R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```
health <- read.csv("C:\\Users\\srava\\Documents\\Ryerson university\\Capstone project\\Maternal Health
health2 <- health;
health2[health2$RiskLevel == "high risk","RiskLevel"] <- 2
health2[health2$RiskLevel == "low risk","RiskLevel"] <- 0
health2[health2$RiskLevel == "mid risk","RiskLevel"] <- 1
head(health2)
```

```
##   ï..Age SystolicBP DiastolicBP    BS BodyTemp HeartRate RiskLevel
## 1     25        130          80 15.00       98        86         2
## 2     35        140          90 13.00       98        70         2
## 3     29         90          70  8.00      100        80         2
## 4     30        140          85  7.00       98        70         2
## 5     35        120          60  6.10       98        76         0
## 6     23        140          80  7.01       98        70         2
```

```
str(health2)
```

```
## 'data.frame':    1014 obs. of  7 variables:
##  $ ï..Age     : int  25 35 29 30 35 23 23 35 32 42 ...
##  $ SystolicBP : int  130 140 90 140 120 140 130 85 120 130 ...
##  $ DiastolicBP: int  80 90 70 85 60 80 70 60 90 80 ...
##  $ BS         : num  15 13 8 7 6.1 7.01 7.01 11 6.9 18 ...
##  $ BodyTemp   : num  98 98 100 98 98 98 98 102 98 98 ...
##  $ HeartRate  : int  86 70 80 70 76 70 78 86 70 70 ...
##  $ RiskLevel  : chr  "2" "2" "2" "2" ...
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

```
health2$RiskLevel<- as.numeric(as.character(health2$RiskLevel))
## Shuffling the data to make sure all the classes are included without bias
shuffle_index <- sample(1:nrow(health2))
head(shuffle_index)
```

```
## [1] 433 118  58 877 395 664
```

```
health2 <- health2[shuffle_index, ]
```

```r
head(health2)
```

```
##     ï..Age SystolicBP DiastolicBP   BS BodyTemp HeartRate RiskLevel
## 433     40        140         100 13.0      101        66         2
## 118     55        140         100 18.0       98        90         2
## 58      42        120          80  6.4       98        70         0
## 877     27        120          70  6.8       98        77         0
## 395     19        120          80  7.0       98        70         1
## 664     15         90          60  6.0       98        80         0
```

```r
## Splitting the dataset into training and test in 1:4 ratio

create_train_test <- function(data, size = 0.8, train = TRUE) {
  n_row = nrow(data)
  total_row = size * n_row
  train_sample <- 1: total_row
  if (train == TRUE) {
    return (data[train_sample, ])
  } else {
    return (data[-train_sample, ])
  }
}

data_train <- create_train_test(health2, 0.8, train = TRUE)
data_test <- create_train_test(health2, 0.8, train = FALSE)

## Dimensions of test and training data
dim(data_train)
```

```
## [1] 811   7
```

```r
dim(data_test)
```

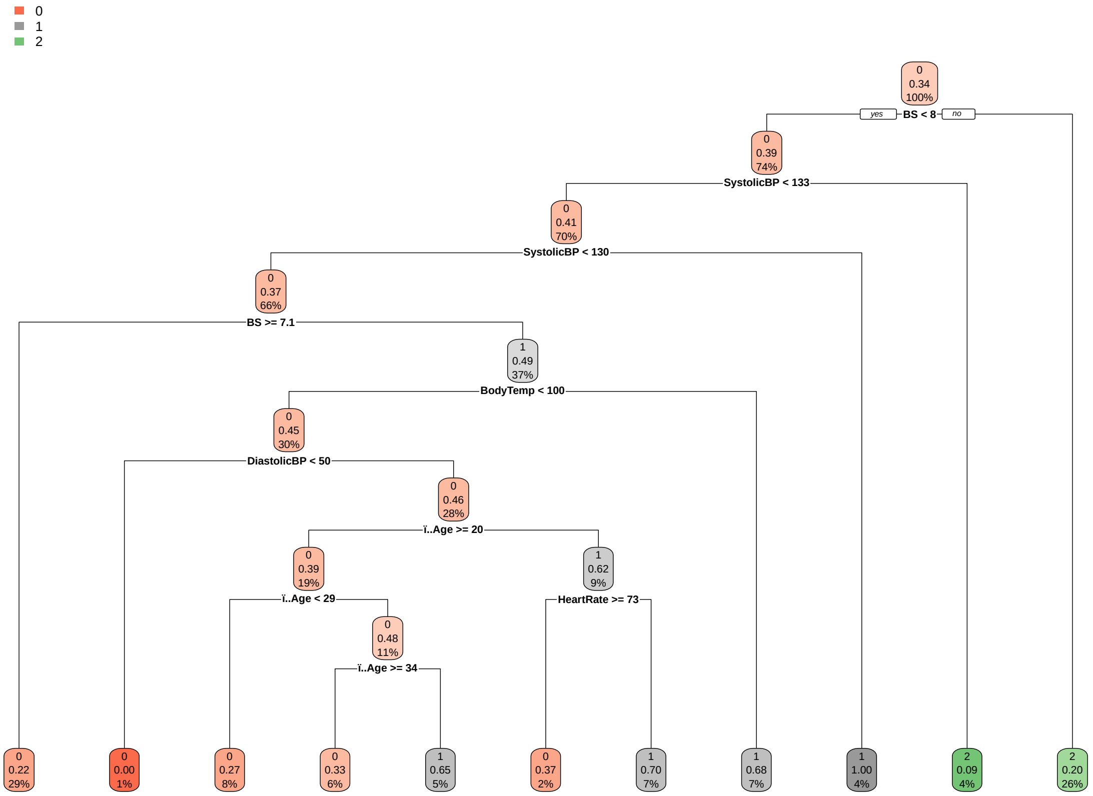
```
## [1] 203   7
```

```r
## Decision tree model

install.packages("rpart.plot",repos ="http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/srava/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)

## package 'rpart.plot' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\srava\AppData\Local\Temp\RtmpW8mwQi\downloaded_packages
```

```r
library(rpart)
library(rpart.plot)
fit <- rpart(RiskLevel~., data = data_train, method = 'class')
rpart.plot(fit, extra = 106)
```

```
## Warning: extra=106 but the response has 3 levels (only the 2nd level is
## displayed)
```



```r
predict_unseen <-predict(fit, data_test, type = 'class')

table_con<- table(data_test$RiskLevel, predict_unseen)
table_con
```

```
##    predict_unseen
##      0  1  2
##   0 75 14  1
##   1 21 23 16
##   2  6  1 46
```

```r
accuracy_Test <- sum(diag(table_con)) / sum(table_con)
```

```r
print(paste('Accuracy for test', accuracy_Test))
```

```
## [1] "Accuracy for test 0.70935960591133"
```

```r
# Installing Packages
install.packages("e1071",repos ="http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/srava/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)

## package 'e1071' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\srava\AppData\Local\Temp\RtmpW8mwQi\downloaded_packages
```

```
install.packages("caTools",repos ="http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/srava/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)

## package 'caTools' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\srava\AppData\Local\Temp\RtmpW8mwQi\downloaded_packages
```

```
install.packages("class",repos ="http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/srava/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)

## package 'class' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\srava\AppData\Local\Temp\RtmpW8mwQi\downloaded_packages
```

```
# Loading package
library(e1071)
library(caTools)
library(class)


#Naive Bayes
# Installing Packages
install.packages("caret",repos ="http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/srava/Documents/R/win-library/4.0'
## (as 'lib' is unspecified)

## package 'caret' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\srava\AppData\Local\Temp\RtmpW8mwQi\downloaded_packages
```

```
# Loading package
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
# Splitting data into train
# and test data
split <- sample.split(health2, SplitRatio = 0.7)
train_cl <- subset(health2, split == "TRUE")
test_cl <- subset(health2, split == "FALSE")

# Feature Scaling
train_scale <- scale(train_cl[, 1:6])
test_scale <- scale(test_cl[, 1:6])
```

```
# Fitting Naive Bayes Model
# to training dataset
set.seed(120) # Setting Seed
classifier_cl <- naiveBayes(RiskLevel ~ ., data = train_cl)
classifier_cl
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1         2
## 0.4034483 0.3465517 0.2500000
##
## Conditional probabilities:
##    ï..Age
## Y        [,1]     [,2]
##   0 26.70940 12.79489
##   1 28.02985 12.31987
##   2 34.95172 12.54178
##
##    SystolicBP
## Y        [,1]     [,2]
##   0 106.6581 15.41203
##   1 112.8358 15.26984
##   2 123.2966 20.90293
##
##    DiastolicBP
## Y        [,1]     [,2]
##   0 72.62821 12.98012
##   1 73.92537 11.12427
##   2 84.57241 14.38737
##
##    BS
## Y         [,1]      [,2]
##   0  7.174487 0.5761944
##   1  7.747861 2.1370678
##   2 11.998828 4.1668667
##
##    BodyTemp
## Y        [,1]     [,2]
##   0 98.37778 1.128569
##   1 98.83085 1.459878
##   2 99.01793 1.652257
##
##    HeartRate
## Y        [,1]     [,2]
##   0 72.74786 8.066282
##   1 74.10448 6.667386
##   2 76.07586 8.434455
```

```
# Predicting on test data
y_pred <- predict(classifier_cl, newdata = test_cl)
# Confusion Matrix
cm <- table(test_cl$RiskLevel, y_pred)
cm
```

```
##    y_pred
##       0   1   2
##   0 154  14   4
##   1  88  31  16
##   2  25  19  83
```

```
confusionMatrix(cm)
```

```
## Confusion Matrix and Statistics
##
##    y_pred
##       0   1   2
##   0 154  14   4
##   1  88  31  16
##   2  25  19  83
##
## Overall Statistics
##
##                Accuracy : 0.6175
##                  95% CI : (0.57, 0.6634)
##     No Information Rate : 0.6152
##     P-Value [Acc > NIR] : 0.4818
##
##                   Kappa : 0.4032
##
##  Mcnemar's Test P-Value : 6.489e-15
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.5768  0.48438   0.8058
## Specificity            0.8922  0.71892   0.8671
## Pos Pred Value         0.8953  0.22963   0.6535
## Neg Pred Value         0.5687  0.88963   0.9349
## Prevalence             0.6152  0.14747   0.2373
## Detection Rate         0.3548  0.07143   0.1912
## Detection Prevalence   0.3963  0.31106   0.2926
## Balanced Accuracy      0.7345  0.60165   0.8364
```

```r
# Model Evaluation
confusionMatrix(cm)
```

```
## Confusion Matrix and Statistics
##
##    y_pred
##       0   1   2
##   0 154  14   4
##   1  88  31  16
##   2  25  19  83
##
## Overall Statistics
##
##                Accuracy : 0.6175
##                  95% CI : (0.57, 0.6634)
##     No Information Rate : 0.6152
##     P-Value [Acc > NIR] : 0.4818
##
##                   Kappa : 0.4032
##
##  Mcnemar's Test P-Value : 6.489e-15
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.5768  0.48438   0.8058
## Specificity            0.8922  0.71892   0.8671
## Pos Pred Value         0.8953  0.22963   0.6535
## Neg Pred Value         0.5687  0.88963   0.9349
## Prevalence             0.6152  0.14747   0.2373
## Detection Rate         0.3548  0.07143   0.1912
## Detection Prevalence   0.3963  0.31106   0.2926
## Balanced Accuracy      0.7345  0.60165   0.8364
```

```r
## knn algorithm
install.packages("e1071",repos ="http://cran.us.r-project.org")
```

```
## Warning: package 'e1071' is in use and will not be installed
```

```r
install.packages("caTools",repos ="http://cran.us.r-project.org")
```

```
## Warning: package 'caTools' is in use and will not be installed
```

```r
install.packages("class",repos ="http://cran.us.r-project.org")
```

```
## Warning: package 'class' is in use and will not be installed
```

```r
library(caret)
library(e1071)
library(caTools)
library(class)
# Splitting the sample
split <- sample.split(health2, SplitRatio = 0.7)
train_cl <- subset(health2, split == "TRUE")
test_cl <- subset(health2, split == "FALSE")
```

```r
# Splitting the sample
split <- sample.split(health2, SplitRatio = 0.7)
train_cl <- subset(health2, split == "TRUE")
test_cl <- subset(health2, split == "FALSE")

# Feature Scaling
train_scale <- scale(train_cl[, 1:6])
test_scale <- scale(test_cl[, 1:6])

classifier_knn <- knn(train = train_scale,test = test_scale,
                      cl = train_cl$RiskLevel,
                      k = 1)
classifier_knn
```

```
##   [1] 2 0 1 1 1 2 0 0 1 0 1 1 2 2 2 2 0 2 2 1 1 0 2 0 1 1 1 0 2 0 2 1 1 1 0 1 2
##  [38] 1 1 1 0 0 0 2 1 1 2 2 1 1 1 1 1 1 0 2 1 0 0 2 1 1 2 2 0 2 0 1 0 0 1 1 0 0
##  [75] 0 1 1 0 0 2 0 1 1 1 0 0 2 1 1 2 1 0 0 1 1 1 2 1 1 0 1 1 2 0 1 0 2 0 1 0 1
## [112] 2 0 1 0 1 2 1 1 0 2 1 2 2 1 2 2 1 2 1 2 0 1 2 2 1 2 1 1 2 2 1 1 1 2 0 1 0
## [149] 0 0 1 0 2 1 1 0 0 1 2 1 1 0 0 0 1 1 2 1 1 0 0 1 0 2 0 1 2 1 0 2 1 1 1 0 1
## [186] 0 1 2 0 0 1 0 1 2 2 1 1 0 0 1 0 1 0 2 0 2 0 1 0 2 1 0 1 1 0 1 1 1 1 0 0 1
## [223] 2 2 0 2 0 0 0 0 1 0 1 1 1 0 0 2 1 1 1 0 0 0 1 2 1 1 0 2 0 1 2 2 1 1 2 2 0
## [260] 0 0 0 1 2 2 0 0 1 1 2 1 1 0 0 0 2 1 0 1 1 1 0 1 1 2 0 1 2 2 2 1 1 1 0 0 0
## [297] 0 1 0 2 1 1 2 1 0 0 1 0 2 1 0 2 0 0 0 0 1 0 0 2 1 1 1 1 2 0 0 0 2 1 1 0 2
## [334] 0 1 2 2 1 1 1 0 2 1 0 0 1 2 0 2 1 2 0 0 2 0 0 2 0 0 1 2 1 1 2 0 2 1 1 0 0
## [371] 0 1 2 1 1 0 2 1 2 1 1 0 1 0 2 1 2 2 1 2 1 2 0 2 2 1 0 1 0 0 1 2 0 2 0 0 0
## [408] 0 1 2 0 1 2 0 0 1 0 0 2 2 2 1 2 1 0 0 0 2 2 0 1 0 0 1 2
## Levels: 0 1 2
```

```r
cm <- table(test_cl$RiskLevel, classifier_knn)
cm
```

```
##    classifier_knn
##       0   1   2
##   0 121  40   9
##   1  25 121   8
##   2   6  11  94
```

```r
#Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_cl$RiskLevel)
print(paste(’Accuracy =’, 1-misClassError))
```

```
## [1] "Accuracy = 0.772413793103448"
```

```
# K = 3
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$RiskLevel,
                      k = 3)
misClassError <- mean(classifier_knn != test_cl$RiskLevel)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.671264367816092"
```

```
# K = 15
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$RiskLevel,
                      k = 15)
misClassError <- mean(classifier_knn != test_cl$RiskLevel)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.67816091954023"
```

```
# K = 19
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$RiskLevel,
                      k = 1)
misClassError <- mean(classifier_knn != test_cl$RiskLevel)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.781609195402299"
```

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.