Data Mining with R part 1

Veerasak Kritsanapraphan

March 10, 2017

Slide and Sample Data

https://github.com/vkrit/chula_datamining.

github
SOCIAL CODING

Agenda

1. Overview and data visualization
2. Data Preparation
3. Predictive Data Mining
   - Decision Tree
   - K-Nearest Neighbor
   - Naive Bayes Classifier
   - Neural Network

Overview

Predictive Data Mining : two phases of processing

1. Training Phase : Create a model from traning data
2. Predicting phase (Testing) : Deploy the model to production and use that to predict the future outcome

Data

Iris Data Set from UCI Machine Learning Repository
https://archive.ics.uci.edu/ml/datasets/iris

Iris Data - Atrribute Information

Column	Data Description
1	Sepal Length in cm
2	Sepal Width in cm
3	Petal Length in cm
4	Petal Width in cm
5	Classes

Iris Data - Class (Label)

Column	Data
Class	• Iris Setosa • Iris Versicolor • Iris Verginica

Getting Data

```
iris <- read.csv("iris.data.csv", header = TRUE)
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##             Species  
##  Iris-setosa    :50  
##  Iris-versicolor:50  
##  Iris-virginica :50  
##                      
##                      
## 
```

Exploring Data

```
nrow(iris)
```

```
## [1] 150
```

```
table(iris$Species)
```

```
## 
##     Iris-setosa Iris-versicolor  Iris-virginica 
##              50              50              50 
```

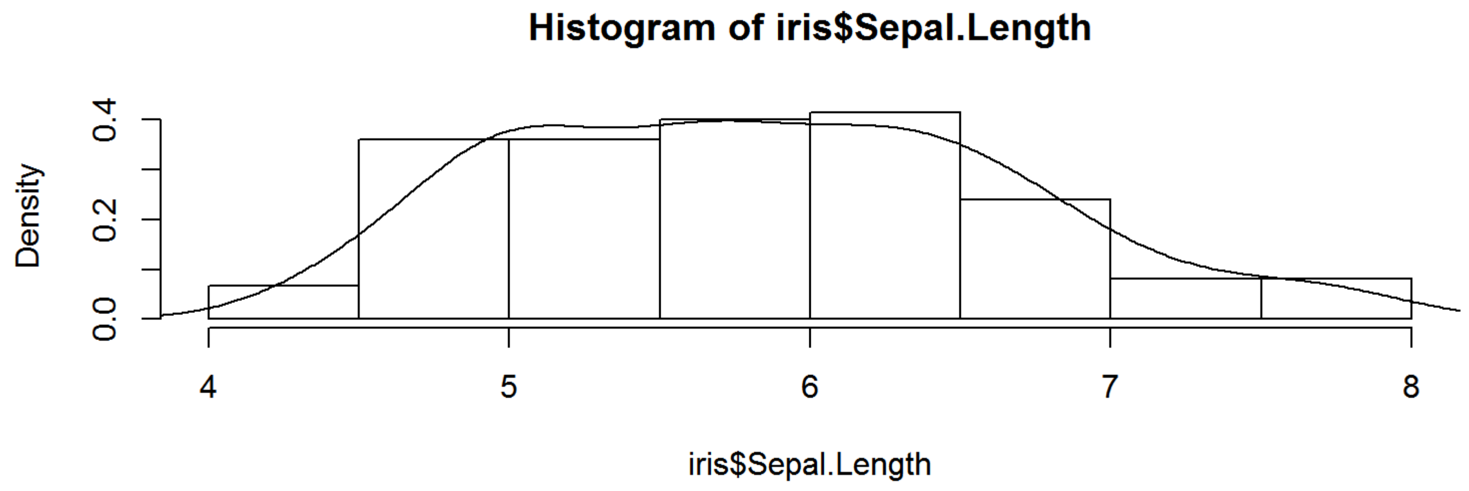
Data Visualization

- Visualizing existing data is a very useful way to come up with ideas about what features should be included.
- "Dataframe" in R is a common way where data samples are organized in a tabular structure.

Plot data according to their types

Let's start with numberic type first

```
# Plot the histogram
hist(iris$Sepal.Length, breaks = 10, prob = T)
# Plot the density curve
lines(density(iris$Sepal.Length))
```

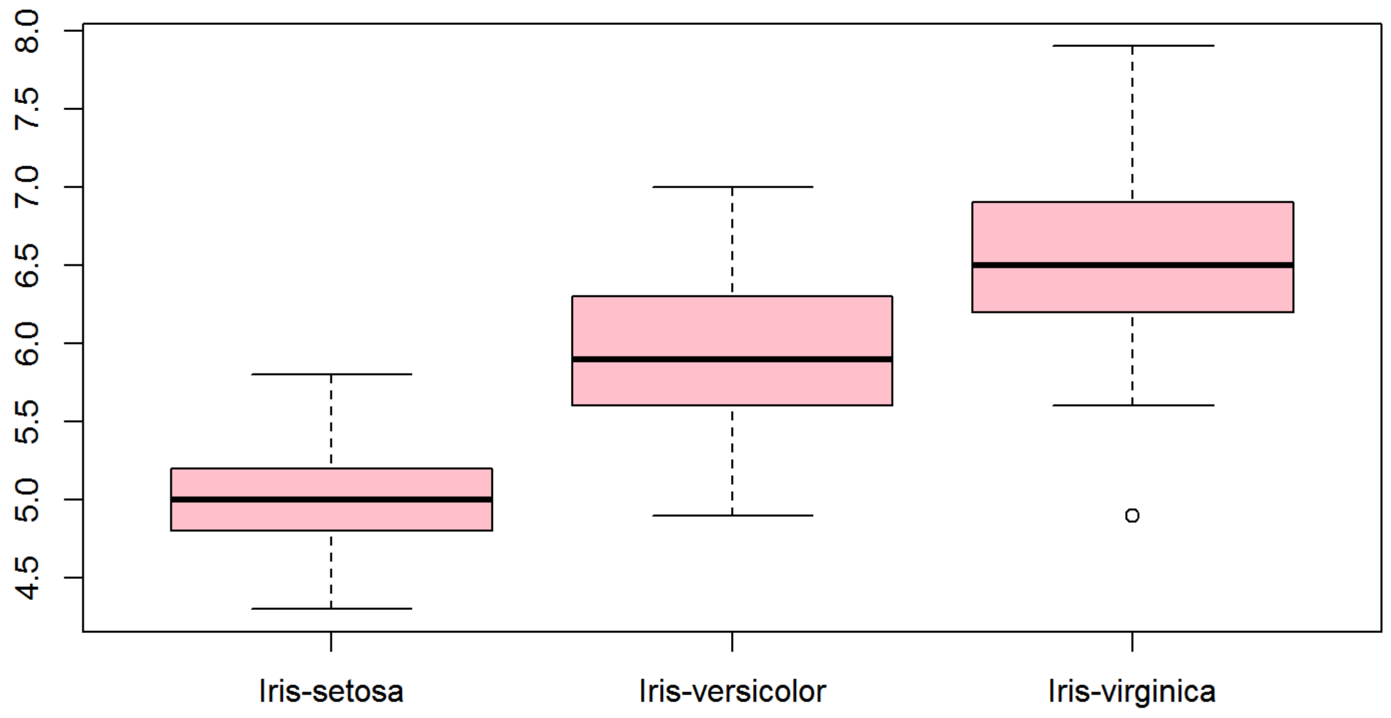


Next data in categorical type

```
categories <- table(iris$Species)
barplot(categories, col = c("red", "white", "blue"))
```

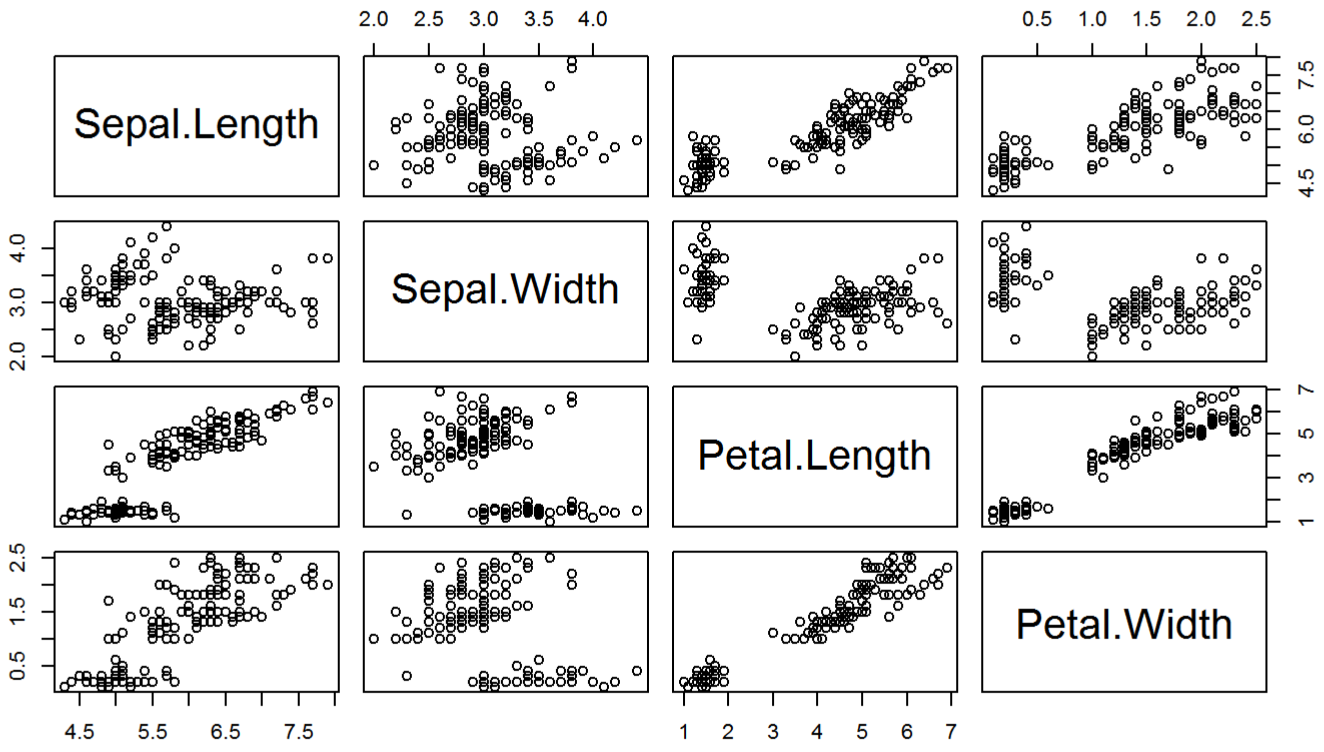
Let cross data between numeric and categorical

```
boxplot(Sepal.Length ~ Species, data = iris, col = c("pink"))
```



Next, let do correlation in charts using "pairs"

```
# Scatter plot for all pairs
pairs(iris[, c(1, 2, 3, 4)])
```



14/50

Compute the correlation matrix

```
# Compute the correlation matrix
cor(iris[, c(1, 2, 3, 4)])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1093692    0.8717542   0.8179536
## Sepal.Width    -0.1093692   1.0000000   -0.4205161  -0.3565441
## Petal.Length    0.8717542  -0.4205161    1.0000000   0.9627571
## Petal.Width     0.8179536  -0.3565441    0.9627571   1.0000000
```

Draw regression line for suspect attribute to see relationship

```r
plot(Petal.Width ~ Sepal.Length, data = iris)
model <- lm(Petal.Width ~ Sepal.Length, data = iris)
abline(model)
model2 <- lowess(iris$Petal.Width ~ iris$Sepal.Length)
lines(model2, col = "red")
```

Preparing Training Data

At this step, the purpose is to transform the raw data in a form that can fit into the data mining model.

```
- Data sampling
- Data validation and handle missing data
- Normalize numeric value into a uniform range
- Compute aggregated value (a special case is to compute frequency counts)
- Expand categorical field to binary fields
- Discretize numeric value into categories
- Create derived fields from existing fields
- Reduce dimensionality
- Power and Log transformation
```

Data Sampling

```
# Select 10 records out from iris with replacement
index <- sample(1:nrow(iris), 10, replace = T)
index
```

```
##  [1]  90 107 115  62  26  16   1 116  87  60
```

```
# Subset iris to irissample from index
irissample <- iris[index, ]
irissample
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width         Species
## 90           5.5         2.5          4.0         1.3 Iris-versicolor
## 107          4.9         2.5          4.5         1.7  Iris-virginica
## 115          5.8         2.8          5.1         2.4  Iris-virginica
## 62           5.9         3.0          4.2         1.5 Iris-versicolor
## 26           5.0         3.0          1.6         0.2     Iris-setosa
## 16           5.7         4.4          1.5         0.4     Iris-setosa
## 1            5.1         3.5          1.4         0.2     Iris-setosa
## 116          6.4         3.2          5.3         2.3  Iris-virginica
## 87           6.7         3.1          4.7         1.5 Iris-versicolor
## 60           5.2         2.7          3.9         1.4 Iris-versicolor
```

Impute missing data

- Discard the whole record
- Infer missing data with Average or Median

```
# Create some missing data
irissample[10, 1] <- NA
irissample
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width         Species
## 90           5.5         2.5          4.0         1.3 Iris-versicolor
## 107          4.9         2.5          4.5         1.7  Iris-virginica
## 115          5.8         2.8          5.1         2.4  Iris-virginica
## 62           5.9         3.0          4.2         1.5 Iris-versicolor
## 26           5.0         3.0          1.6         0.2     Iris-setosa
## 16           5.7         4.4          1.5         0.4     Iris-setosa
## 1            5.1         3.5          1.4         0.2     Iris-setosa
## 116          6.4         3.2          5.3         2.3  Iris-virginica
## 87           6.7         3.1          4.7         1.5 Iris-versicolor
## 60            NA         2.7          3.9         1.4 Iris-versicolor
```

```
# Fix using Mean
library(e1071)
fixiris1 <- impute(irissample[, 1:4], what = "mean")
fixiris1
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 90      5.500000         2.5          4.0         1.3
## 107     4.900000         2.5          4.5         1.7
## 115     5.800000         2.8          5.1         2.4
## 62      5.900000         3.0          4.2         1.5
## 26      5.000000         3.0          1.6         0.2
## 16      5.700000         4.4          1.5         0.4
## 1       5.100000         3.5          1.4         0.2
## 116     6.400000         3.2          5.3         2.3
## 87      6.700000         3.1          4.7         1.5
## 60      5.666667         2.7          3.9         1.4
```

```
# Fix using Median
library(e1071)
fixiris2 <- impute(irissample[, 1:4], what = "median")
fixiris2
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 90           5.5         2.5          4.0         1.3
## 107          4.9         2.5          4.5         1.7
## 115          5.8         2.8          5.1         2.4
## 62           5.9         3.0          4.2         1.5
## 26           5.0         3.0          1.6         0.2
## 16           5.7         4.4          1.5         0.4
## 1            5.1         3.5          1.4         0.2
## 116          6.4         3.2          5.3         2.3
## 87           6.7         3.1          4.7         1.5
## 60           5.7         2.7          3.9         1.4
```

Normalize numeric value

```
# Scale the colums x-mean(x)/standard deviation
scaleiris <- scale(iris[, 1:4])
head(scaleiris)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]   -0.8976739   1.0286113    -1.336794   -1.308593
## [2,]   -1.1392005  -0.1245404    -1.336794   -1.308593
## [3,]   -1.3807271   0.3367203    -1.393470   -1.308593
## [4,]   -1.5014904   0.1060900    -1.280118   -1.308593
## [5,]   -1.0184372   1.2592416    -1.336794   -1.308593
## [6,]   -0.5353840   1.9511326    -1.166767   -1.046525
```

Reduce dimension

There are two ways to reduce the number of input attributes.

```
1. Removing irrelevant input variables.
2. Removing redundant input variables.
```

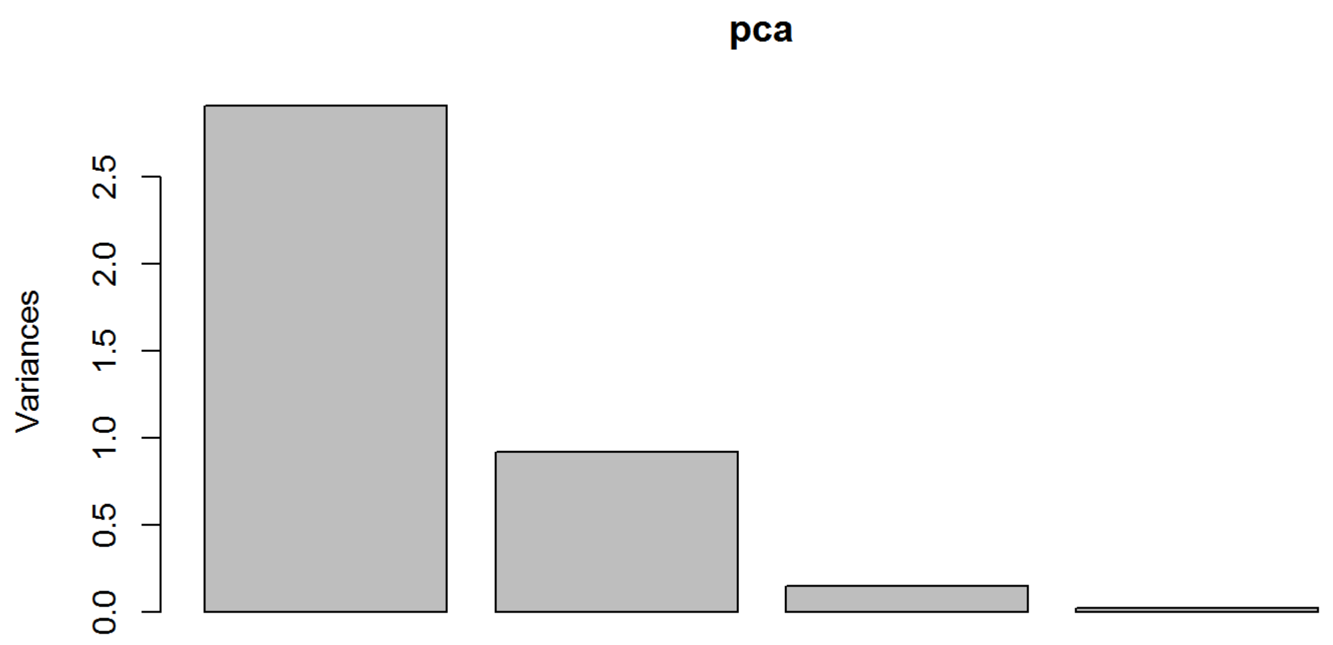
```
# Use iris data set
cor(iris[, -5])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1093692    0.8717542   0.8179536
## Sepal.Width    -0.1093692   1.0000000   -0.4205161  -0.3565441
## Petal.Length    0.8717542  -0.4205161    1.0000000   0.9627571
## Petal.Width     0.8179536  -0.3565441    0.9627571   1.0000000
```

```
# Compute PCA Notice that PC1 and PC2 covers most variation
pca <- prcomp(iris[, -5], scale = T)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7061 0.9598 0.38387 0.14355
## Proportion of Variance 0.7277 0.2303 0.03684 0.00515
## Cumulative Proportion  0.7277 0.9580 0.99485 1.00000
```

```
# Plot PCA
plot(pca)
```



26/50

```
pca$rotation
```

```
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5223716 -0.37231836  0.7210168  0.2619956
## Sepal.Width  -0.2633549 -0.92555649 -0.2420329 -0.1241348
## Petal.Length  0.5812540 -0.02109478 -0.1408923 -0.8011543
## Petal.Width   0.5656110 -0.06541577 -0.6338014  0.5235463
```

```
predict(pca)[1:2, ]
```

```
##            PC1        PC2       PC3        PC4
## [1,] -2.256981 -0.5040154 0.1215362 0.02299628
## [2,] -2.079459  0.6532164 0.2264921 0.10286364
```

```
biplot(pca)
```

Add derived attributes

```
iris2 <- transform(iris, ratio = round(Sepal.Length/Sepal.Width,
    2))
head(iris2)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width     Species ratio
## 1          5.1         3.5          1.4         0.2 Iris-setosa  1.46
## 2          4.9         3.0          1.4         0.2 Iris-setosa  1.63
## 3          4.7         3.2          1.3         0.2 Iris-setosa  1.47
## 4          4.6         3.1          1.5         0.2 Iris-setosa  1.48
## 5          5.0         3.6          1.4         0.2 Iris-setosa  1.39
## 6          5.4         3.9          1.7         0.4 Iris-setosa  1.38
```

Discretize numeric value into categories

```
# Equal width cuts
segments = 10
maxL <- max(iris$Petal.Length)
minL <- min(iris$Petal.Length)
theBreaks <- seq(minL, maxL, by = (maxL - minL)/segments)
cutPetalLength <- cut(iris$Petal.Length, breaks = theBreaks,
    include.lowest = T)
newdata <- data.frame(origin.Petal.Len = iris$Petal.Length,
    cut.Petal.Len = cutPetalLength)
head(newdata)
```

```
##   origin.Petal.Len cut.Petal.Len
## 1              1.4      [1,1.59]
## 2              1.4      [1,1.59]
## 3              1.3      [1,1.59]
## 4              1.5      [1,1.59]
## 5              1.4      [1,1.59]
## 6              1.7   (1.59,2.18]
```

```r
# Constant frequency / height
myBreaks <- quantile(iris$Petal.Length, probs = seq(0, 1, 1/segments))
cutPetalLength2 <- cut(iris$Petal.Length, breaks = myBreaks,
    include.lowest = T)
newdata2 <- data.frame(orig.Petal.Lens = iris$Petal.Length,
    cut.Petal.Len = cutPetalLength2)
head(newdata2)
```

```
##   orig.Petal.Lens cut.Petal.Len
## 1             1.4       [1,1.4]
## 2             1.4       [1,1.4]
## 3             1.3       [1,1.4]
## 4             1.5     (1.4,1.5]
## 5             1.4       [1,1.4]
## 6             1.7     (1.7,3.9]
```

Binarize categorical attributes

```
cat <- levels(iris$Species)
binarize <- function(x) {
    return(iris$Species == x)
}
newcols <- sapply(cat, binarize)
colnames(newcols) <- cat
data <- cbind(iris[, c("Species")], newcols)
```

Binarize categorical attributes (cont)

```
data[45:55, ]
```

```
##         Iris-setosa Iris-versicolor Iris-virginica
##  [1,] 1           1               0              0
##  [2,] 1           1               0              0
##  [3,] 1           1               0              0
##  [4,] 1           1               0              0
##  [5,] 1           1               0              0
##  [6,] 1           1               0              0
##  [7,] 2           0               1              0
##  [8,] 2           0               1              0
##  [9,] 2           0               1              0
## [10,] 2           0               1              0
## [11,] 2           0               1              0
```

Data Mining Techniques

Iris Data Preparation

```
# sample iris into 2 sets (training, testing) with 70%/30%
set.seed(1234)
ind <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7,
    0.3))
trainData <- iris[ind == 1, ]
testData <- iris[ind == 2, ]
```

Decision Tree

Libray name -> party

Function name -> ctree

Installation

install.packages("party")

Decision Tree - Create Model

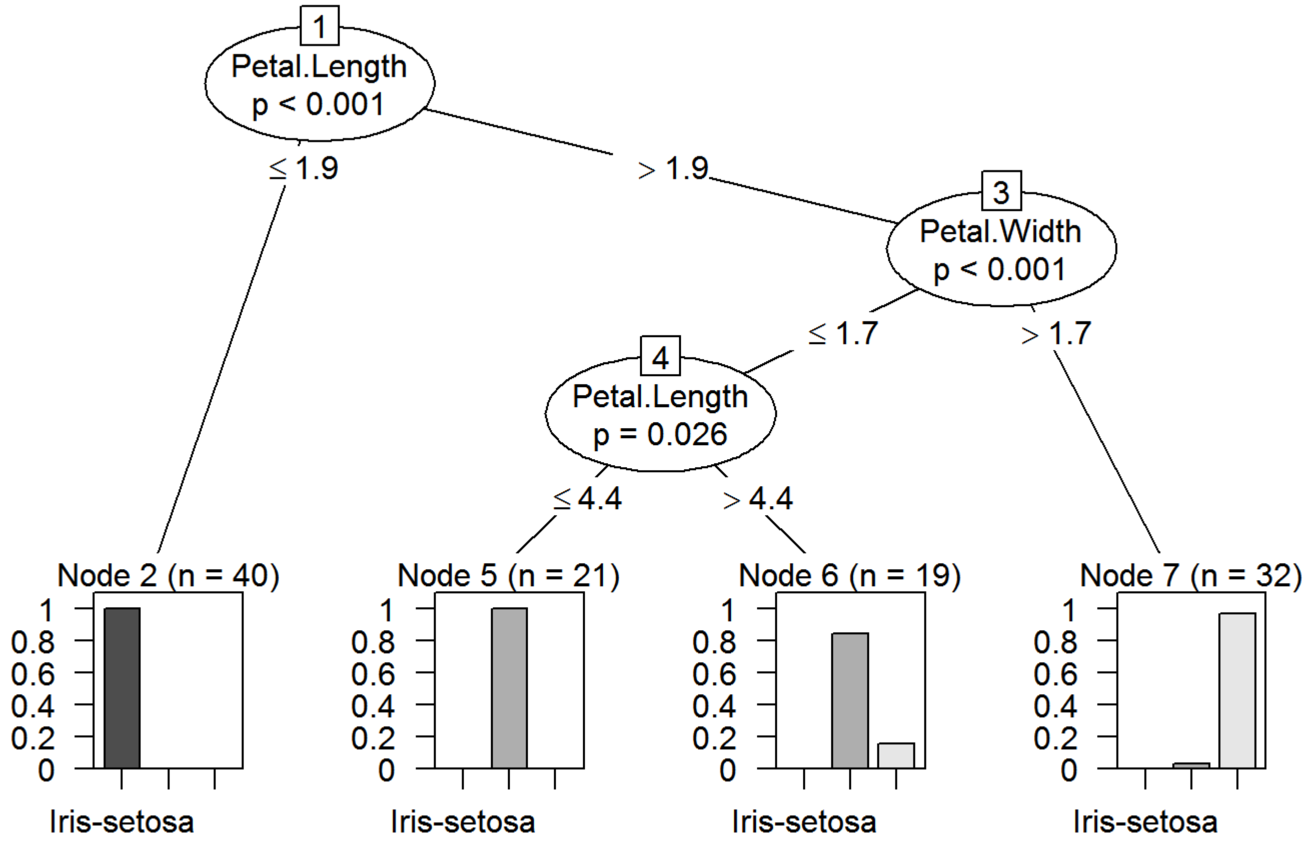
```
library(party)
myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length +
    Petal.Width
iris_ctree <- ctree(myFormula, data = trainData)
# Check the prediction
table(predict(iris_ctree), trainData$Species)
```

```
##
##                     Iris-setosa Iris-versicolor Iris-virginica
##   Iris-setosa                40               0              0
##   Iris-versicolor             0              37              3
##   Iris-virginica              0               1             31
```

```
print(iris_ctree)
```

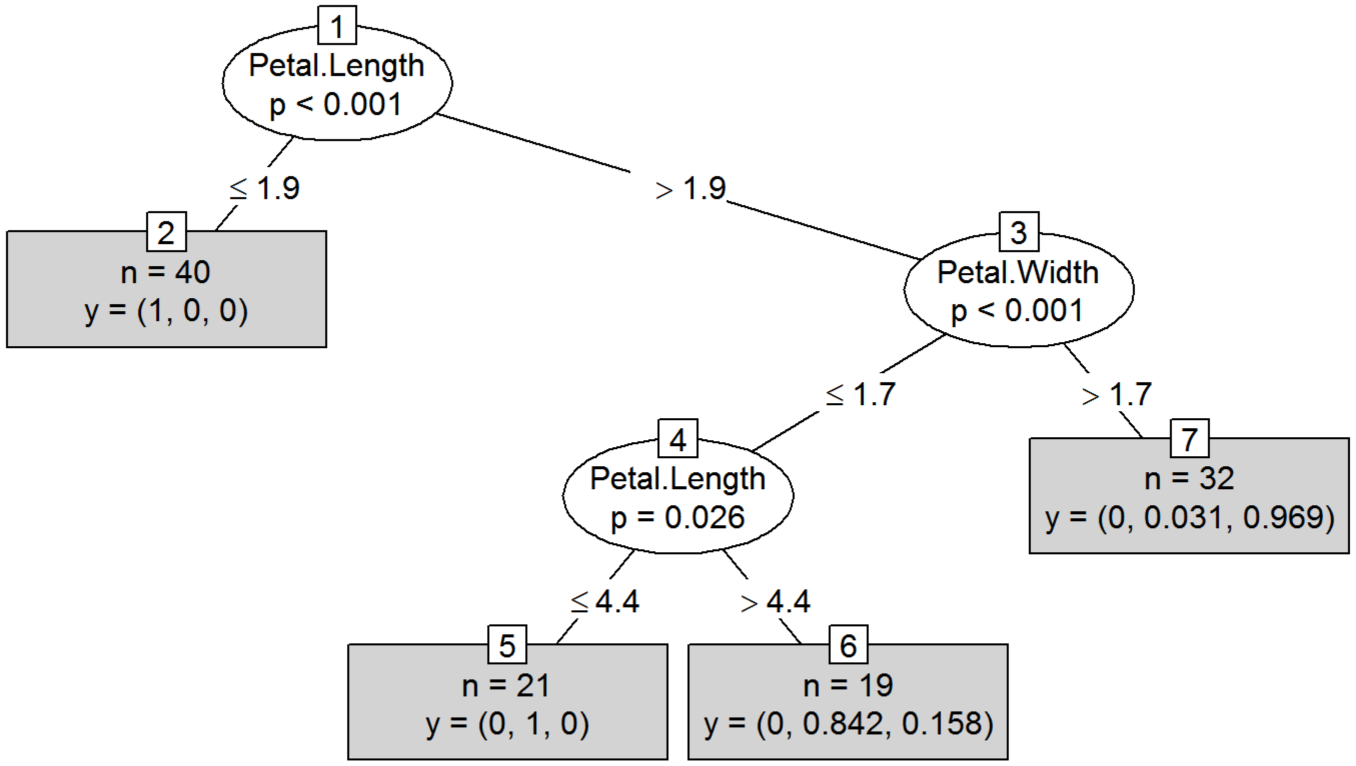
```
##
##   Conditional inference tree with 4 terminal nodes
##
## Response:  Species
## Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
## Number of observations:  112
##
## 1) Petal.Length <= 1.9; criterion = 1, statistic = 104.637
##   2)*  weights = 40
## 1) Petal.Length > 1.9
##   3) Petal.Width <= 1.7; criterion = 1, statistic = 48.939
##     4) Petal.Length <= 4.4; criterion = 0.974, statistic = 7.397
##       5)*  weights = 21
##     4) Petal.Length > 4.4
##       6)*  weights = 19
##   3) Petal.Width > 1.7
##     7)*  weights = 32
```

```
plot(iris_ctree)
```



39/50

```r
plot(iris_ctree, type = "simple")
```



Decision Tree - Prediction

```
# Predict on test data
testPred <- predict(iris_ctree, newdata = testData)
table(testPred, testData$Species)
```

```
##
## testPred          Iris-setosa Iris-versicolor Iris-virginica
##   Iris-setosa              10               0              0
##   Iris-versicolor           0              12              2
##   Iris-virginica            0               0             14
```

K-Nearest Neighbor Classfication (k-NN)

Library -> class

Function -> knn

Installation

install.packages("class")

K-NN

```
library(class)
train_input <- as.matrix(trainData[, -5])
train_output <- as.vector(trainData[, 5])
test_input <- as.matrix(testData[, -5])
prediction <- knn(train_input, test_input, train_output, k = 5)
table(prediction, testData$Species)
```

```
##
## prediction        Iris-setosa Iris-versicolor Iris-virginica
##   Iris-setosa              10               0              0
##   Iris-versicolor           0              12              0
##   Iris-virginica            0               0             16
```

Naive Bayes Classifier

Library –> e1071

Function –> naiveBayes

Installation

install.packages("e1071")

Naive Bayes

```
library(e1071)
# can handle both categorical and numeric input but output
# must be categorical
model <- naiveBayes(Species ~ ., data = trainData)
prediction <- predict(model, testData[, -5])
table(prediction, testData[, 5])
```

```
##
## prediction        Iris-setosa Iris-versicolor Iris-virginica
##   Iris-setosa              10               0              0
##   Iris-versicolor           0              12              2
##   Iris-virginica            0               0             14
```

Neural Network

Library –> neuralnet

Function –> neuralnet

Installation

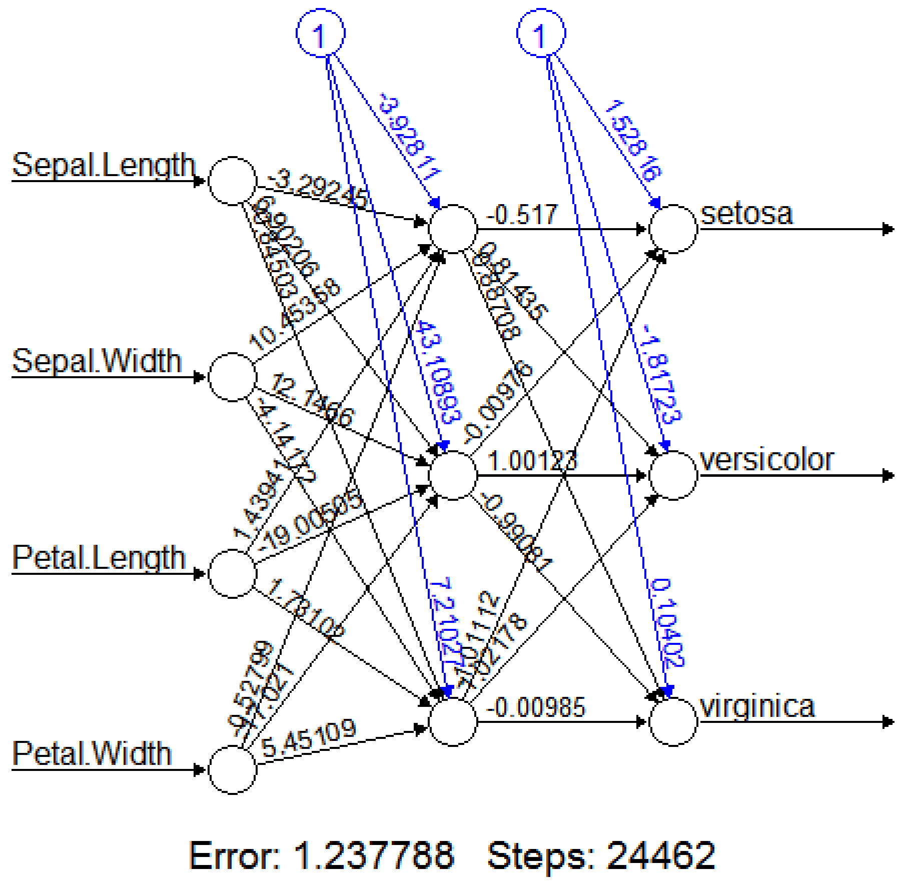
install.packages('neuralnet')

Neural Network

```
library(neuralnet)
nnet_trainData <- trainData
# Binarize the categorical output
nnet_trainData <- cbind(nnet_trainData, trainData$Species ==
    "Iris-setosa")
nnet_trainData <- cbind(nnet_trainData, trainData$Species ==
    "Iris-versicolor")
nnet_trainData <- cbind(nnet_trainData, trainData$Species ==
    "Iris-virginica")
names(nnet_trainData)[6] <- "setosa"
names(nnet_trainData)[7] <- "versicolor"
names(nnet_trainData)[8] <- "virginica"
nn <- neuralnet(setosa + versicolor + virginica ~ Sepal.Length +
    Sepal.Width + Petal.Length + Petal.Width, data = nnet_trainData,
    hidden = c(3))
```

Neural Network - Plot Model

```
plot(nn)
```



nn

Neural Network - Prediction

```
mypredict <- compute(nn, testData[-5])$net.result
# put multiple binary output to categorical output
maxidx <- function(arr) {
    return(which(arr == max(arr)))
}
idx <- apply(mypredict, c(1), maxidx)
prediction <- c("Iris-setosa", "Iris-versicolor", "Iris-virginica")[idx]
table(prediction, testData$Species)
```

```
##
## prediction        Iris-setosa Iris-versicolor Iris-virginica
##   Iris-setosa              10               0              0
##   Iris-versicolor           0              12              0
##   Iris-virginica            0               0             16
```

End of Part 1