Decision Trees and Random Forest

This shows how to build predictive models with packages “party”, “rpart” and “randomForest”. It starts with building decision trees with package”party” and using the built tree for classification, followed by another way to build decision trees with package “rpart”.

After that, it represents an example on training a random forest model with package randomForest.

Random Forest:

The package **randomForest** is used below to build a predictive model for the “iris” data. There are two limitations with function **randomForest.**

1. It cannot handle data with missing values, and users have to impute data before feeding them into function.
2. There is a limit of 32 to the maximum number of levels of each categorical attribute. Attributes with more than 32 levels have to be transformed first before using **randomForest()**

An alternative way to build a random forest is to use function **cforest()** from package **party,** which is not limited to the above maximum levels. However categorical variables with more levels will make it require more memory and take longer time to build a random forest.

The iris data is first split into two subsets:

1. Training (70%)
2. Test (30%)

```
> ind<-sample(2, nrow(iris), replace= TRUE, prob=c(0.7,0.3))
> trainData<-iris[ind==1,]
> testData<-iris[ind==2,]
> 
```

Loading the package “randomForest”:

After loading the package , you need to train the random forest. In the code below , the formula is set to “**Species ~** .” , which means to predict **Species** with all other variables in the data.

```
> library(randomForest)
randomForest 4.6-7
Type rfNews() to see new features/changes/bug fixes.
Warning message:
package ‘randomForest’ was built under R version 2.15.3
> 
```

```
> rf<- randomForest(Species ~ .,data=trainData, ntree=100, proximity=TRUE)
> table(predict(rf), trainData$Species)

             setosa versicolor virginica
  setosa         37          0         0
  versicolor      0         26         3
  virginica       0          3        32
>
```

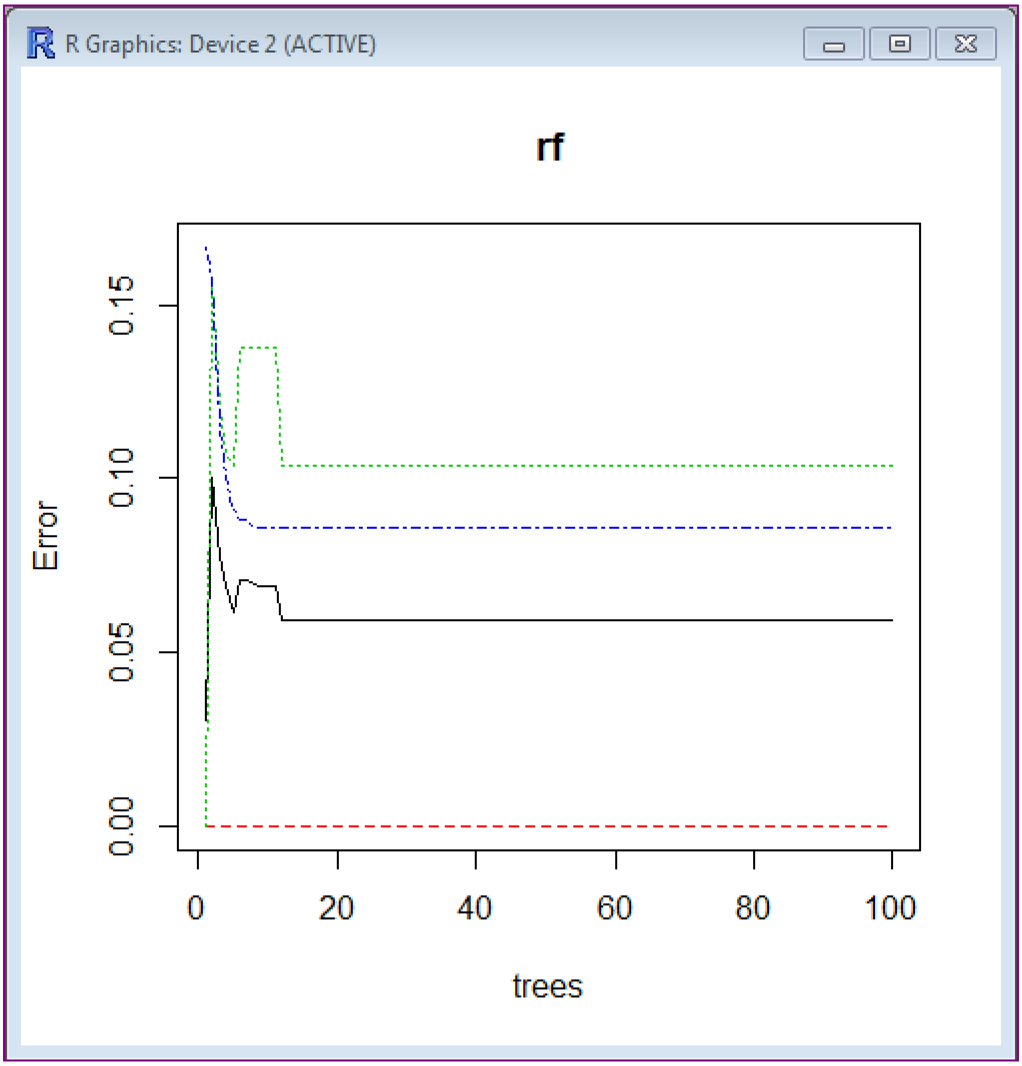
```
> print(rf)

Call:
 randomForest(formula = Species ~ ., data = trainData, ntree = 100,      proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 5.94%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         37          0         0  0.00000000
versicolor      0         26         3  0.10344828
virginica       0          3        32  0.08571429
>
```

```
> attributes(rf)
$names
 [1] "call"            "type"            "predicted"       "err.rate"
 [5] "confusion"       "votes"           "oob.times"       "classes"
 [9] "importance"      "importanceSD"    "localImportance" "proximity"
[13] "ntree"           "mtry"            "forest"          "y"
[17] "test"            "inbag"           "terms"

$class
[1] "randomForest.formula" "randomForest"

>
```

Plot : Error Rate of Random Forest

We plot the error rates with various number of trees.

```
> plot(rf)
```



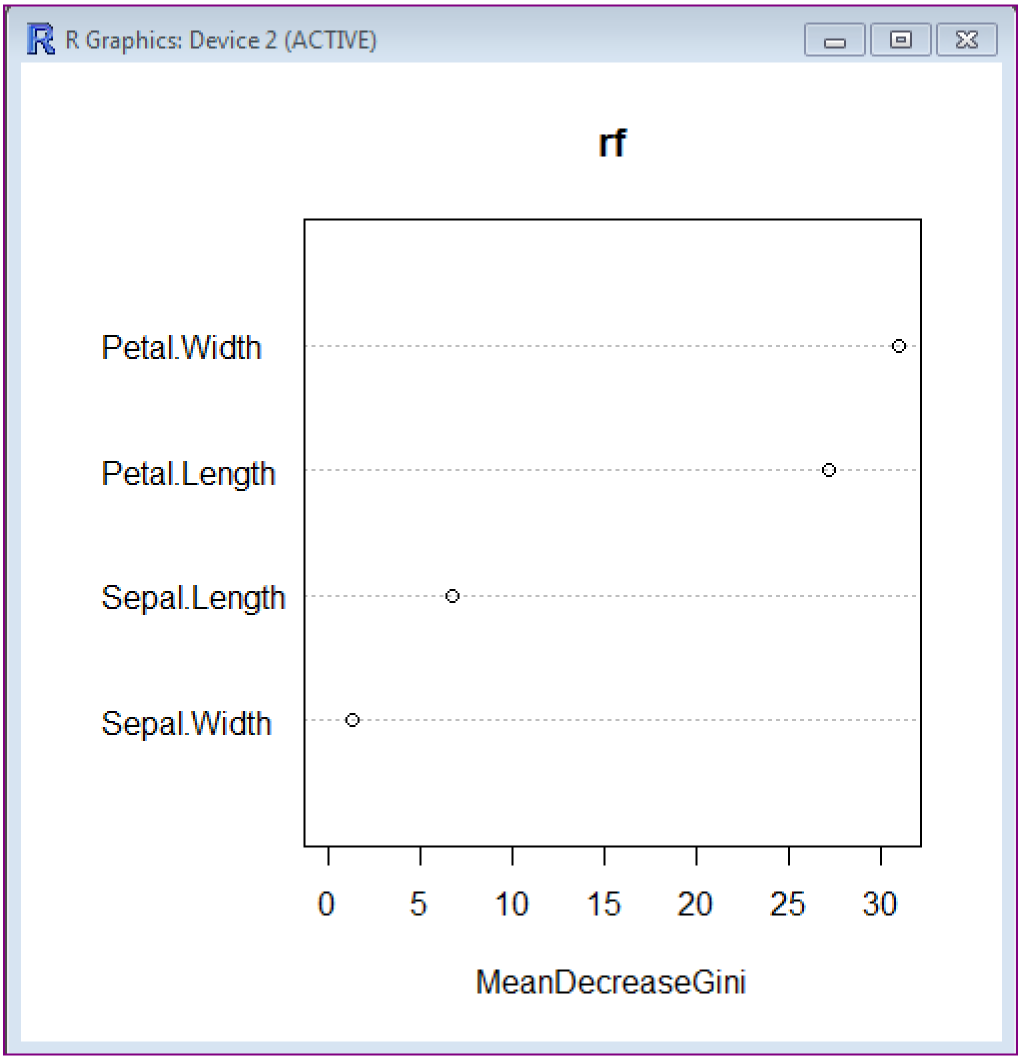
The importance of variables can be obtained with functions:

- **Importance()**
- **varImpPlot()**

```
> importance(rf)
             MeanDecreaseGini
Sepal.Length         6.746604
Sepal.Width          1.374903
Petal.Length        27.162735
Petal.Width         30.920526
> 
```

Plot : Variable Importance

```
> varImpPlot(rf)
```



Finally, the built random forest is tested on test data, and the result is checked with functions **table()** and **margin().** The margin of a data point is as the proportion of votes for the correct class minus maximum proportion of votes for other classes. Also, Positive margin means correct classification.

```
> irisPred <- predict(rf, newdata=testData)
> table(irisPred, testData$Species)

irisPred     setosa versicolor virginica
  setosa         13          0         0
  versicolor      0         20         3
  virginica       0          1        12
> 
```

Plot : Margin of Predictions

```
> plot(margin(rf, testData$Species))
Loading required package: RColorBrewer
```

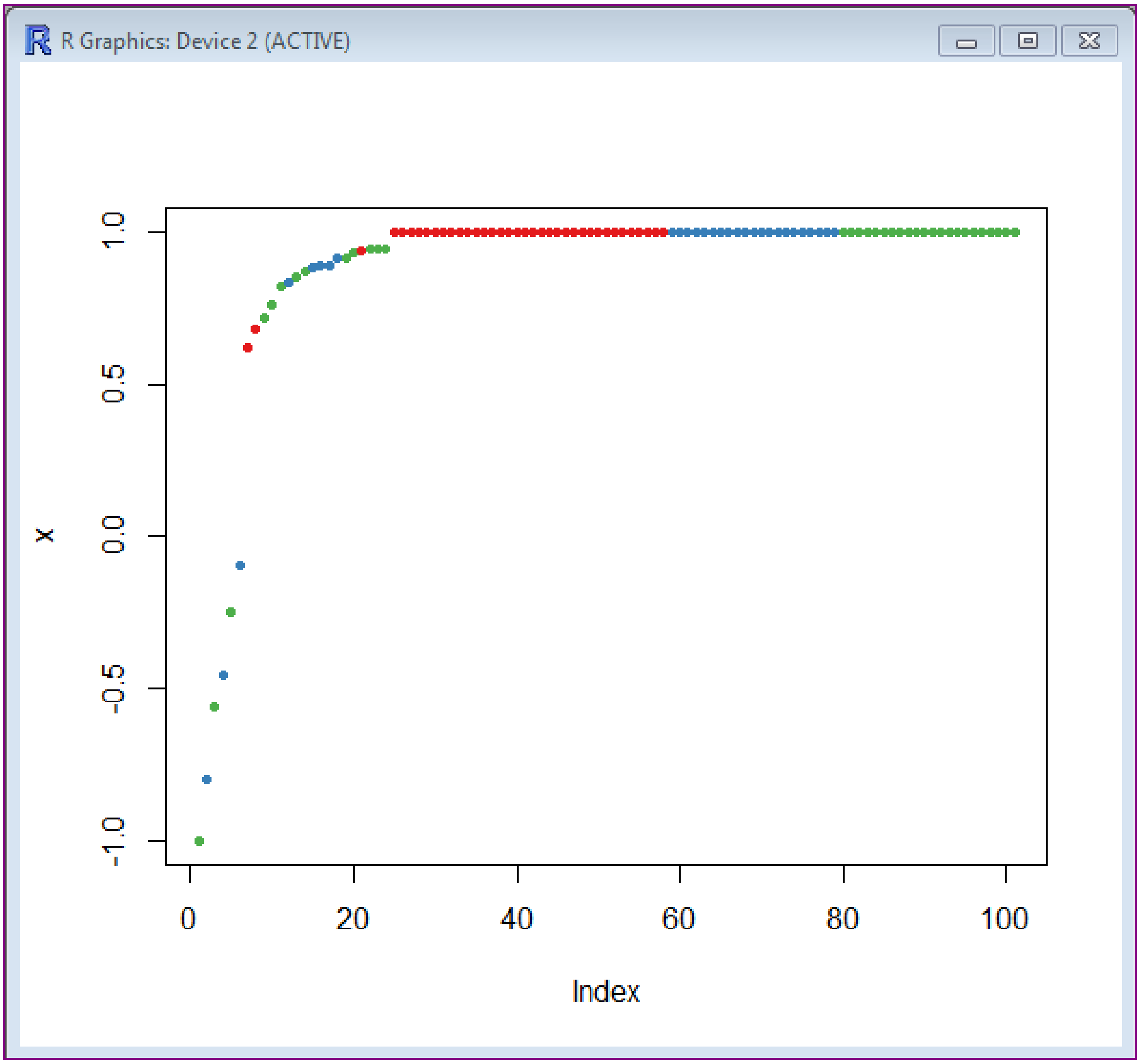


Fig: Margin of Predictions