Machine Learning

Ravi Kumar Tiwari

14 June 2016

Introduction

1. Definition: It is a method of teaching computers to make predictions based on data
2. Types of machine learning:
    - Supervised learning
    - Unsupervised learning
3. Machine Learning in Everday life:
    - Forecasting
    - Spam filtering
    - Product recommendation
    - Fraud detection

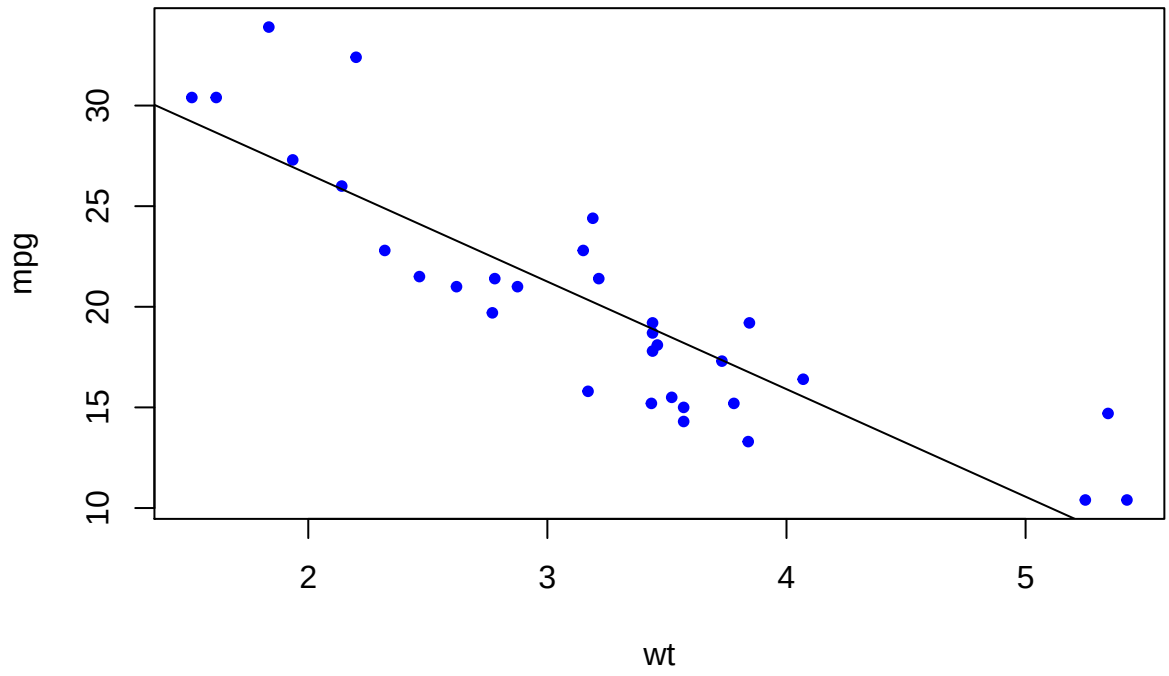
Linear Regression

Working principle

Find a straight line that best describes the relationship between the dependent and the independent variables. In order to obtain the predicted value of the dependent variable, plug in the the values of the independent variable in the equation of the line.

Example

Build a linear model to describe the relationship between mpg (miles per gallon) and wt (weight of the car) in the mtcars dataset



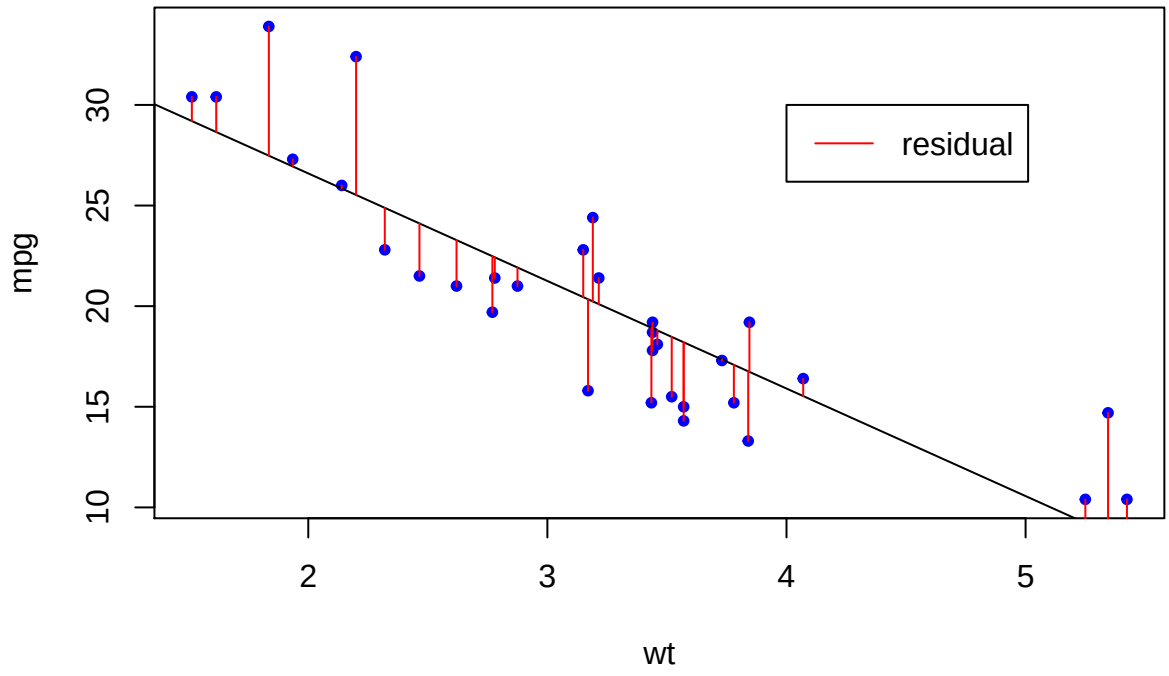
Codes

```
## Build the linear model object
lmModel <- lm(mpg ~ wt, data = mtcars)

## Obtain the model parameters
sumModel <- summary(lmModel)
sumModel$coefficients

## Prediction using the model
```

```
predValue <- predict(lmModel, data.frame(wt = c(3.5, 4.2)))
predValue <- predict(lmModel, data.frame(wt = mtcars$wt))
```

Model Assessment

1. Visual Inspection



2. R-squared value

```
sumModel <- summary(lmModel)
sumModel$r.squared
```

```
## [1] 0.7528328
```

3. F-statistics

```
sumModel$fstatistic
```

```
##    value    numdf    dendf
## 91.37533  1.00000 30.00000
```

Extension of linear model

```
## More than one predictors
lmModel2 <- lm(mpg ~ wt+hp+disp, data = mtcars) # wt, hp, and disp will be used as predictor
lmModel3  <- lm(mpg~ ., data = mtcars)   # All the variable will be used

## subset selection: 1) Identify the best model that contains a given number of predictors
## 2) Identify the overall best model

library(leaps)  # subset selection library
fwdSelection <- regsubsets(mpg ~ ., data = mtcars, method = "forward")
sumFwdSel <- summary(fwdSelection)
sumFwdSel$outmat  # 1) Included predictor in the Best Model when the number of predictors is fixed
which.max(sumFwdSel$adjr2) # 2) overall best model has the highest adjusted r-squared value
```

Output

1. Included predictors in the best model when the number of predictors are fixed

```
sumFwdSel$outmat
```

```
##          cyl disp hp  drat wt  qsec vs  am  gear carb
## 1  ( 1 ) " " " "  " " " " "*" " "  " " " " " " " "
## 2  ( 1 ) "*" " "  " " " " "*" " "  " " " " " " " "
## 3  ( 1 ) "*" " "  "*" " " "*" " "  " " " " " " " "
## 4  ( 1 ) "*" " "  "*" " " "*" " "  " " "*" " " " "
## 5  ( 1 ) "*" " "  "*" " " "*" "*"  " " "*" " " " "
## 6  ( 1 ) "*" "*"  "*" " " "*" "*"  " " "*" " " " "
## 7  ( 1 ) "*" "*"  "*" "*" "*" "*"  " " "*" " " " "
## 8  ( 1 ) "*" "*"  "*" "*" "*" "*"  " " "*" "*" " "
```

2. Overall best model

```
n <- which.max(sumFwdSel$adjr2)
coef(fwdSelection, n)
```

```
## (Intercept)         cyl        disp          hp          wt        qsec
## 20.05169952 -0.50206577  0.01396099 -0.01956054 -3.99773180  0.81017782
##          am
##  2.94074955
```

Challenge

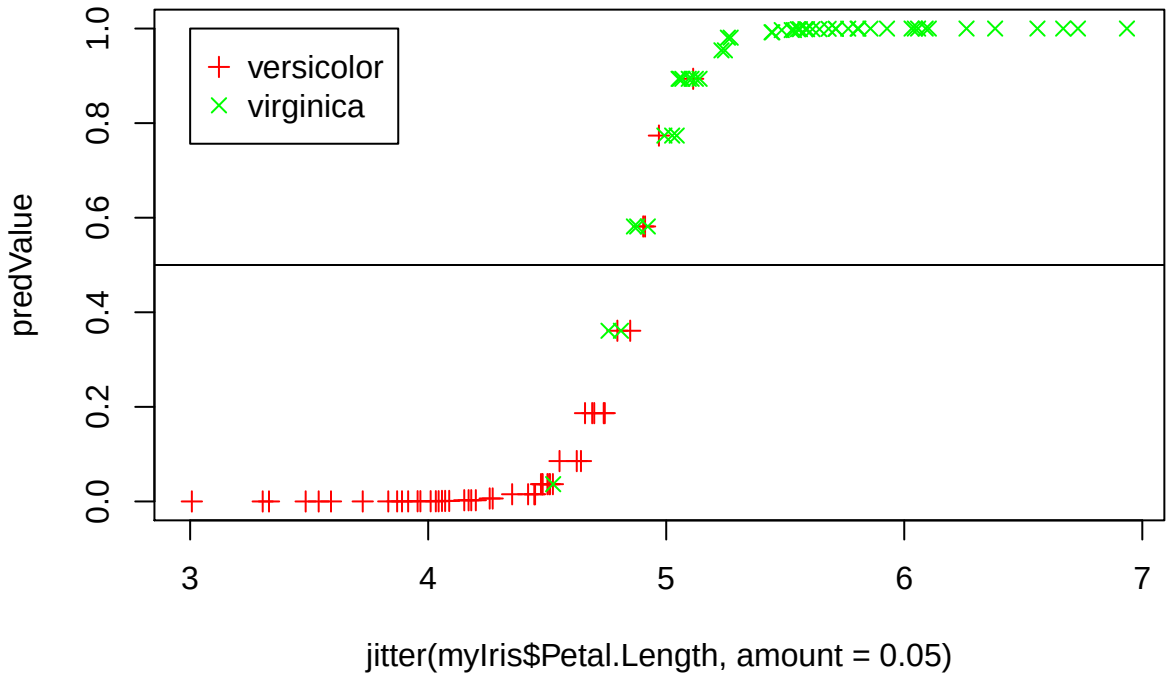
Use backward selection model to find the best model for mpg

Logistic Regression

Working Principle

Fit the predictor values to a function whose value lies between 0 and 1. Choose a cut-off value to separates the function output values in two regions corresponding to two classes. A new observation class is decided by the region in which the function values corresponding to this observation lies.

Example



Codes

```
inSetosa <- iris$Species == "setosa"
myIris <- iris[!inSetosa,]
myIris$Species <- factor(myIris$Species, levels = c("versicolor", "virginica"))
glmModel <- glm(Species ~ Petal.Length, data = myIris, family = binomial(link="logit"))
predValue <- predict(glmModel, myIris, type = "response")
```

Model Assessment

```r
prediction <- ifelse(predValue > 0.5, "virginica", "versicolor")
table(prediction, myIris$Species)
```

```
##
## prediction   versicolor virginica
##   versicolor         46         3
##   virginica           4        47
```

Tree based algorithm

Used both for classification and regression

Working principle

Divide the data set into several small regions such that the response variables are (nearly) homogeneous in those regions. The predictd value of a new observation is the most dominant class of the region to which the observation belongs.

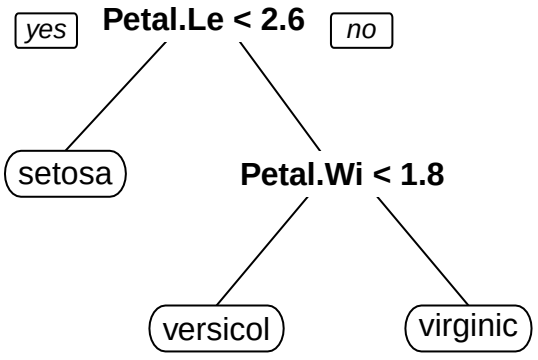
Example

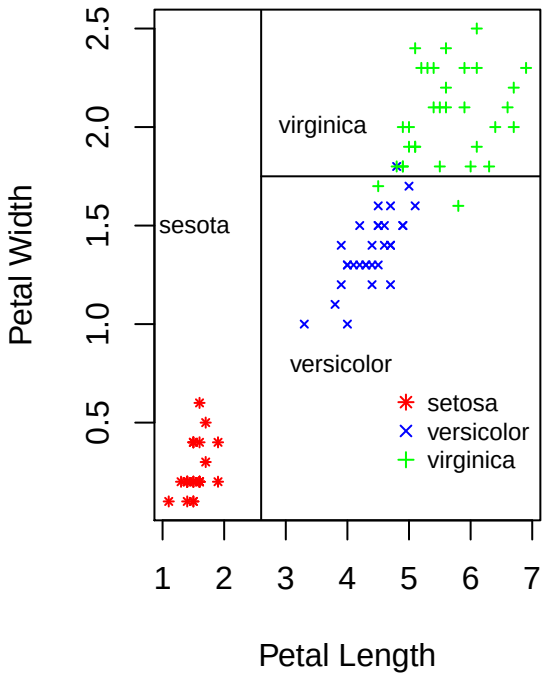
Find the decision rule to predict the species of iris dataset based on Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width

```
iris[c(1,100,150),]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 100          5.7         2.8          4.1         1.3 versicolor
## 150          5.9         3.0          5.1         1.8  virginica
```

Decision Tree visualization

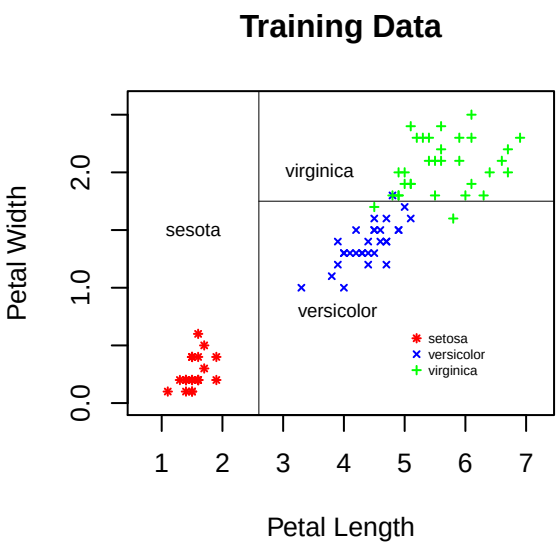


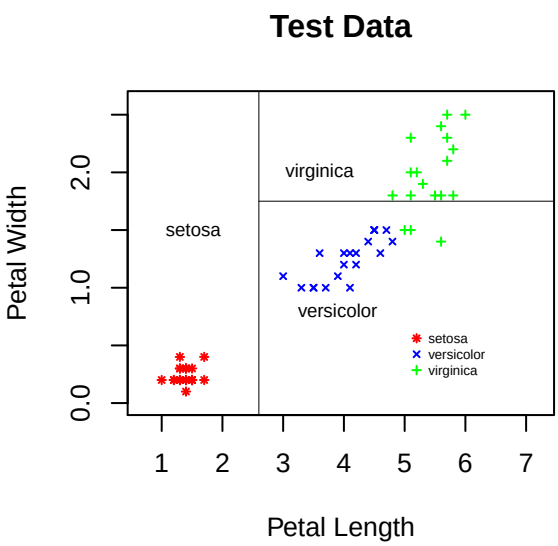


Codes

```
## Load the required libraries
library(rpart)
library(rpart.plot)  # For decision tree visualization

## create the data partition
set.seed(1)
inTrain <- sample(c(TRUE, FALSE), size = nrow(iris), replace = TRUE, prob = c(0.6,0.4))
trainData <- iris[inTrain,]
testData <- iris[!inTrain,1:4]
testClass <- iris[!inTrain,5]

## Create the tree model
treeModel <- rpart(Species ~ ., data = trainData)

## Use the tree model to predict the class of the test data
predTrainClass <- predict(treeModel, newdata = trainData, type = "class")
predTestClass <- predict(treeModel, newdata = testData, type = "class")

## Find out the performance of the decision tree
table(predTrainClass, trainData$Species)  # Confusion Matrix
mean(predTrainClass == trainData$Species) # Prediction Accuracy

table(predTestClass, testClass)           # Confusion Matrix
mean(predTestClass == testClass)          # Prediction Accuracy
```

Decision tree prediction visualization





	setosa	versicolor	virginica
setosa	27	0	0
versicolor	0	29	2
virginica	0	1	30

	setosa	versicolor	virginica
setosa	23	0	0
versicolor	0	20	3
virginica	0	0	15

Add some challenge

Advantages of decision tree

Easy to interpret

Problem with the decision tree

Lower prediction accuracy

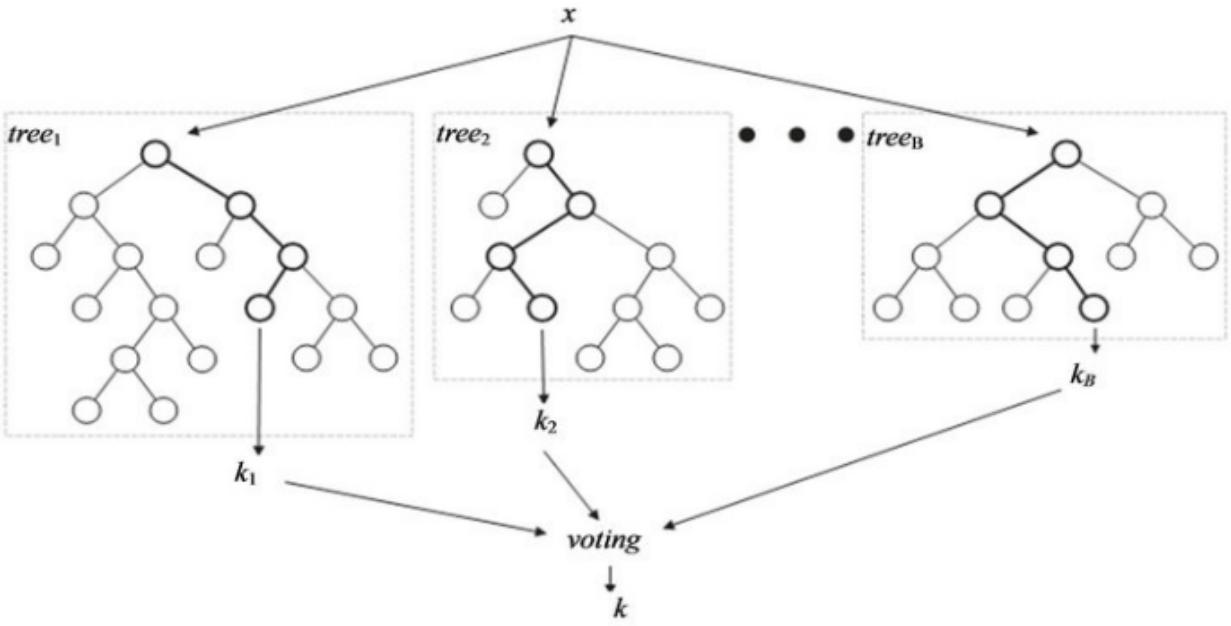
Solution

Aggregate many decision trees (bagging, random forest, boosting)

random Forest

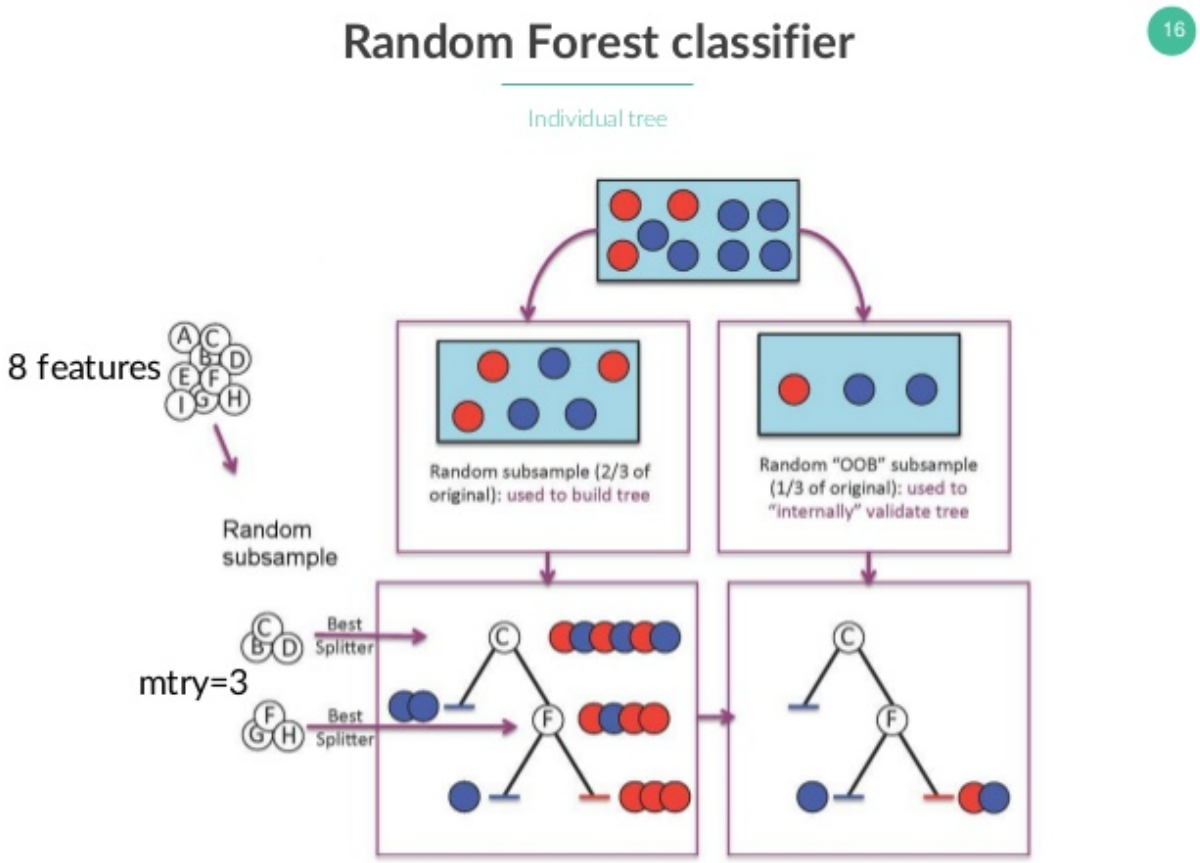
Need to decorrelate the trees. Making it more accurate

ntree



mtry

Decorrelate the trees a random sample of m predictors is chosen as split candidates from the full set of p predictors.

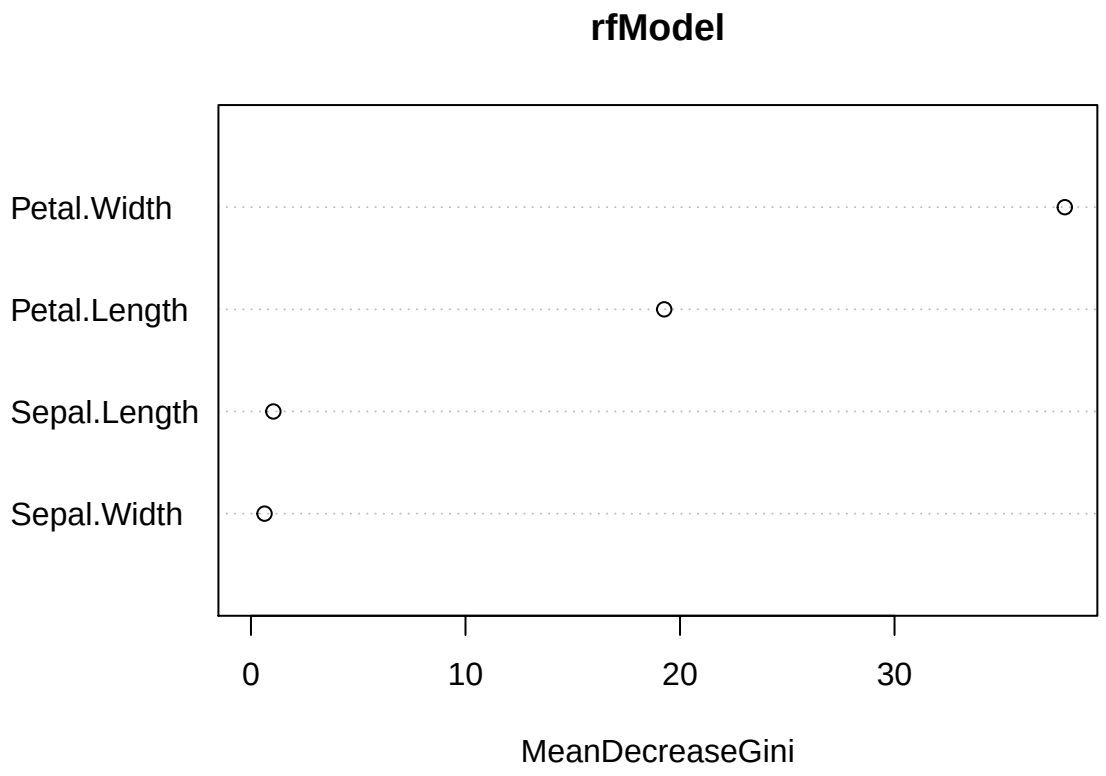


random forest example

```
library(randomForest)
rfModel <- randomForest(Species ~ ., data=trainData, mtry=3, ntree=15)
predClass <- predict(rfModel, newdata = testData)
table(predClass, testClass)
```

```
##             testClass
## predClass    setosa versicolor virginica
##   setosa         23          0         0
##   versicolor      0         20         3
##   virginica       0          0        15
```

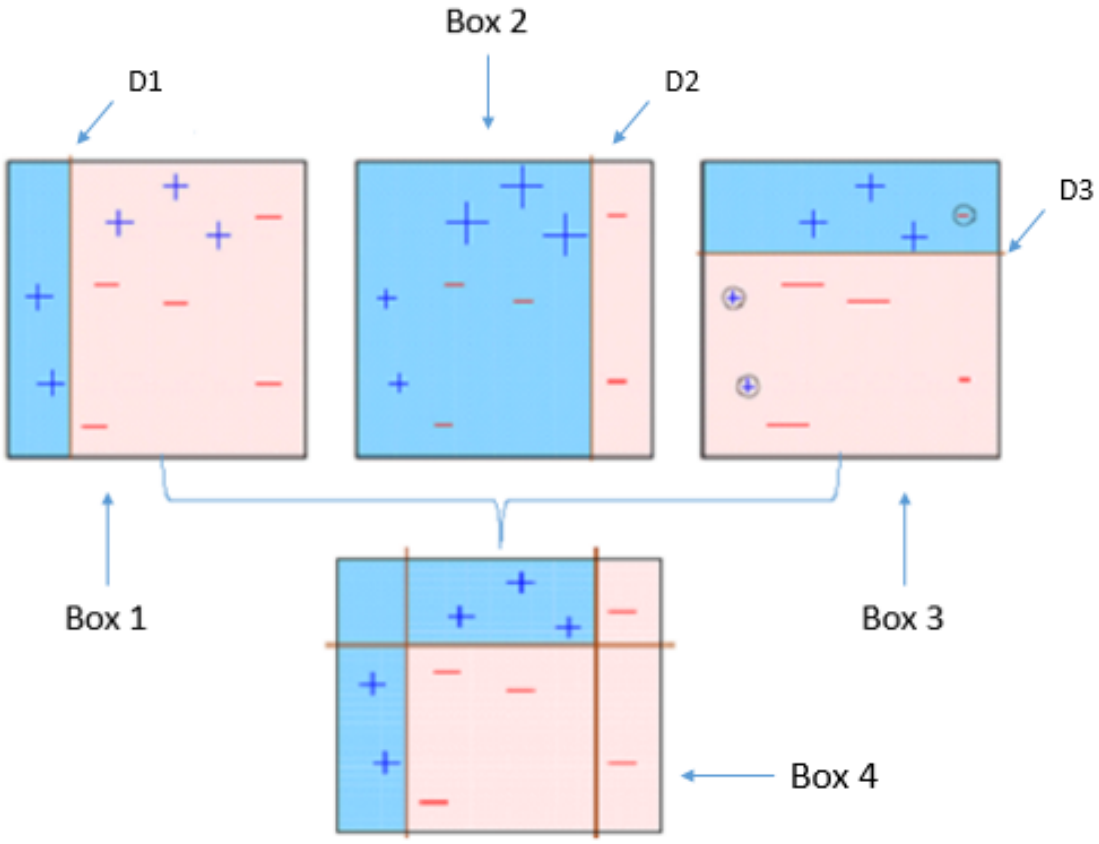
```r
varImpPlot(rfModel)
```



Add some challenge

Boosting

Illustration



example

```
#gbmModel <- gbm(Species ~ ., data = trainData, distribution = "multinomial",
#                n.trees = 20)
#predict(gbmModel, newdata = testData, single.tree = TRUE, n.trees = 20, type = "response")


# boost.boston=gbm(medv~.,data=Boston[train,],distribution= # #"gaussian",n.trees=5000,interaction.dept
```

The argument n.teees = 5000 indicates that we want 5000 trees, and the option interaction.depth = 4 limits the depth of each tree

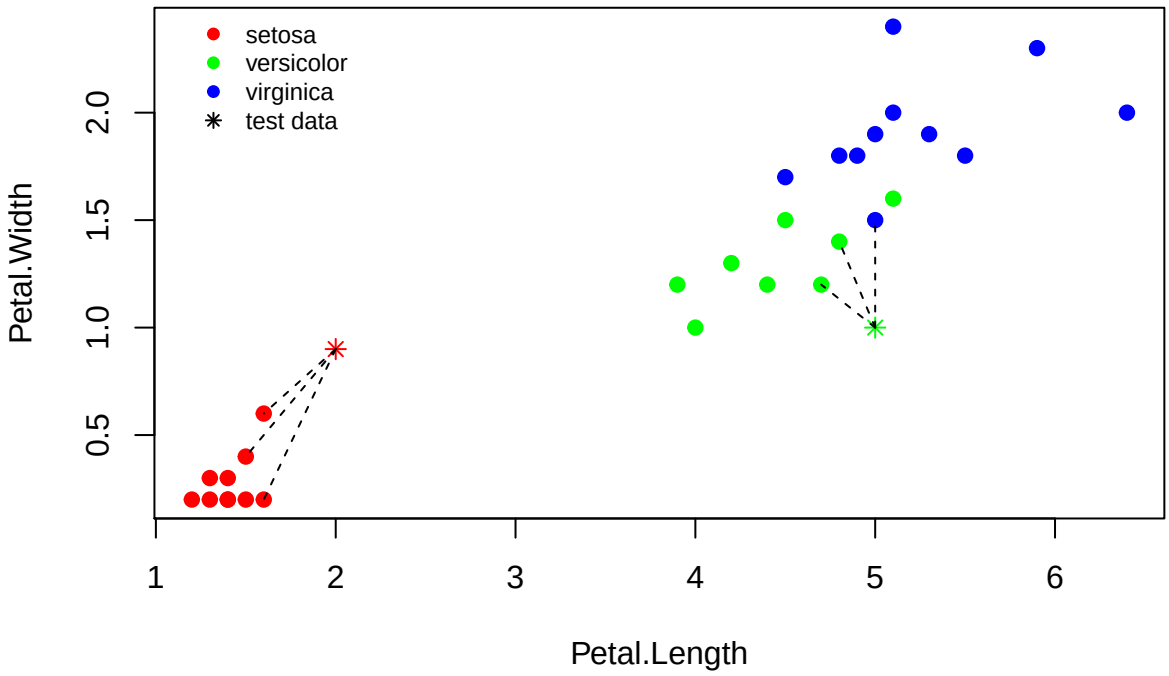
knn

make sure, you scale the data and also with an example tell why it is important to scale the data

Working principle

It assumes that the members of a given class have similar characteristics. So, a given observation is assigned the class of its nearest neighbours (number of nearest neighbour to be decided by the user)

Example



codes

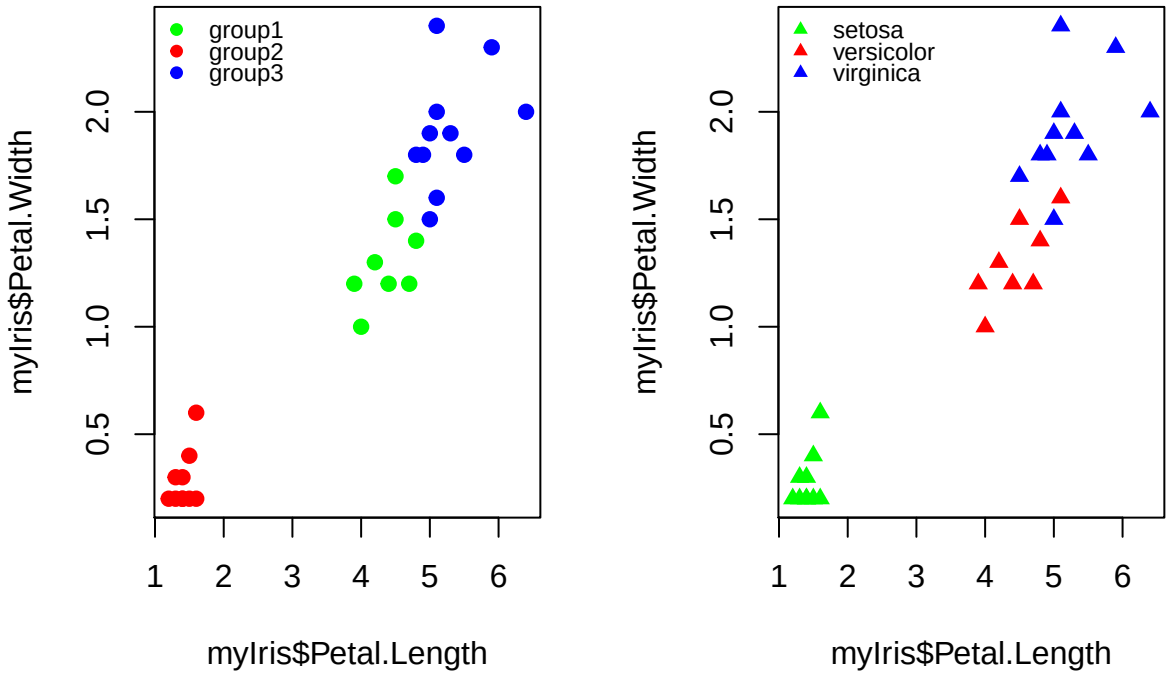
```
library(class)
myIris <- iris[,3:5]

set.seed(100)
inTrain <- sample(c(TRUE, FALSE), size = nrow(myIris), replace = TRUE, prob = c(0.2,0.8))
trainData <- myIris[inTrain,1:2]
trainClass <- myIris[inTrain,3]
testData <- myIris[!inTrain,1:2]
testClass <- myIris[!inTrain,3]
```

```r
predClass <- knn(trainData, testData, cl = trainClass, k = 3)
table(predClass, testClass)
```

```
##             testClass
## predClass    setosa versicolor virginica
##   setosa         40          0         0
##   versicolor      0         40         1
##   virginica       0          2        38
```

clustering example

kmeans clustering

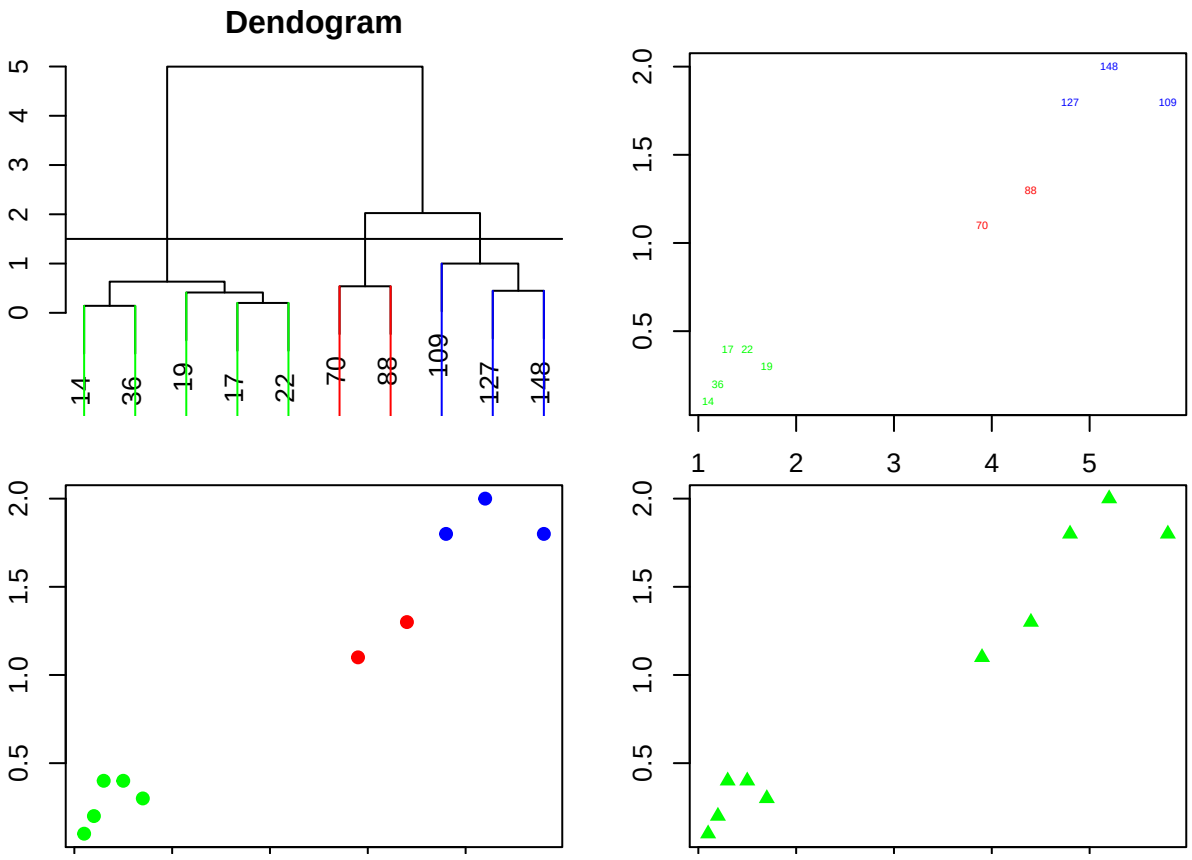


Codes

```
set.seed(100)
index <- sample(c(TRUE, FALSE), nrow(iris), p = c(0.2, 0.8), replace = TRUE)
myIris <- iris[index,3:4]
group <- iris$Species[index]
set.seed(100)
predGroup <- kmeans(myIris, centers = 3, nstart = 10)
predGroupC <- ifelse(predGroup$cluster==1, "setosa", ifelse(predGroup$cluster==2,
                                                            "versicolor", "virginnica"))
predGroupC <- factor(predGroupC)
table(predGroupC, group)
```

```
##             group
## predGroupC   setosa versicolor virginica
##   setosa          0          7         1
##   versicolor     10          0         0
##   virginnica      0          1        10
```

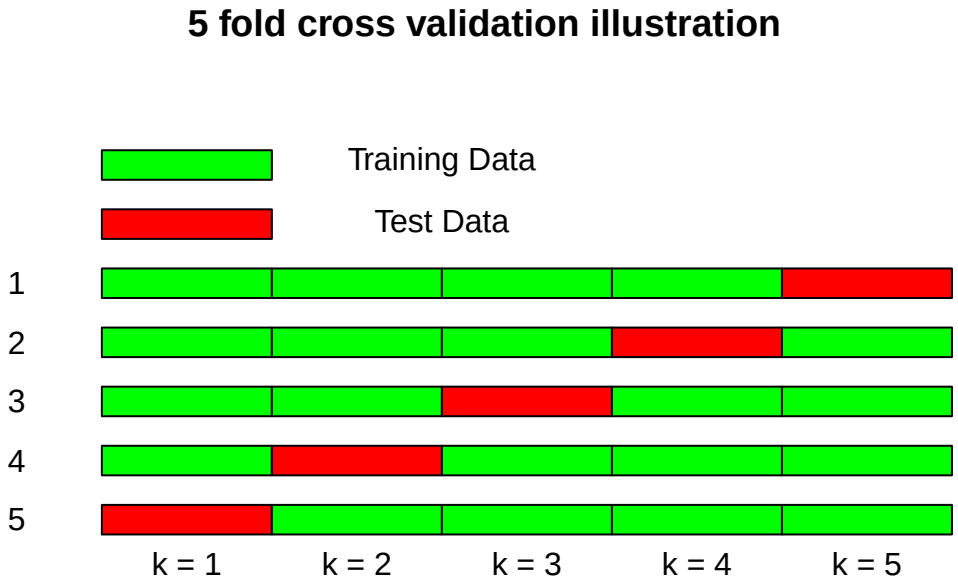
Hierarchichal Clustering




```
set.seed(4)
index <- sample(c(TRUE, FALSE), nrow(iris), p = c(0.05, 0.95), replace = TRUE)
myIris <- iris[index,3:4]
disM <- dist(myIris)
irisClust <- hclust(disM)
clusters <- cutree(irisClust, k = 3)
```

Codes

Cross-validation



baye's theorem

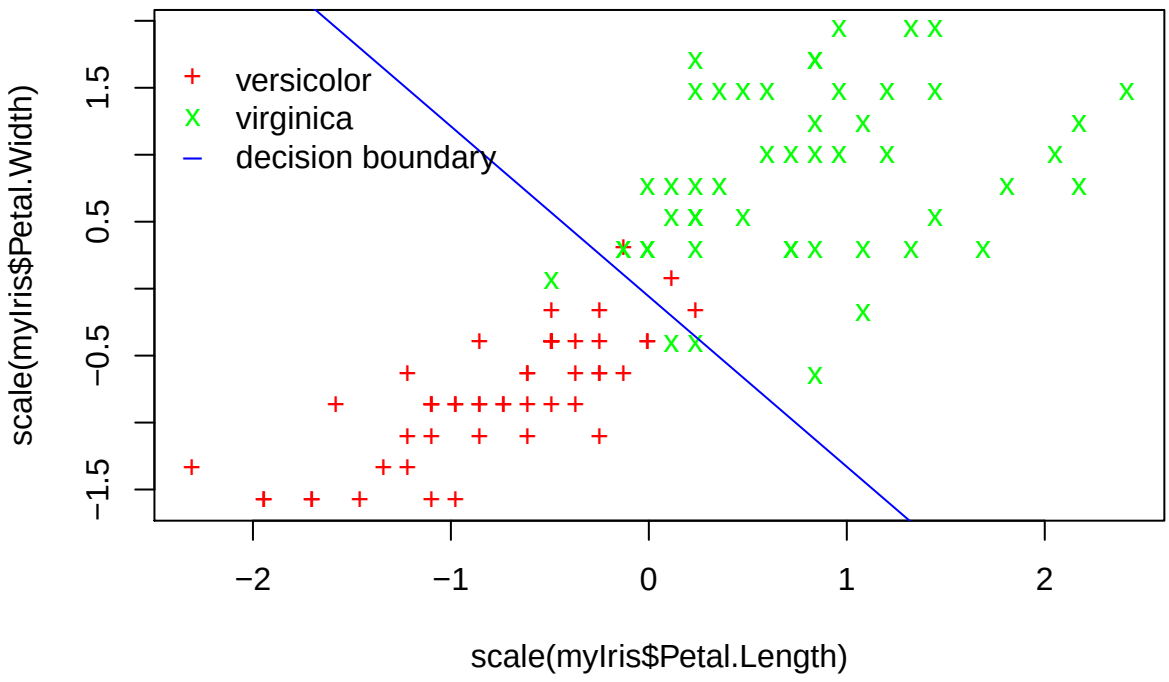
```
#head(Titanic)
```

SVM

Working Principle

It classifies a test observation depending on which side of a hyperplane it lies. The hyperplabe is chosen to correctly separate most of the training observations into two classes

Example



Codes

```
library(e1071)
inSetosa <- iris$Species == "setosa"
myIris <- iris[!inSetosa, c("Petal.Length", "Petal.Width", "Species")]
myIris$Species <- factor(myIris$Species, levels = c("versicolor", "virginica"))
svmModel <- svm(Species ~ ., data = myIris, kernal = "linear",
                scale = TRUE)
```

```
summary(svmModel)
prediction <- predict(svmModel, myIris[, 1:2])
table(prediction, myIris$Species)
```

Assessment

```
prediction <- predict(svmModel, myIris[, 1:2])
table(prediction, myIris$Species)
```

```
##
## prediction   versicolor virginica
##   versicolor         47         3
##   virginica           3        47
```

```
mean(prediction==myIris$Species)
```

```
## [1] 0.94
```