despite their complexity which can be a problem when dealing with large data sets, they are considered white-box classifiers. Well implemented random forests are more resilient to over fitting Random forests have less variance than single decision trees.

Complexity when large ensembles are being used to analyse large data sets has been the main impediment to their use. Which makes random forests a victim of their own success. This can be time consuming and complicated process in a world where there are many alternative machine learning algorithms. Because of this random forests are used less often than decision trees in healthcare informatics(5). I many ways healthcare informatics is behind the curve in implementing this class of classifiers as decision aids.

###Adaptive Boosting or Adaboost Adaptive boosing is another machine learning meta-algorithm which has been designed based on the ensamble theory. It is used in conjunction with other classifiers such as regression anlyasis or classification and regression trees in order to improve their predicitive accuracy. The weighted output of weaker classifiers are combined to improve the accuracy of the weaker classifiers. The resultant classifier is called the boosted classifier. Adaboost classifiers are very sensitive to noisy data but resistant to overfitting. This makes preprocessing steps such as normalisation important in their design.

The following example is an example of an Adaboost classifier. The data comes from i the *Institute for Local Self Reliance* (ILSR) database. It is a non profit NGO which advocates for reform of economic systems so that they embody social and democratic values.The Wages database records the wages of employees, their demograpphics and the type of employment.

This is an example of an adaboost classifier

loading the appropriate packages

```
## Data: Wage dataset, ISLR library
library(ISLR); library(ggplot2); library(caret)
```

```
## Warning: package 'ISLR' was built under R version 4.0.2
```

```
data(Wage)
```

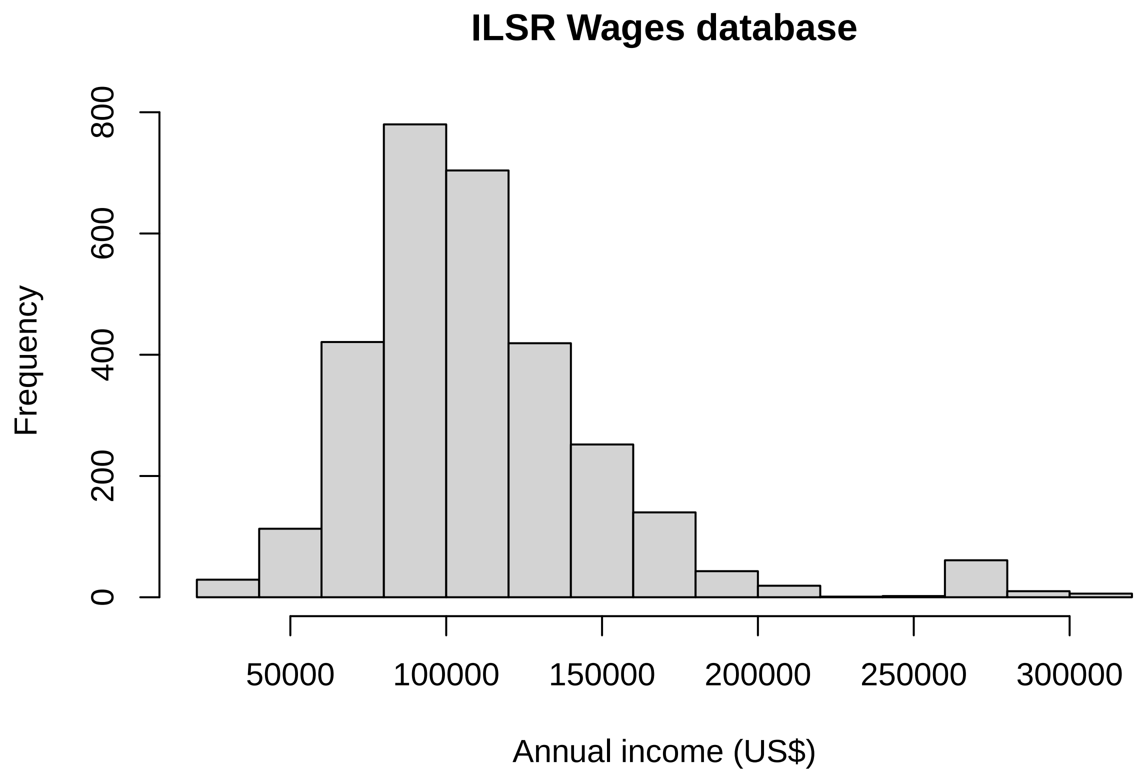
Creating training and testing databsets

As is customary the dataset is partitioned into mutually exclusive training and testing subgroups. In this project the partition is performed through random selection with 70% of dataset being used for training and 30% for testing and cross-validation.

```
inTrain<- createDataPartition(y=Wage$wage, p=0.7, list=FALSE)
training5<- Wage[inTrain,]
testing5<- Wage[-inTrain,]
```

Analysis of the Wage variable as a function of other variables

The main question which needs to be answered from the database is how income is related to socio-economic and demographic factors and the type of employment workers are engaged in. The following histogram illustrates the distribution in incomes:



```r
mean(Wage$wage)
```

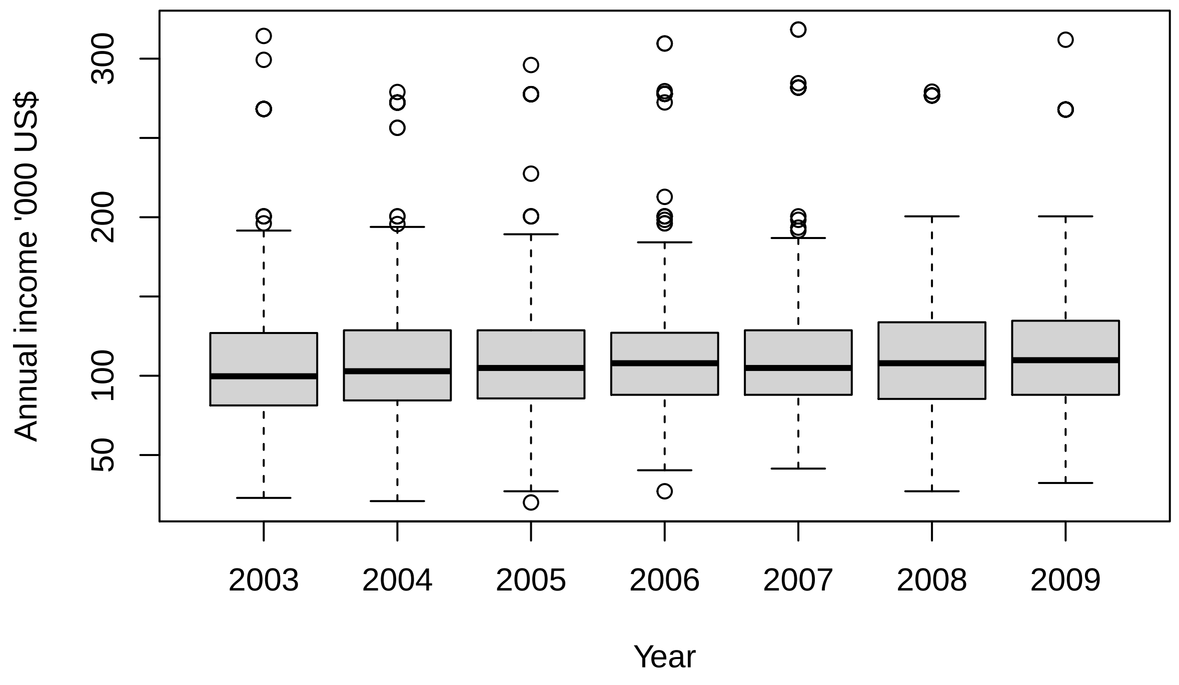
```
## [1] 111.7036
```

```r
sd(Wage$wage)
```

```
## [1] 41.7286
```

The following boxplot illustrates that annual incomes have remained unchanged during between 2003-2009.

```r
boxplot(Wage$wage~Wage$year, ylab="Annual income '000 US$", xlab="Year")
```



Linear regression analysis

In this study there are 10 potential input variables which can be used to design the adaboost machine learning algorithm. Using multivariate regression analysis it is possible to identify the variables which have no correlation with the wage variable. These variables are unlikely to contribute to the eventual model design.

In order to able to perform multivariate regression analysis, age, marital status, education, region, jobclass, health and health insurance are converted to numeric variables:

```
Wage$Marital<- as.numeric(Wage$maritl)
Wage$Race<- as.numeric(Wage$race)
Wage$Education<- as.numeric(Wage$education)
Wage$Region<- as.numeric(Wage$region)
Wage$Jobclass<- as.numeric(Wage$jobclass)
Wage$Health<- as.numeric(Wage$health)
```

In the design of the multivariate regression model wage was set as a dependent variable, jobclass as independent variable and race, marital status, education, health and region were co-variates.

```
lmWage<- lm(wage~Jobclass+Race+Education+Health+Region+Marital, data=Wage)
lmWage
```

```
##
## Call:
## lm(formula = wage ~ Jobclass + Race + Education + Health + Region +
##     Marital, data = Wage)
##
## Coefficients:
## (Intercept)     Jobclass         Race    Education       Health       Region
##      41.355        5.581       -2.741       15.317        6.697           NA
##     Marital
##       4.004
```

It is easy to see that the variable "Region" is not going to contribute to the final model design so it will be removed prior to training the adaboost model.

The model used is called thr Generalised Boost Regression GBM model. This model starts with a tree algorithm and improvs upon it through gradient boosting described above. Further detail can be obtained from *"utils::browseVignettes("gbm")"*. In this model the Wage variable is examined as a function of all the other variables apart form "Region" which is removed from the training data set (6):

```
library(gbm)
```

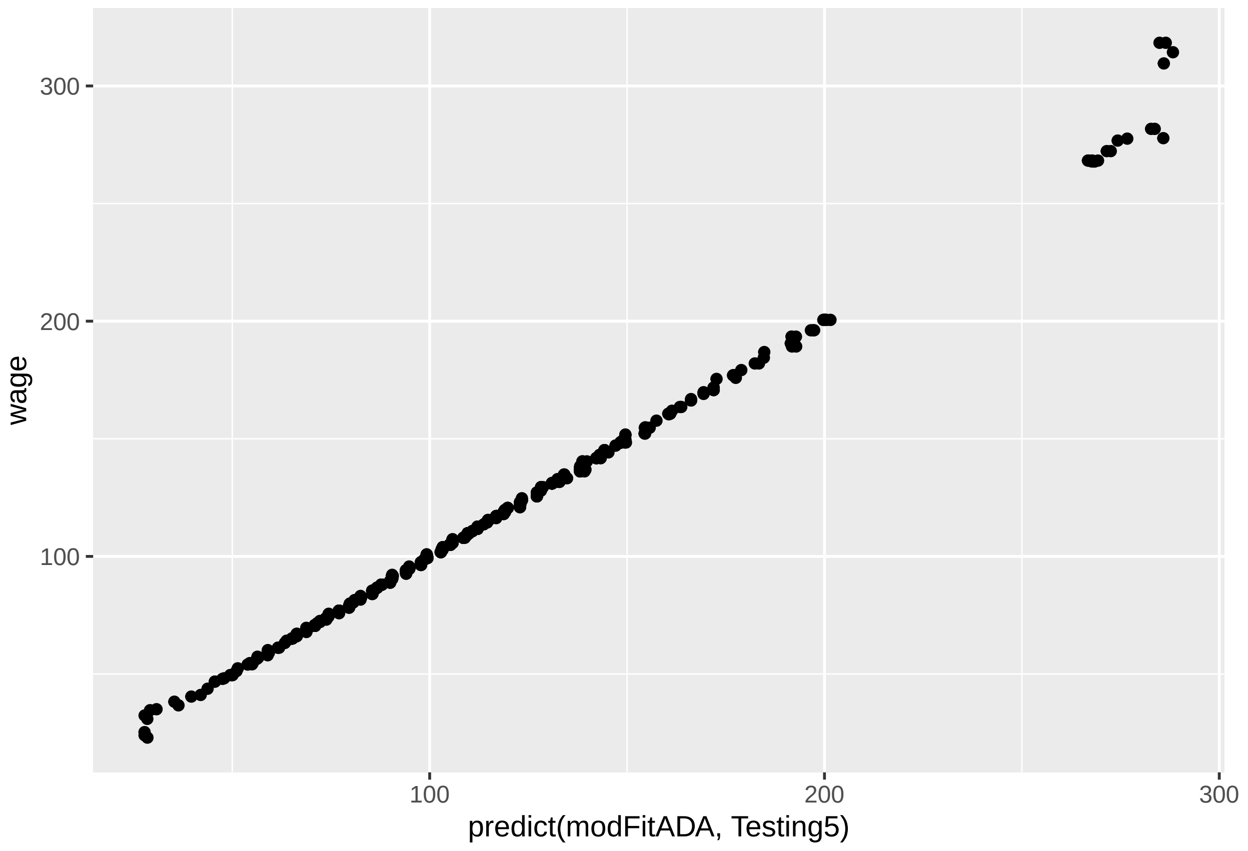
```
## Warning: package 'gbm' was built under R version 4.0.2
```

```
## Loaded gbm 2.1.5
```

```
library(ggplot2)
Training5<- subset(training5, select = -c(region))
Testing5<-subset(testing5, select=-c(region))
modFitADA<- train(wage~.,method="gbm", data=Training5, verbose=FALSE)
print(modFitADA)
```

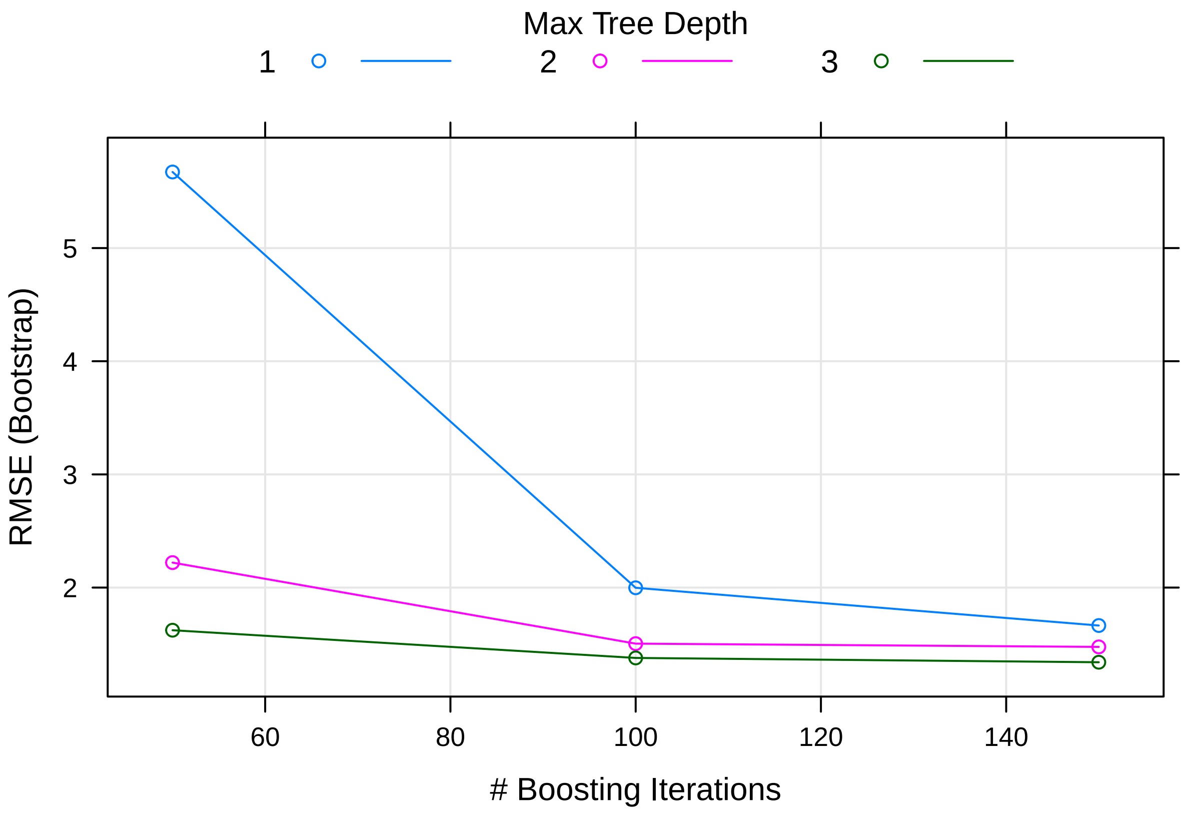
```
## Stochastic Gradient Boosting
##
## 2102 samples
##    9 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                   50      5.672277  0.9935933  3.2377747
##   1                  100      1.998088  0.9981414  1.0667749
##   1                  150      1.664680  0.9984278  0.8530828
##   2                   50      2.220607  0.9981736  1.1465292
##   2                  100      1.504602  0.9987075  0.7002029
##   2                  150      1.475950  0.9987538  0.6942292
##   3                   50      1.623264  0.9987270  0.7041470
##   3                  100      1.378170  0.9989120  0.5531844
##   3                  150      1.340194  0.9989664  0.5495373
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
qplot(predict(modFitADA, Testing5),wage,data=Testing5)
```



The final plot illustrates the progress of the ensemble (root mean squared error) through boosting iterations.

```r
plot(modFitADA)
```



Conclusions

Ensemble classifiers such as random forests and adaptive boosting algorithms are some of the most accurate classifiers in use today, both for classification and regression problems. They achieve this level of accuracy by using the collective power of imperfect "weak" classifiers. Random forests work through bagging which was described earlier whilst adaptive boosting algorithms improve the accuracy of classification through iterative improvements in the classification process. They are prone sensitive to noise and outlying variables and therefore need to be used with caution.

References

1- Lewis RJ. An introduction to classification and regression tree (CART) analysis. In Annual meeting of the society for academic emergency medicine in San Francisco, California 2000, (Vol. 14).

2- McBride OM, Mofidi R, Griffiths GD, Dawson AR, Chalmers RT, Stonebridge PA. Development of a decision tree to streamline infrainguinal vein graft surveillance. Annals of vascular surgery;36:182-9.

3- Mofidi R, McBride OM, Green BR, Gatenby T, Walker P, Milburn S. Validation of a decision tree to streamline infrainguinal vein graft surveillance. Annals of vascular surgery 2017;40:216-22.

4- Prasad AM, Iverson LR, Liaw A. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. Ecosystems. 2006;9(2):181-99.

5- Masetic Z, Subasi A. Congestive heart failure detection using random forest classifier. Computer methods and programs in biomedicine. 2016;130:54-64.

6- Freund Y, and Schapire RE (1997) “A decision-theoretic generalization of on-line learning and an application to boosting,” Journal of Computer and System Sciences, 55(1):119-139.