Decision Tree and Random Forest

Philippe Jacquet

June 7, 2018

Decision tree: We are going to build a decision tree to predict the species a given iris plant (represented by the four measurements Sepal.Width, Sepal.Length, Petal.Length, Petal.Width) belongs to. Try to go through all the steps and to understand what each step is doing. You will need the following packages:

```
install.packages("rpart")
install.packages("rpart.plot")
```

Load the iris dataset and split it into a training set and test set:

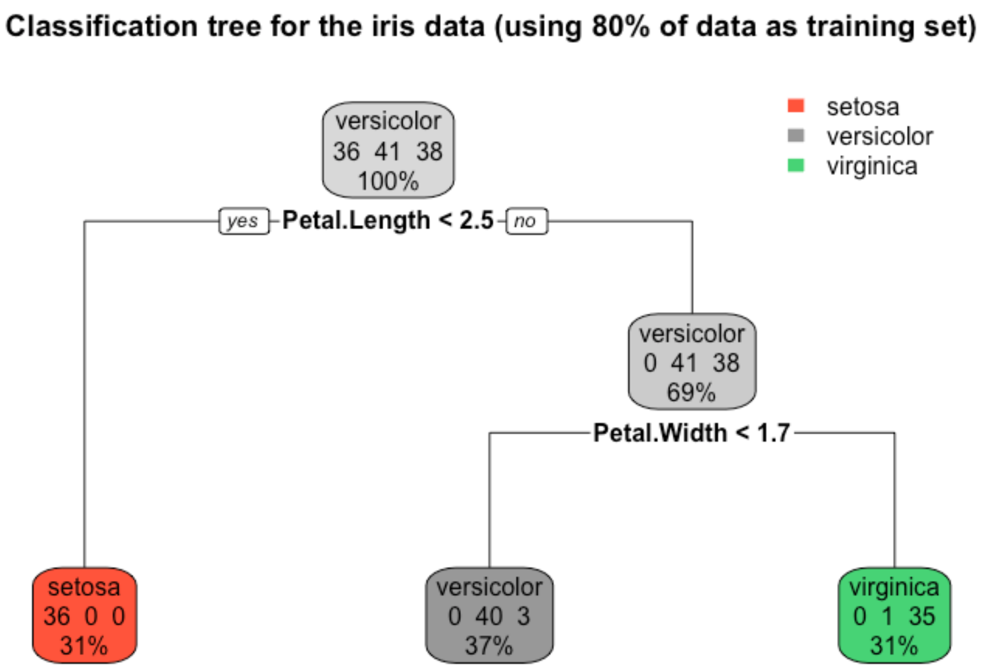
```
data(iris)

set.seed(2)
ind=sample(2,nrow(iris),replace=TRUE,prob=c(0.80,0.20

iris.training=iris[ind==1,]
iris.test=iris[ind==2,]
```

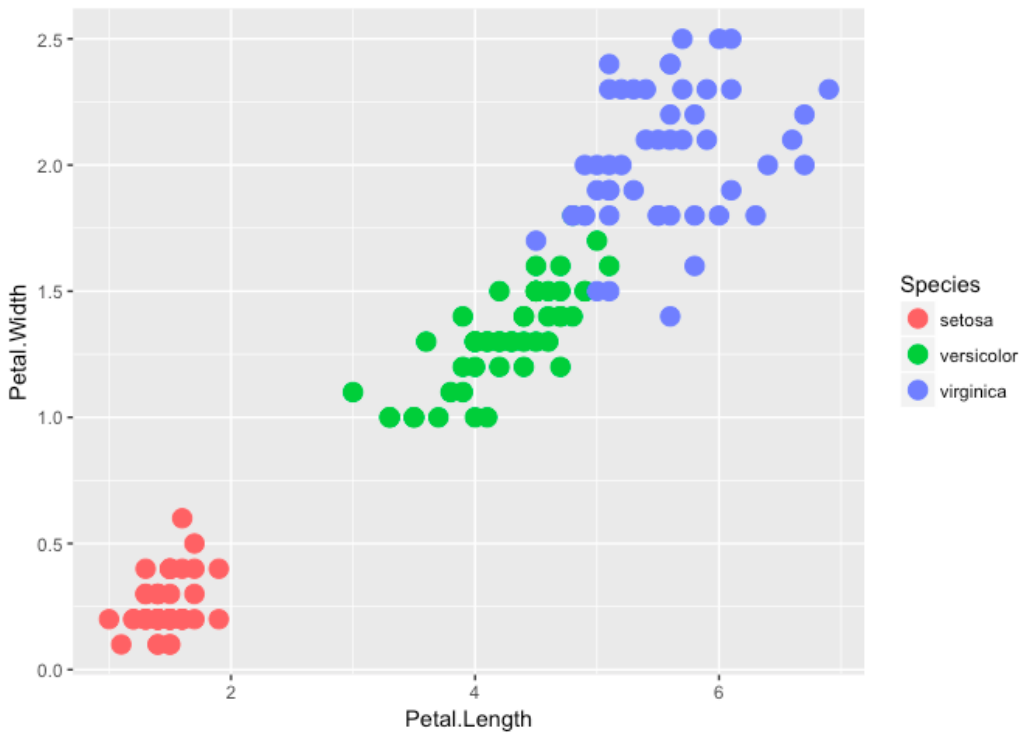
Make the decision tree model using the entropy as the impurity index:

```
library(rpart)
library(rpart.plot)
tree=rpart(data=iris.training,Species~Sepal.Width+Sep
rpart.plot(tree,main="Classification tree for the iri
```



We see that the first two splits of the decision tree use Petal.Length and Petal.Width. We may want to look again at the exploratory analysis that we have done before building the neural network. This may help you to understand better the iris tree structure. At least let us plot Petal.Width versus Petal.Length:

```
#install.packages("ggplot2")
library(ggplot2)
qplot(Petal.Length, Petal.Width, data=iris, colour=Sp
```



We clearly see the three groups (setosa, versicolor, virginica) are well separated by using only Petal.Length and Petal.Width.

Predict the species for the "iris.training" dataset:

```
predictions=predict(tree,newdata=iris.training,type="
actuals=iris.training$Species
table(actuals,predictions)
```

```
##             predictions
## actuals       setosa versicolor virginica
##   setosa          36          0         0
##   versicolor       0         40         1
##   virginica        0          3        35
```

Predict the species for the "iris.test" dataset:

```
predictions=predict(tree,newdata=iris.test,type="clas
actuals=iris.test$Species
confusion.matrix=table(actuals,predictions)
```

```
print(confusion.matrix)
```

```
##             predictions
## actuals      setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0          9         0
##   virginica       0          2        10
```

This confusion matrix shows that the decision tree has made 2 mistakes in 35 preditions. The test sets used for the neural network and the decision tree are the same, so we may compare their performances.

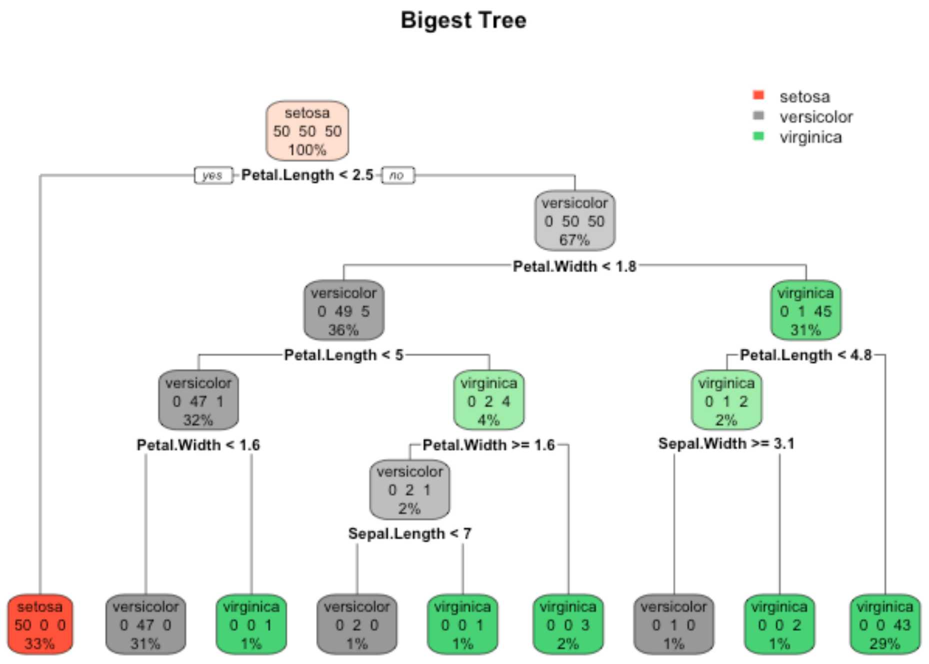
Let us finally compute the accuracy of the tree:

```
accuracy=sum(diag(confusion.matrix))/sum(confusion.ma
print(accuracy)
```

```
## [1] 0.9428571
```

Above we used (minsplit=10,minbucket=5) as stopping rules. If you want to use the pruning method instead to find the best tree:

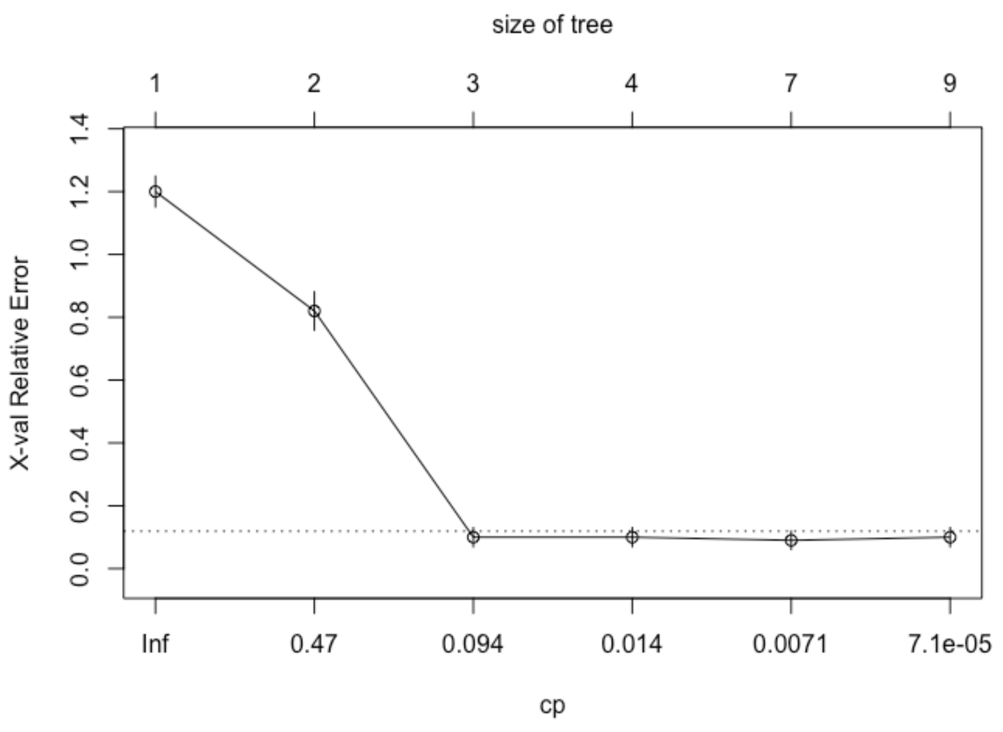
```
tree=rpart(data=iris,Species~Sepal.Width+Sepal.Length
rpart.plot(tree,main="Bigest Tree",extra=101)
```



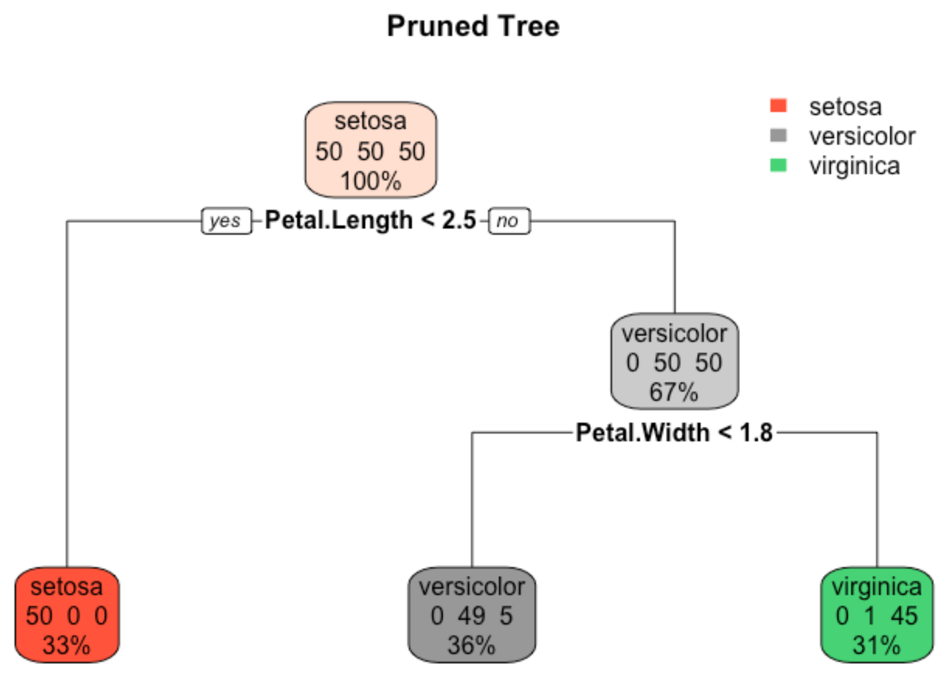
```r
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = Species ~ Sepal.Width + Sepal.Leng
##     Petal.Width, data = iris, method = "class", pa
##     control = rpart.control(minsplit = 1, minbucke
##
## Variables actually used in tree construction:
## [1] Petal.Length Petal.Width  Sepal.Length Sepal.W
##
## Root node error: 100/150 = 0.66667
##
## n= 150
##
##         CP nsplit rel error xerror     xstd
## 1 5.0e-01      0      1.00   1.20 0.048990
## 2 4.4e-01      1      0.50   0.82 0.060970
## 3 2.0e-02      2      0.06   0.10 0.030551
## 4 1.0e-02      3      0.04   0.10 0.030551
## 5 5.0e-03      6      0.01   0.09 0.029086
## 6 1.0e-06      8      0.00   0.10 0.030551
```

```
plotcp(tree)
```



```
ptree=prune(tree,cp=2.0e-02)
rpart.plot(ptree,main="Pruned Tree",extra=101)
```



Questions: Play around with different hyperparameter values [split (“gini”), minsplit, minbucket, Species (Petal.Length+Petal.Width)] in the hope that your model will perform better. Use the test set as a validation set.

Random forest: We are going to build a random forest to predict the species a given iris plant belongs to. Try to go through all the steps and to understand what each step is doing. You will need the following package:

```
install.packages("randomForest")
```

Load the package:

```
library(randomForest)
```

Build the random forest model (using the Gini index):

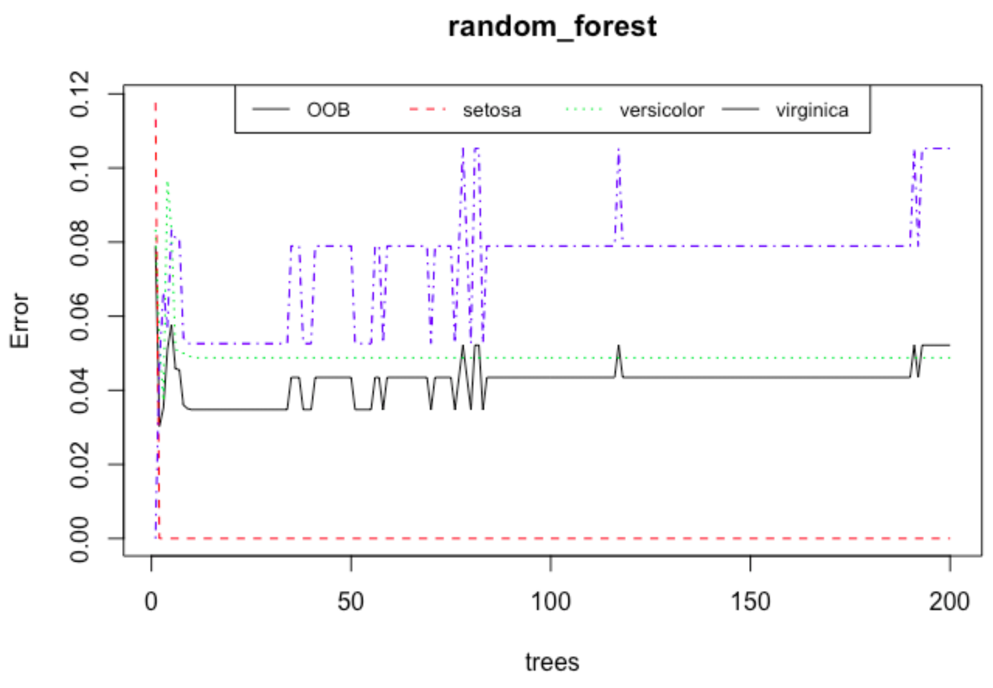
```
random_forest=randomForest(data=iris.training,Species
print(random_forest)
```

```
##
## Call:
##  randomForest(formula = Species ~ Sepal.Width + Se
##                Type of random forest: classificati
##                      Number of trees: 200
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 5.22%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         36          0         0   0.00000000
## versicolor      0         39         2   0.04878049
## virginica       0          4        34   0.10526316
```

The above confusion matrix and the out-of-bag (OOB) error rate [6/115*100=5.22%] are obtained as follows: for each plant in the training dataset, make a random forest prediction using only the trees that did not use that particular plant in their bootstrap training subset. The random forest model is thus used to predict the data NOT drawn (the "out-of-bag" sample).

Let us plot the misclassification error rate as a function of the number of trees used in the random forest:

```
plot(random_forest)
legend("top",cex=0.8,legend=colnames(random_forest$er
```



The x-axis is the "number of trees" and the y-axis is the "misclassification error rate". The black solid line represents the overall OOB error. The colour lines are the class errors (one for each species: red=Setosa, green=Versicolor and blue=Virginica). We see that we should use about 25 trees:

```
random_forest=randomForest(data=iris.training,Species
```

Predict the species for the "iris.training" dataset (using all trees in the random forest):

```
predictions=predict(random_forest,newdata=iris.traini
actuals=iris.training$Species
table(actuals,predictions)
```

```
##             predictions
## actuals      setosa versicolor virginica
##   setosa         36          0         0
##   versicolor      0         41         0
##    virginica       0           0         38
```

We see 100% accuracy: the random forest has perfecly learned the training data.

Predict the species for the “iris.test” dataset:

```
predictions=predict(random_forest,newdata=iris.test,t
actuals=iris.test$Species
confusion.matrix=table(actuals,predictions)
print(confusion.matrix)
```

```
##             predictions
## actuals      setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0          8         1
##   virginica       0          2        10
```

Finally the accuracy:

```
accuracy=sum(diag(confusion.matrix))/sum(confusion.ma
print(accuracy)
```

```
## [1] 0.9142857
```

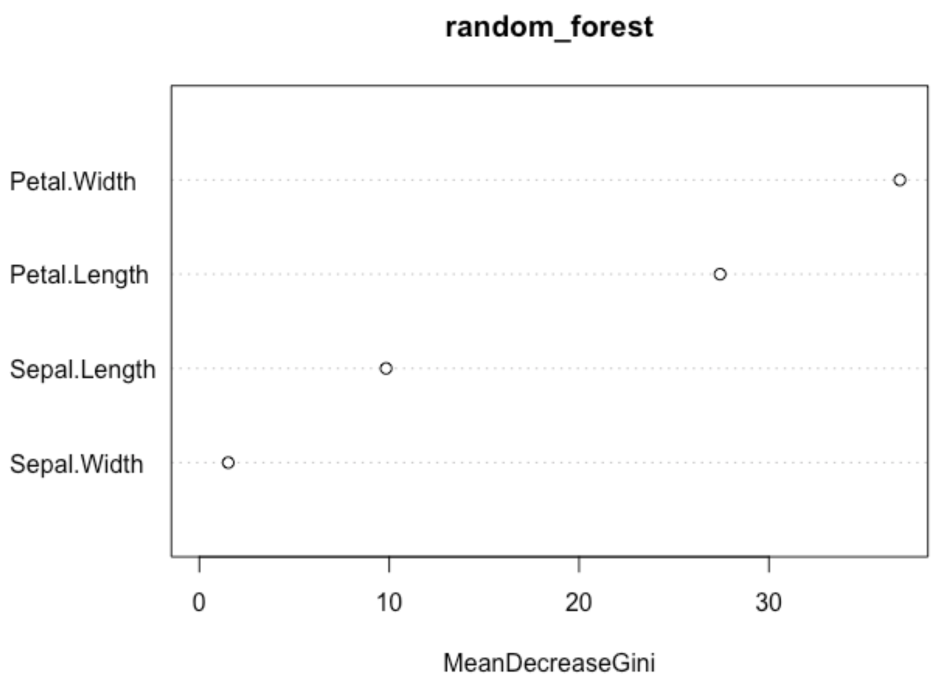
We see that the accuracy of the random forest is slighly lower than that of the decision tree. This may happen when the number of feature variables (4 for the iris data) is small.

Compute and plot the importance (of all the variables):

```
importance(random_forest)
```

```
##              MeanDecreaseGini
## Sepal.Width          1.522782
## Sepal.Length         9.836576
## Petal.Length        27.427145
## Petal.Width         36.896801
```

```
varImpPlot(random_forest)
```



As in the decision tree, we see that the Petal.Length and Petal.Width are the most important predictors.

Questions: Play around with different parameter values (impurity ('entropy'), ntree, replace (FALSE), sampsize (size of sample to draw; default: N if replace=TRUE, 0.63*N otherwise), mtry (the number of variables randomly sampled as candidates at each split; default: sqrt(4)=2)] in the hope that your model will perform better.