Forest_Fire_Prediction_Using_Random_Forest.R

DELL

2021-04-12

```r
#Author: James Johnson
#Title: Forest_Fire_Prediction_Using_Random_Forest

#Predicting the Algerian forest fires (Bejaia region) using the Random Forest Classification

#Setting the working directory
setwd('C:/Users/DELL/Desktop/Project_Files/Algerian_Forest_Fires')

#Importing the data and viewing the first few rows
dataset = readxl::read_excel('Algerian_Forest_Fires.xlsx')
head(dataset)
```

```
## # A tibble: 6 x 12
##     Day Temperature    RH    Ws  Rain  FFMC   DMC    DC   ISI   BUI   FWI
##   <dbl>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1          29    57    18   0    65.7   3.4   7.6   1.3   3.4   0.5
## 2     2          29    61    13   1.3  64.4   4.1   7.6   1     3.9   0.4
## 3     3          26    82    22  13.1  47.1   2.5   7.1   0.3   2.7   0.1
## 4     4          25    89    13   2.5  28.6   1.3   6.9   0     1.7   0
## 5     5          27    77    16   0    64.8   3    14.2   1.2   3.9   0.5
## 6     6          31    67    14   0    82.6   5.8  22.2   3.1   7     2.5
## # ... with 1 more variable: Classes <chr>
```

```r
#Converting the dataset into a data frame
dataset = as.data.frame(dataset)

#Removing the unwanted 'Days' column from the dataset
dataset = dataset[-1]
View(dataset)
#Converting 'Classes' into a factor and encoding them
dataset$Classes = factor(dataset$Classes, levels = c('not fire', 'fire'), labels = c(0,1))
dataset
```

```
##     Temperature RH Ws Rain FFMC  DMC    DC  ISI  BUI  FWI Classes
## 1            29 57 18  0.0 65.7  3.4   7.6  1.3  3.4  0.5       0
## 2            29 61 13  1.3 64.4  4.1   7.6  1.0  3.9  0.4       0
## 3            26 82 22 13.1 47.1  2.5   7.1  0.3  2.7  0.1       0
## 4            25 89 13  2.5 28.6  1.3   6.9  0.0  1.7  0.0       0
## 5            27 77 16  0.0 64.8  3.0  14.2  1.2  3.9  0.5       0
## 6            31 67 14  0.0 82.6  5.8  22.2  3.1  7.0  2.5       1
## 7            33 54 13  0.0 88.2  9.9  30.5  6.4 10.9  7.2       1
## 8            30 73 15  0.0 86.6 12.1  38.3  5.6 13.5  7.1       1
## 9            25 88 13  0.2 52.9  7.9  38.8  0.4 10.5  0.3       0
## 10           28 79 12  0.0 73.2  9.5  46.3  1.3 12.6  0.9       0
## 11           31 65 14  0.0 84.5 12.5  54.3  4.0 15.8  5.6       1
## 12           26 81 19  0.0 84.0 13.8  61.4  4.8 17.7  7.1       1
## 13           27 84 21  1.2 50.0  6.7  17.0  0.5  6.7  0.2       0
## 14           30 78 20  0.5 59.0  4.6   7.8  1.0  4.4  0.4       0
## 15           28 80 17  3.1 49.4  3.0   7.4  0.4  3.0  0.1       0
## 16           29 89 13  0.7 36.1  1.7   7.6  0.0  2.2  0.0       0
## 17           30 89 16  0.6 37.3  1.1   7.8  0.0  1.6  0.0       0
## 18           31 78 14  0.3 56.9  1.9   8.0  0.7  2.4  0.2       0
## 19           31 55 16  0.1 79.9  4.5  16.0  2.5  5.3  1.4       0
## 20           30 80 16  0.4 59.8  3.4  27.1  0.9  5.1  0.4       0
## 21           30 78 14  0.0 81.0  6.3  31.6  2.6  8.4  2.2       1
## 22           31 67 17  0.1 79.1  7.0  39.5  2.4  9.7  2.3       0
## 23           32 62 18  0.1 81.4  8.2  47.7  3.3 11.5  3.8       1
## 24           32 66 17  0.0 85.9 11.2  55.8  5.6 14.9  7.5       1
## 25           31 64 15  0.0 86.7 14.2  63.8  5.7 18.3  8.4       1
## 26           31 64 18  0.0 86.8 17.8  71.8  6.7 21.6 10.6       1
## 27           34 53 18  0.0 89.0 21.6  80.3  9.2 25.8 15.0       1
## 28           32 55 14  0.0 89.1 25.5  88.5  7.6 29.7 13.9       1
## 29           32 47 13  0.3 79.9 18.4  84.4  2.2 23.8  3.9       0
## 30           33 50 14  0.0 88.7 22.9  92.8  7.2 28.3 12.9       1
## 31           29 68 19  1.0 59.9  2.5   8.6  1.1  2.9  0.4       0
## 32           27 75 19  1.2 55.7  2.4   8.3  0.8  2.8  0.3       0
## 33           32 76 20  0.7 63.1  2.6   9.2  1.3  3.0  0.5       0
## 34           33 78 17  0.0 80.1  4.6  18.5  2.7  5.7  1.7       0
## 35           33 66 14  0.0 85.9  7.6  27.9  4.8  9.1  4.9       1
## 36           32 63 14  0.0 87.0 10.9  37.0  5.6 12.5  6.8       1
## 37           35 64 18  0.2 80.0  9.7  40.4  2.8 12.1  3.2       0
## 38           33 68 19  0.0 85.6 12.5  49.8  6.0 15.4  8.0       1
## 39           32 68 14  1.4 66.6  7.7   9.2  1.1  7.4  0.6       0
## 40           33 69 13  0.7 66.6  6.0   9.3  1.1  5.8  0.5       0
## 41           33 76 14  0.0 81.1  8.1  18.7  2.6  8.1  2.2       0
## 42           31 75 13  0.1 75.1  7.9  27.7  1.5  9.2  0.9       0
## 43           34 81 15  0.0 81.8  9.7  37.2  3.0 11.7  3.4       0
## 44           34 61 13  0.6 73.9  7.8  22.9  1.4  8.4  0.8       0
## 45           30 80 19  0.4 60.7  5.2  17.0  1.1  5.9  0.5       0
## 46           28 76 21  0.0 72.6  7.0  25.5  0.7  8.3  0.4       0
## 47           29 70 14  0.0 82.8  9.4  34.1  3.2 11.1  3.6       1
## 48           31 68 14  0.0 85.4 12.1  43.1  4.6 14.2  6.0       1
## 49           35 59 17  0.0 88.1 12.0  52.8  7.7 18.2 10.9       1
## 50           33 65 15  0.1 81.4 12.3  62.1  2.8 16.5  4.0       1
## 51           33 70 17  0.0 85.4 18.5  71.5  5.2 22.4  8.8       1
## 52           28 79 18  0.1 73.4 16.4  79.9  1.8 21.7  2.8       0
## 53           27 66 22  0.4 68.2 10.5  71.3  1.8 15.4  2.1       0
## 54           28 78 16  0.1 70.0  9.6  79.7  1.4 14.7  1.3       0
## 55           31 65 18  0.0 84.3 12.5  88.7  4.8 18.5  7.3       1
## 56           36 53 19  0.0 89.2 17.1  98.6 10.0 23.9 15.3       1
## 57           36 48 13  0.0 90.3 22.2 108.5  8.7 29.4 15.3       1
## 58           33 76 15  0.0 86.5 24.4 117.8  5.6 32.1 11.3       1
## 59           32 73 15  0.0 86.6 26.7 127.0  5.6 35.0 11.9       1
## 60           31 79 15  0.0 85.4 28.5 136.0  4.7 37.4 10.7       1
## 61           35 64 17  0.0 87.2 31.9 145.7  6.8 41.2 15.7       1
## 62           36 45 14  0.0 78.8  4.8  10.2  2.0  4.7  0.9       0
## 63           35 55 12  0.4 78.0  5.8  10.0  1.7  5.5  0.8       0
## 64           35 63 14  0.3 76.6  5.7  10.0  1.7  5.5  0.8       0
## 65           34 69 13  0.0 85.0  8.2  19.8  4.0  8.2  3.9       1
## 66           34 65 13  0.0 86.8 11.1  29.7  5.2 11.5  6.1       1
## 67           32 75 14  0.0 86.4 13.0  39.1  5.2 14.2  6.8       1
## 68           32 69 16  0.0 86.5 15.5  48.6  5.5 17.2  8.0       1
## 69           32 60 18  0.3 77.1 11.3  47.0  2.2 14.1  2.6       0
## 70           35 59 17  0.0 87.4 14.8  57.0  6.9 17.9  9.9       1
## 71           35 55 14  0.0 88.9 18.6  67.0  7.4 21.9 11.6       1
## 72           35 63 13  0.0 88.9 21.7  77.0  7.1 25.5 12.1       1
## 73           35 51 13  0.3 81.3 15.6  75.1  2.5 20.7  4.2       0
## 74           35 63 15  0.0 87.0 19.0  85.1  5.9 24.4 10.2       1
## 75           33 66 14  0.0 87.0 21.7  94.7  5.7 27.2 10.6       1
## 76           36 55 13  0.3 82.4 15.6  92.5  3.7 22.0  6.3       1
## 77           36 61 18  0.3 80.2 11.7  90.4  2.8 17.6  4.2       1
## 78           37 52 18  0.0 89.3 16.0 100.7  9.7 22.9 14.6       1
## 79           36 54 18  0.0 89.4 20.0 110.9  9.7 27.5 16.1       1
## 80           35 62 19  0.0 89.4 23.2 120.9  9.7 31.3 17.2       1
## 81           35 68 19  0.0 88.3 25.9 130.6  8.8 34.7 16.8       1
## 82           36 58 19  0.0 88.6 29.6 141.1  9.2 38.8 18.4       1
## 83           36 55 18  0.0 89.1 33.5 151.3  9.9 43.1 20.4       1
## 84           36 53 16  0.0 89.5 37.6 161.5 10.4 47.5 22.3       1
## 85           34 64 14  0.0 88.9 40.5 171.3  9.0 50.9 20.9       1
## 86           35 60 15  0.0 88.9 43.9 181.3  8.2 54.7 20.3       1
## 87           31 78 18  0.0 85.8 45.6 190.6  4.7 57.1 13.7       1
## 88           33 82 21  0.0 84.9 47.0 200.2  4.4 59.3 13.2       1
## 89           34 64 16  0.0 89.4 50.2 210.4  7.3 62.9 19.9       1
## 90           35 48 18  0.0 90.1 54.2 220.4 12.5 67.4 30.2       1
## 91           35 70 17  0.8 72.7 25.2 180.4  1.7 37.4  4.2       0
## 92           28 80 21 16.8 52.5  8.7   8.7  0.6  8.3  0.3       0
## 93           25 76 17  7.2 46.0  1.3   7.5  0.2  1.8  0.1       0
## 94           22 86 15 10.1 30.5  0.7   7.0  0.0  1.1  0.0       0
## 95           25 78 15  3.8 42.6  1.2   7.5  0.1  1.7  0.0       0
## 96           29 73 17  0.1 68.4  1.9  15.7  1.4  2.9  0.5       0
## 97           29 75 16  0.0 80.8  3.4  24.0  2.8  5.1  1.7       1
## 98           29 74 19  0.1 75.8  3.6  32.2  2.1  5.6  0.9       0
## 99           31 71 17  0.3 69.6  3.2  30.1  1.5  5.1  0.6       0
## 100          30 73 17  0.9 62.0  2.6   8.4  1.1  3.0  0.4       0
## 101          30 77 15  1.0 56.1  2.1   8.4  0.7  2.6  0.2       0
## 102          33 73 12  1.8 59.9  2.2   8.9  0.7  2.7  0.3       0
## 103          30 77 21  1.8 58.5  1.9   8.4  1.1  2.4  0.3       0
## 104          29 88 13  0.0 71.0  2.6  16.6  1.2  3.7  0.5       0
## 105          25 86 21  4.6 40.9  1.3   7.5  0.1  1.8  0.0       0
## 106          22 76 26  8.3 47.4  1.1   7.0  0.4  1.6  0.1       0
## 107          24 82 15  0.4 44.9  0.9   7.3  0.2  1.4  0.0       0
## 108          30 65 14  0.0 78.1  3.2  15.7  1.9  4.2  0.8       0
## 109          31 52 14  0.0 87.7  6.4  24.3  6.2  7.7  5.9       1
## 110          32 49 11  0.0 89.4  9.8  33.1  6.8 11.3  7.7       1
## 111          29 57 14  0.0 89.3 12.5  41.3  7.8 14.2  9.7       1
## 112          28 84 18  0.0 83.8 13.5  49.3  4.5 16.0  6.3       1
## 113          31 55 11  0.0 87.8 16.5  57.9  5.4 19.2  8.3       1
## 114          31 50 19  0.6 77.8 10.6  41.4  2.4 12.9  2.8       0
## 115          32 54 11  0.5 73.7  7.9  30.4  1.2  9.6  0.7       0
## 116          29 65 19  0.6 68.3  5.5  15.2  1.5  5.8  0.7       0
## 117          26 81 21  5.8 48.6  3.0   7.7  0.4  3.0  0.1       0
## 118          31 54 11  0.0 82.0  6.0  16.3  2.5  6.2  1.7       0
## 119          31 66 11  0.0 85.7  8.3  24.9  4.0  9.0  4.1       1
## 120          32 47 14  0.7 77.5  7.1   8.8  1.8  6.8  0.9       0
## 121          26 80 16  1.8 47.4  2.9   7.7  0.3  3.0  0.1       0
## 122          25 78 14  1.4 45.0  1.9   7.5  0.2  2.4  0.1       0
```

```r
dim(dataset)
```

```
## [1] 122  11
```

```r
#Splitting the data into training and test sets
set.seed(123)  #The results would then be reproducible
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
train = createDataPartition(y = dataset$Classes, p = 0.75, list = FALSE) #75% into training set
#There is uneven distribution between 'fire' and 'not fire' which might bias the results
#createDataPartition will make sure both training and test sets has balanced observations of classes
training_set = dataset[train,]
test_set = dataset[-train,]
dim(training_set); dim(test_set) #dimensions of training_set and test_set
```

```
## [1] 93 11
```

```
## [1] 29 11
```

```r
head(training_set); head(test_set)
```

```
##    Temperature RH Ws Rain FFMC  DMC   DC ISI  BUI FWI Classes
## 2           29 61 13  1.3 64.4  4.1  7.6 1.0  3.9 0.4       0
## 3           26 82 22 13.1 47.1  2.5  7.1 0.3  2.7 0.1       0
## 5           27 77 16  0.0 64.8  3.0 14.2 1.2  3.9 0.5       0
## 8           30 73 15  0.0 86.6 12.1 38.3 5.6 13.5 7.1       1
## 9           25 88 13  0.2 52.9  7.9 38.8 0.4 10.5 0.3       0
## 10          28 79 12  0.0 73.2  9.5 46.3 1.3 12.6 0.9       0
```

```
##    Temperature RH Ws Rain FFMC DMC   DC ISI  BUI FWI Classes
## 1           29 57 18  0.0 65.7 3.4  7.6 1.3  3.4 0.5       0
## 4           25 89 13  2.5 28.6 1.3  6.9 0.0  1.7 0.0       0
## 6           31 67 14  0.0 82.6 5.8 22.2 3.1  7.0 2.5       1
## 7           33 54 13  0.0 88.2 9.9 30.5 6.4 10.9 7.2       1
## 16          29 89 13  0.7 36.1 1.7  7.6 0.0  2.2 0.0       0
## 22          31 67 17  0.1 79.1 7.0 39.5 2.4  9.7 2.3       0
```

```r
#Conventional method of data splitting (which we won't be using)
#set.seed(1234)
#library(caTools)
#split = sample.split(dataset$Classes, SplitRatio = 0.75)
#training_set = subset(dataset, split == TRUE)
#test_set = subset(dataset, split == FALSE)
#dim(training_set); dim(test_set)
#head(training_set); head(test_set)

#Using the Random Forest classification algorithm from the 'randomForest' package
suppressPackageStartupMessages(library(randomForest))  #supress used to supress messages in output
classifier = randomForest(data = training_set, x = training_set[-11], y = training_set$Classes, ntree = 10)

#Predicting the classifier for the 'test_set'
y_pred = predict(classifier, newdata = test_set, type = 'response')

#Checking model performance on the 'test_set' and checking the accuracy
cm = table(test_set[,11], y_pred)
cm
```

```
##    y_pred
##      0  1
##   0 14  1
##   1  1 13
```

```r
accuracy = (cm[1,1] + cm[2,2])/(cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
round(accuracy*100, 2) #in percentage #gives a 93.1% accuracy
```

```
## [1] 93.1
```

```r
#Another method for checking model performance by using the K-Fold Cross Validation Method
#First loading the 'caret' package (already loaded) and performing the technique
set.seed(1234)   #results would be reproducible
suppressPackageStartupMessages(library(caret))  #to supress any messages
folds = createFolds(training_set$Classes, k = 10)
cv = lapply(folds, function(x){
  training_fold = training_set[-x, ]
  test_fold = training_set[x, ]
  classifier = randomForest(data = training_fold, x = training_fold[-11], y = training_fold$Classes, ntree = 10)
  y_pred = predict(classifier, newdata = test_fold, type = 'response')
  cm = table(test_fold[,11], y_pred)
  accuracy = (cm[1,1]+cm[2,2])/(cm[1,1]+cm[2,2]+cm[1,2]+cm[2,1])
  return(accuracy)
})

#Checking the 20 accuracies
cv
```

```
## $Fold01
## [1] 1
##
## $Fold02
## [1] 1
##
## $Fold03
## [1] 0.8888889
##
## $Fold04
## [1] 1
##
## $Fold05
## [1] 0.8888889
##
## $Fold06
## [1] 1
##
## $Fold07
## [1] 1
##
## $Fold08
## [1] 0.9
##
## $Fold09
## [1] 1
##
## $Fold10
## [1] 0.9
```

```r
#Finding the mean (and final) model accuracy
round(mean(as.numeric(cv))*100, 2)  #gives 92.19% accuracy
```

```
## [1] 95.78
```

```r
#In the above steps, we took all the variables into the model (after excluding the 'Days' column)
#Now, we take only the significant variables by comparing each of them with the 'Classes'
#We do this using the Chi-Square Test and iterate over the variables using a For loop

suppressWarnings(for(i in 1:10){
  print(names(dataset[i]))
  print(chisq.test(dataset[i], dataset$Classes))
}) #SupressWarnings is used to supress the warning messages which might pop up in this code
```

```
## [1] "Temperature"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 35.846, df = 14, p-value = 0.0011
##
## [1] "RH"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 48.407, df = 38, p-value = 0.1201
##
## [1] "Ws"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 19.843, df = 12, p-value = 0.07011
##
## [1] "Rain"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 73.058, df = 24, p-value = 7.455e-07
##
## [1] "FFMC"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 122, df = 100, p-value = 0.06673
##
## [1] "DMC"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 115.33, df = 93, p-value = 0.05823
##
## [1] "DC"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 122, df = 107, p-value = 0.1524
##
## [1] "ISI"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 116.99, df = 66, p-value = 0.0001122
##
## [1] "BUI"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 113.32, df = 98, p-value = 0.138
##
## [1] "FWI"
##
##  Pearson's Chi-squared test
##
## data:  dataset[i] and dataset$Classes
## X-squared = 112.66, df = 70, p-value = 0.0009314
```

Temperature', 'Rain', 'ISI', 'FWI' are the only variables which are significant

```r
#We subset the dataset into these columns and look at them
data = dataset[c(1,4,8,10,11)]
data
```

```
##     Temperature Rain  ISI  FWI Classes
## 1            29  0.0  1.3  0.5       0
## 2            29  1.3  1.0  0.4       0
## 3            26 13.1  0.3  0.1       0
## 4            25  2.5  0.0  0.0       0
## 5            27  0.0  1.2  0.5       0
## 6            31  0.0  3.1  2.5       1
## 7            33  0.0  6.4  7.2       1
## 8            30  0.0  5.6  7.1       1
## 9            25  0.2  0.4  0.3       0
## 10           28  0.0  1.3  0.9       0
## 11           31  0.0  4.0  5.6       1
## 12           26  0.0  4.8  7.1       1
## 13           27  1.2  0.5  0.2       0
## 14           30  0.5  1.0  0.4       0
## 15           28  3.1  0.4  0.1       0
## 16           29  0.7  0.0  0.0       0
## 17           30  0.6  0.0  0.0       0
## 18           31  0.3  0.7  0.2       0
## 19           31  0.1  2.5  1.4       0
## 20           30  0.4  0.9  0.4       0
## 21           30  0.0  2.6  2.2       1
## 22           31  0.1  2.4  2.3       0
## 23           32  0.1  3.3  3.8       1
## 24           32  0.0  5.6  7.5       1
## 25           31  0.0  5.7  8.4       1
## 26           31  0.0  6.7 10.6       1
## 27           34  0.0  9.2 15.0       1
## 28           32  0.0  7.6 13.9       1
## 29           32  0.3  2.2  3.9       0
## 30           33  0.0  7.2 12.9       1
## 31           29  1.0  1.1  0.4       0
## 32           27  1.2  0.8  0.3       0
## 33           32  0.7  1.3  0.5       0
## 34           33  0.0  2.7  1.7       0
## 35           33  0.0  4.8  4.9       1
## 36           32  0.0  5.6  6.8       1
## 37           35  0.2  2.8  3.2       0
## 38           33  0.0  6.0  8.0       1
## 39           32  1.4  1.1  0.6       0
## 40           33  0.7  1.1  0.5       0
## 41           33  0.0  2.6  2.2       0
## 42           31  0.1  1.5  0.9       0
## 43           34  0.0  3.0  3.4       0
## 44           34  0.6  1.4  0.8       0
## 45           30  0.4  1.1  0.5       0
## 46           28  0.0  0.7  0.4       0
## 47           29  0.0  3.2  3.6       1
## 48           31  0.0  4.6  6.0       1
## 49           35  0.0  7.7 10.9       1
## 50           33  0.1  2.8  4.0       1
## 51           33  0.0  5.2  8.8       1
## 52           28  0.1  1.8  2.8       0
## 53           27  0.4  1.8  2.1       0
## 54           28  0.1  1.4  1.3       0
## 55           31  0.0  4.8  7.3       1
## 56           36  0.0 10.0 15.3       1
## 57           36  0.0  8.7 15.3       1
## 58           33  0.0  5.6 11.3       1
## 59           32  0.0  5.6 11.9       1
## 60           31  0.0  4.7 10.7       1
## 61           35  0.0  6.8 15.7       1
## 62           36  0.0  2.0  0.9       0
## 63           35  0.4  1.7  0.8       0
## 64           35  0.3  1.7  0.8       0
## 65           34  0.0  4.0  3.9       1
## 66           34  0.0  5.2  6.1       1
## 67           32  0.0  5.2  6.8       1
## 68           32  0.0  5.5  8.0       1
## 69           32  0.3  2.2  2.6       0
## 70           35  0.0  6.9  9.9       1
## 71           35  0.0  7.4 11.6       1
## 72           35  0.0  7.1 12.1       1
## 73           35  0.3  2.5  4.2       0
## 74           35  0.0  5.9 10.2       1
## 75           33  0.0  5.7 10.6       1
## 76           36  0.3  3.7  6.3       1
## 77           36  0.3  2.8  4.2       1
## 78           37  0.0  9.7 14.6       1
## 79           36  0.0  9.7 16.1       1
## 80           35  0.0  9.7 17.2       1
## 81           35  0.0  8.8 16.8       1
## 82           36  0.0  9.2 18.4       1
## 83           36  0.0  9.9 20.4       1
## 84           36  0.0 10.4 22.3       1
## 85           34  0.0  9.0 20.9       1
## 86           35  0.0  8.2 20.3       1
## 87           31  0.0  4.7 13.7       1
## 88           33  0.0  4.4 13.2       1
## 89           34  0.0  7.3 19.9       1
## 90           35  0.0 12.5 30.2       1
## 91           35  0.8  1.7  4.2       0
## 92           28 16.8  0.6  0.3       0
## 93           25  7.2  0.2  0.1       0
## 94           22 10.1  0.0  0.0       0
## 95           25  3.8  0.1  0.0       0
## 96           29  0.1  1.4  0.5       0
## 97           29  0.0  2.8  1.7       1
## 98           29  0.1  2.1  0.9       0
## 99           31  0.3  1.5  0.6       0
## 100          30  0.9  1.1  0.4       0
## 101          30  1.0  0.7  0.2       0
## 102          33  1.8  0.7  0.3       0
## 103          30  1.8  1.1  0.3       0
## 104          29  0.0  1.2  0.5       0
## 105          25  4.6  0.1  0.0       0
## 106          22  8.3  0.4  0.1       0
## 107          24  0.4  0.2  0.0       0
## 108          30  0.0  1.9  0.8       0
## 109          31  0.0  6.2  5.9       1
## 110          32  0.0  6.8  7.7       1
## 111          29  0.0  7.8  9.7       1
## 112          28  0.0  4.5  6.3       1
## 113          31  0.0  5.4  8.3       1
## 114          31  0.6  2.4  2.8       0
## 115          32  0.5  1.2  0.7       0
## 116          29  0.6  1.5  0.7       0
## 117          26  5.8  0.4  0.1       0
## 118          31  0.0  2.5  1.7       0
## 119          31  0.0  4.0  4.1       1
## 120          32  0.7  1.8  0.9       0
## 121          26  1.8  0.3  0.1       0
## 122          25  1.4  0.2  0.1       0
```

```r
#Splitting the Data
set.seed(123)
#library(caret)
split = createDataPartition(y = data$Classes, p = 0.75, list = FALSE)
train_set = data[split,]
test_set = data[-split,]

#We go through the K-Fold Cross Validation Method to check the new model accuracy
set.seed(123)   #results would be reproducible
fold = createFolds(train_set$Classes, k = 20)
ab = lapply(fold, function(x){
  training_fold1 = train_set[-x, ]
  test_fold1 = train_set[x, ]
  classifier1 = randomForest(x = training_fold1[-5], y = training_fold1$Classes, data = training_fold1, ntree = 1
0)
  y_pred1 = predict(classifier1, newdata = test_fold1, type = 'response')
  cm1 = table(test_fold1[,5], y_pred1)
  accuracy1 = (cm1[1,1] + cm1[2,2]) / (cm1[1,1] + cm1[2,2] + cm1[1,2] + cm1[2,1])
  return(accuracy1)
})

ab
```

```
## $Fold01
## [1] 1
##
## $Fold02
## [1] 1
##
## $Fold03
## [1] 1
##
## $Fold04
## [1] 1
##
## $Fold05
## [1] 1
##
## $Fold06
## [1] 1
##
## $Fold07
## [1] 1
##
## $Fold08
## [1] 1
##
## $Fold09
## [1] 1
##
## $Fold10
## [1] 1
##
## $Fold11
## [1] 0.8333333
##
## $Fold12
## [1] 1
##
## $Fold13
## [1] 1
##
## $Fold14
## [1] 1
##
## $Fold15
## [1] 1
##
## $Fold16
## [1] 0.8
##
## $Fold17
## [1] 0.75
##
## $Fold18
## [1] 0.75
##
## $Fold19
## [1] 1
##
## $Fold20
## [1] 1
```

```r
#Calculating the final accuracy
round(mean(as.numeric(ab))*100,2)
```

```
## [1] 95.67
```

```r
#The accuracy increases but only slightly (to 95.67%), but this is a good model anyway with an accuracy of more t
han 90% consistently
```