Machine Learning

Ravi Kumar Tiwari

14 June 2016

```r
library(caret)
library(rpart.plot)
library(rattle)
library(calibrate)
library(randomForest)
```

Decision Tree Example

Problem Description

Given a data set that contains some observation and corresponding class label, can a machine learning algorithm be trained to determine the class label of any data set (not necessarily the data that was used for training) from its observation

Solution using decision tree

```r
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
```

Create data partition

```r
inTrain <- createDataPartition(iris$Species, p = 0.6, list = FALSE)
trainData <- iris[inTrain,]
testData <- iris[-inTrain,]
```

Build a decision tree model and use it for prediction on test data set

```r
treeModel <- train(Species ~ ., data = trainData, method = "rpart")
preClass <- predict(treeModel, newdata = testData)
cMatrix <- confusionMatrix(preClass, testData$Species)
cMatrix$table
```

```
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         19         3
##   virginica       0          1        17
```

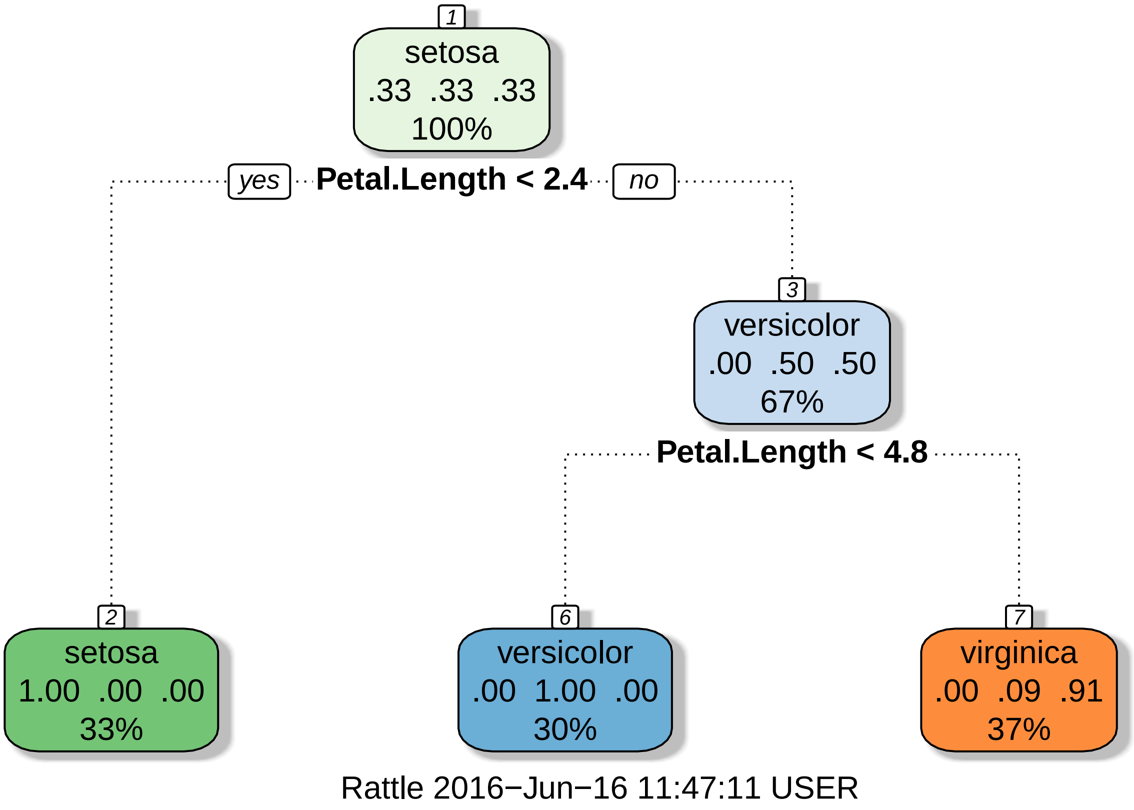
Look at what are the important variables

```
varImp(treeModel)
```

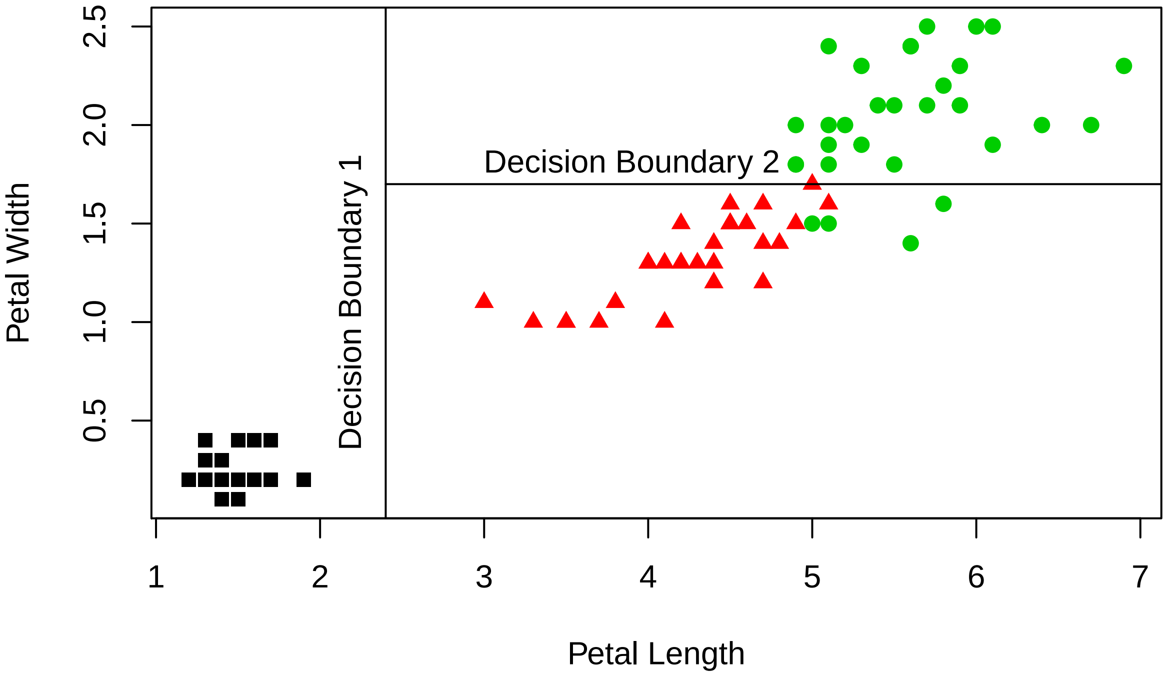
```
## rpart variable importance
##
##              Overall
## Petal.Length  100.00
## Petal.Width    95.87
## Sepal.Length   34.05
## Sepal.Width     0.00
```

Visualization of the decision tree

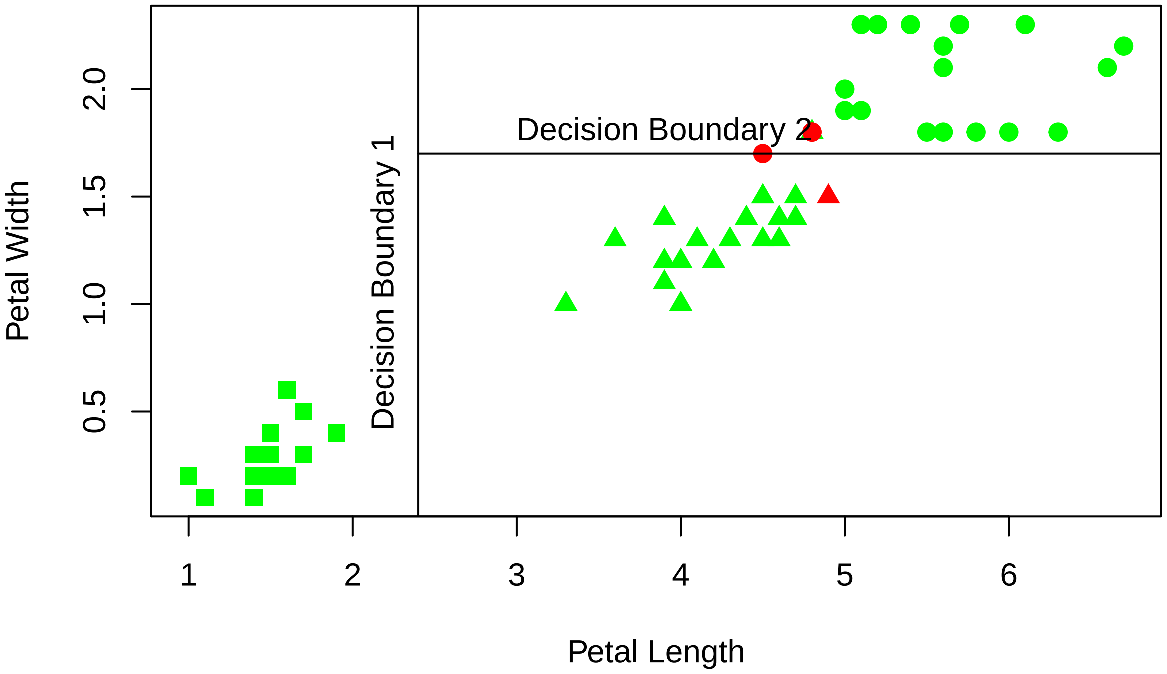
```
fancyRpartPlot(treeModel$finalModel)
```



Alternative visualization of the decision tree



Visualization of the prediction result

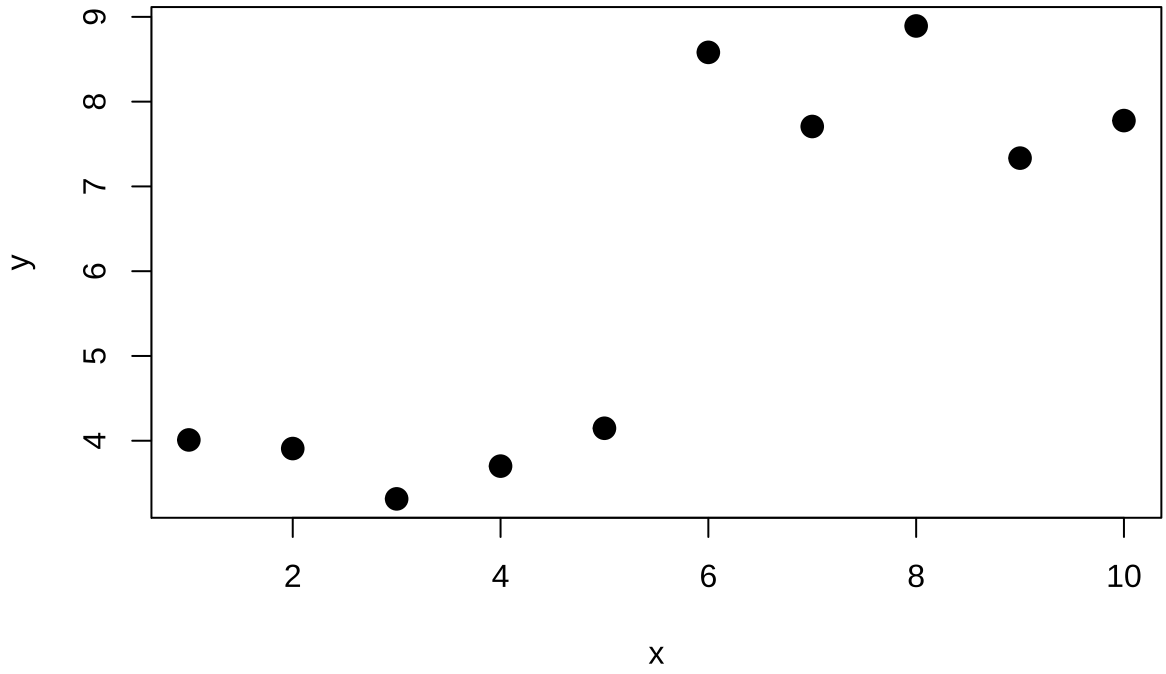

Decision Boundary 1
Decision Boundary 2
Petal Width
2.0
1.5
1.0
0.5
1
2
3
4
5
6
Petal Length

Clustering Example

K-means clustering

```
head(obs)
```

```
##    x        y
## 1  1 4.009373
## 2  2 3.907874
## 3  3 3.314335
## 4  4 3.700416
## 5  5 4.147273
## 6  6 8.581343
```



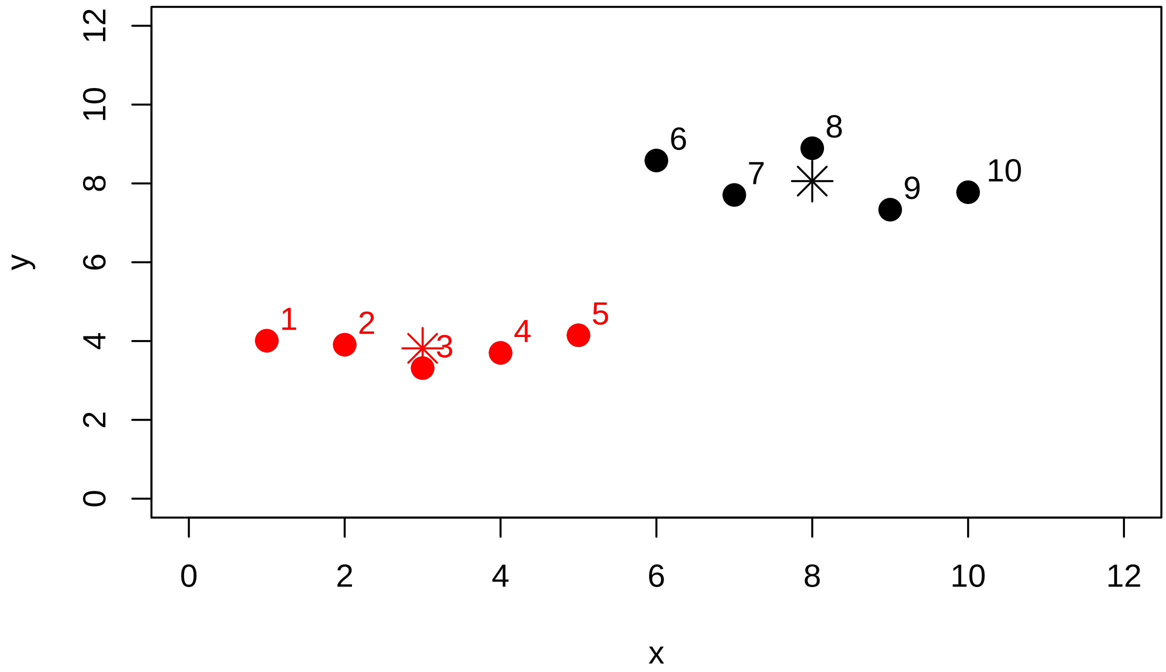
Create 2 clusters

```
kmeansObj <-  kmeans(obs, centers = 2)
data.frame(obs, cluster = kmeansObj$cluster)
```

```
##      x        y cluster
## 1    1 4.009373       2
## 2    2 3.907874       2
## 3    3 3.314335       2
## 4    4 3.700416       2
## 5   5 4.147273       2
## 6   6 8.581343       1
## 7   7 7.707038       1
## 8   8 8.892733       1
## 9   9 7.333703       1
## 10 10 7.776717       1
```

```
kmeansObj$centers
```

```
##   x        y
## 1 8 8.058307
## 2 3 3.815854
```



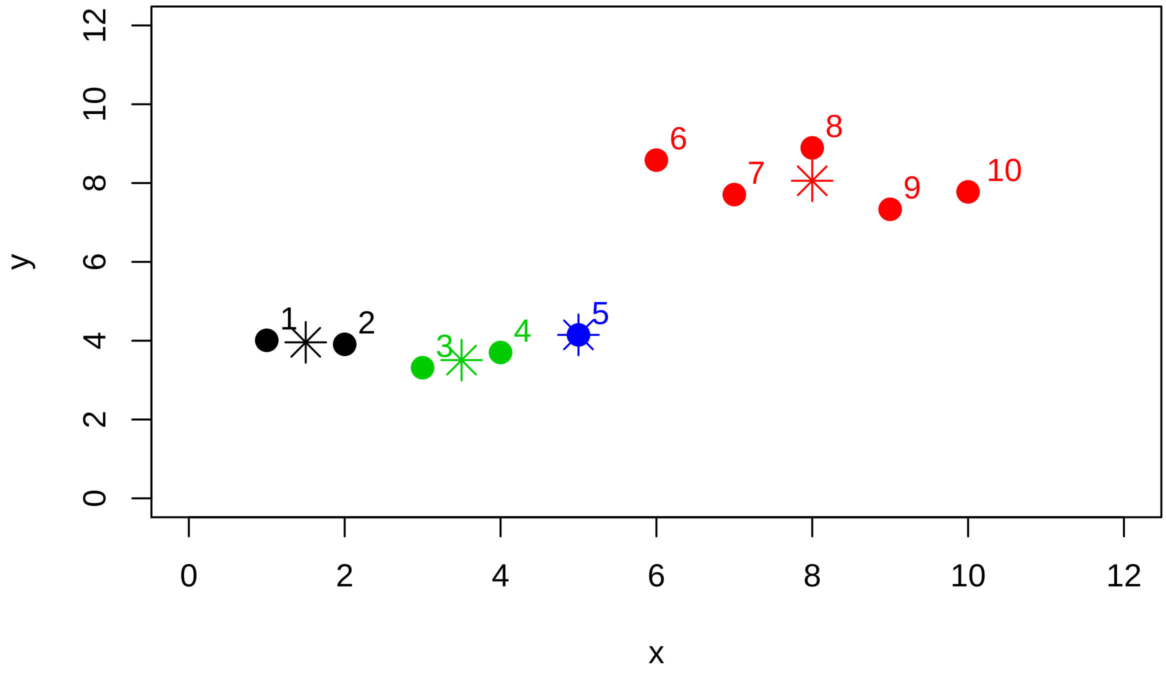
Create 4 clusters

```
kmeansObj <-  kmeans(obs, centers = 4)
data.frame(obs, cluster = kmeansObj$cluster)
```

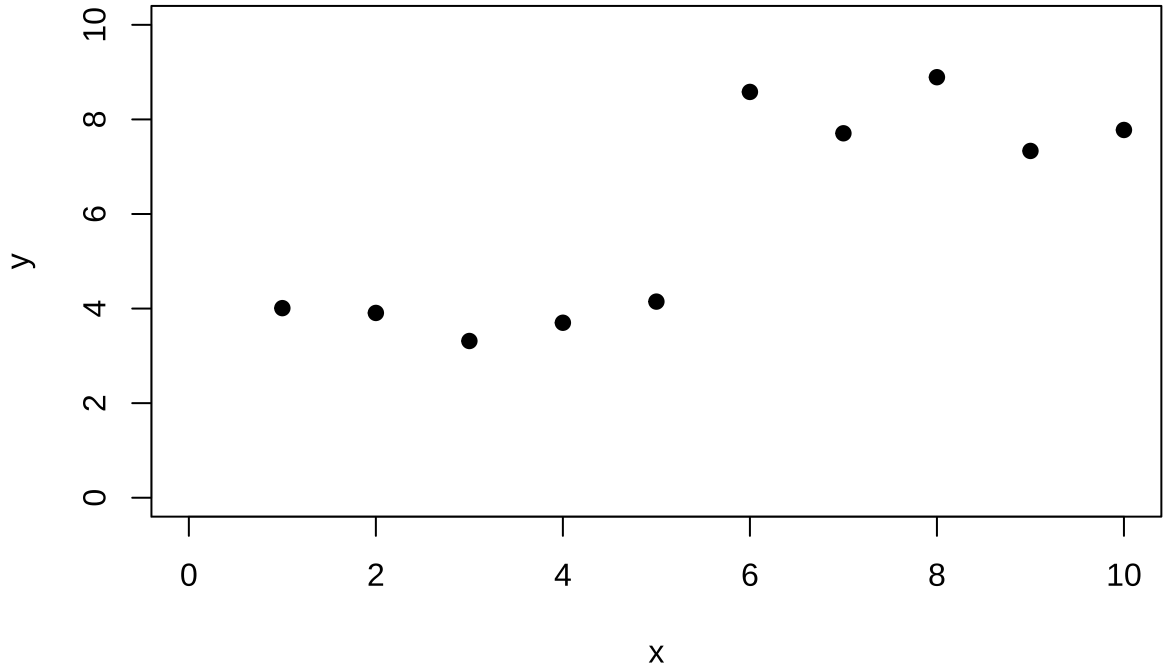
```
##     x        y cluster
## 1   1 4.009373       1
## 2   2 3.907874       1
## 3   3 3.314335       3
## 4   4 3.700416       3
## 5   5 4.147273       4
## 6   6 8.581343       2
## 7   7 7.707038       2
## 8   8 8.892733       2
## 9   9 7.333703       2
## 10 10 7.776717       2
```

```
kmeansObj$centers
```

```
##     x        y
## 1 1.5 3.958623
## 2 8.0 8.058307
## 3 3.5 3.507375
## 4 5.0 4.147273
```



Hierarchical Clustering

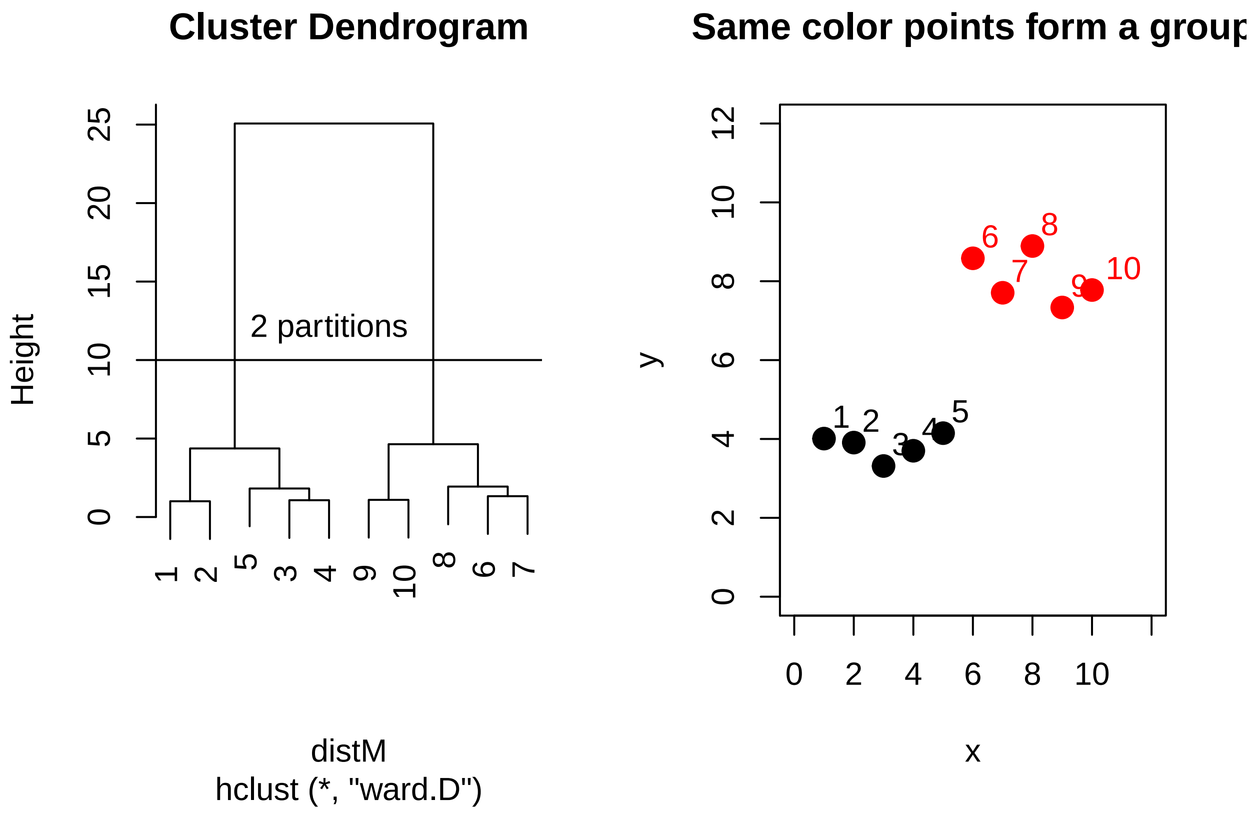


```
distM <- dist(obs)
clusters <- hclust(distM, method = "ward.D")
```

Create 2 partitions

```
mem <- cutree(clusters, k =2)
data.frame(obs, cluster = mem)
```

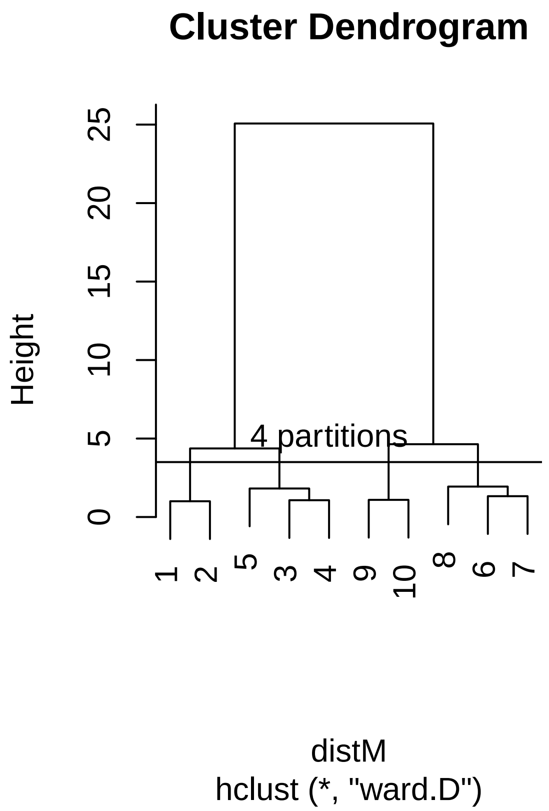
```
##     x        y cluster
## 1   1 4.009373       1
## 2   2 3.907874       1
## 3   3 3.314335       1
## 4   4 3.700416       1
## 5   5 4.147273       1
## 6   6 8.581343       2
## 7   7 7.707038       2
## 8   8 8.892733       2
## 9   9 7.333703       2
## 10 10 7.776717       2
```

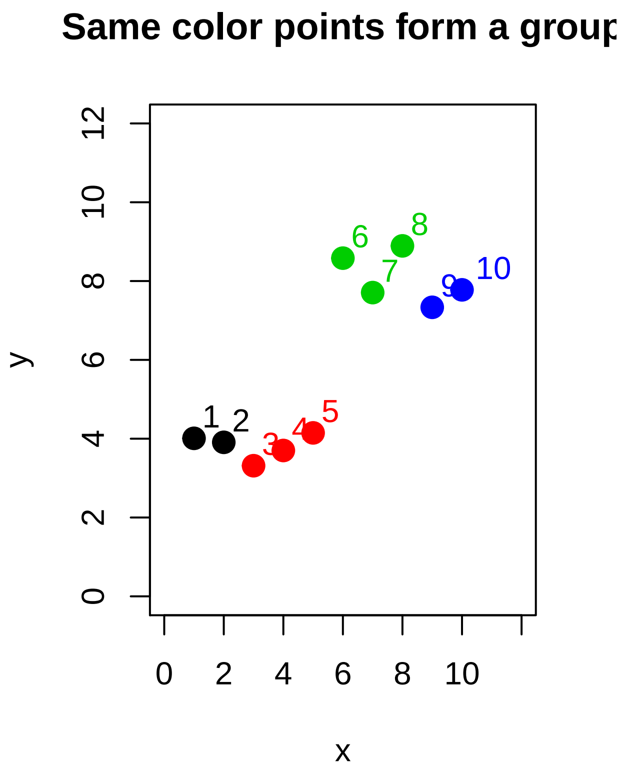


Create 4 partitions

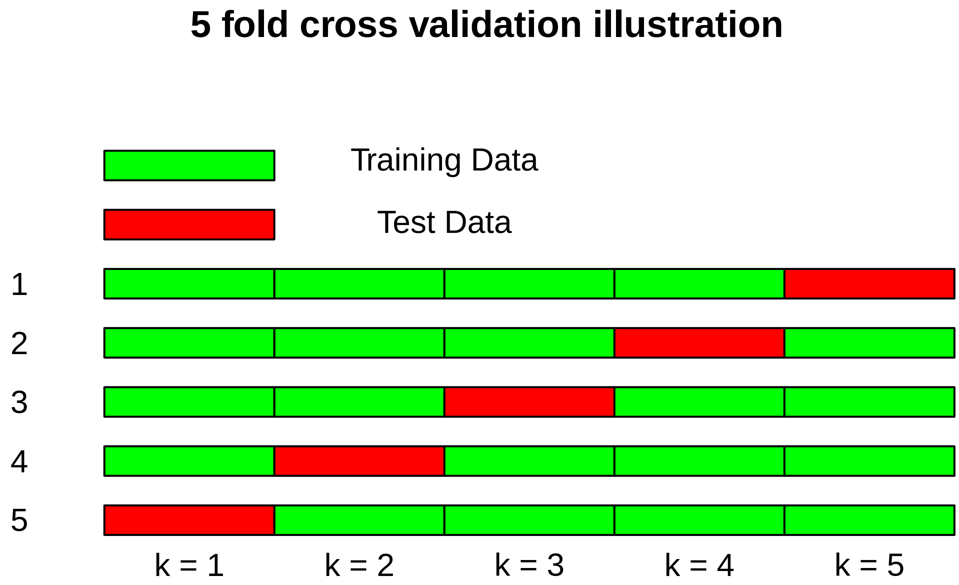
```
mem <- cutree(clusters, k =4)
data.frame(obs, cluster = mem)
```

```
##      x        y cluster
## 1    1 4.009373       1
## 2    2 3.907874       1
## 3    3 3.314335       2
## 4    4 3.700416       2
## 5    5 4.147273       2
## 6    6 8.581343       3
## 7    7 7.707038       3
## 8    8 8.892733       3
## 9    9 7.333703       4
## 10  10 7.776717       4
```





Cross-validation

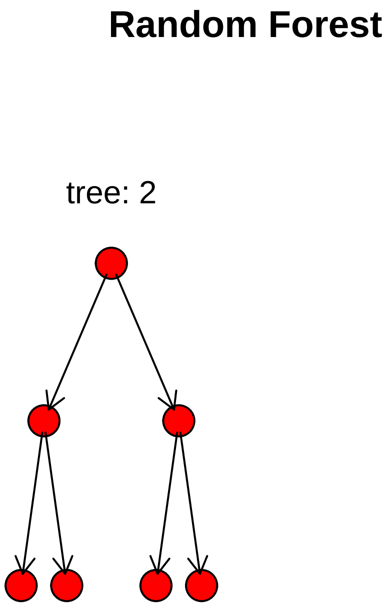


### random Forest

There are two parameters in the random forest ntree mtry

```
## [1] 2 1 1
## [1] 2 2 1
## [1] 3 1 1
## [1] 3 2 1
## [1] 3 3 2
## [1] 3 4 2
```

```
## [1] 2 1 1
## [1] 2 2 1
## [1] 3 1 1
## [1] 3 2 1
## [1] 3 3 2
## [1] 3 4 2
```



Random Forest

tree: 2

```
rfModel <- randomForest(Species ~ . , data = trainData, ntree = 3)
```