Coursera – Prediction Assignment Writeup

Giorgio Bresciani – 18/04/2020

Project execution

According to the goal of the project (practical machine learning assignment), I am requested to develop a model through a machine learning approach to predict the target variable "classe", using other variables which can have a statistical relevance with respect to the target variable.

I performed a data analysis to prepare the data set properly for the training phase and the validation phase to assess the performance of the models. I used different R methods to evaluate which of the models has better performance for the prediction of the target variable, by comparing the outcomes obtained. Using the validation data set provided, I predicted the target variable of 20 different cases with the final model.

DATA INGESTION

Setting up of R libraries and packages necessary to run the code.

```
# Set libraries

library(knitr)

library(caret)

library(rpart)

library(rpart.plot)

library(rattle)

library(randomForest)

library(corrplot)

library(RColorBrewer)
```

Setting up of URL to download training data and test data.

```
# Set the URL for the download from external link

UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"


# Download the datasets

training_data <- read.csv(url(UrlTrain))

testing_data  <- read.csv(url(UrlTest))
```

Creation of two partition of data by splitting the training data set into two different parts, with a proportion of 70% for the training data set and 30% for the test data set.

I obtained a training data set with 13737 rows and 160 columns, 5885 are the remaining rows to be used as test data.

```
# Create a partition with the training dataset

set.seed(14042020)

inTrain  <- createDataPartition(training_data$classe, p=0.7, list=FALSE)

TrainDataSet <- training_data[inTrain, ]

TestDataSet  <- training_data[-inTrain, ]

dim(TrainDataSet)

dim(TestDataSet)
```

```
> # Create a partition with the training dataset
> inTrain  <- createDataPartition(training_data$classe, p=0.7, list=FALSE)
> TrainDataSet <- training_data[inTrain, ]
> TestDataSet  <- training_data[-inTrain, ]
> dim(TrainDataSet)
[1] 13737   160
> dim(TestDataSet)
[1] 5885  160
```

DATA CLEANING

Data cleaning phases necessary to remove variables with near zero variance, not relevant for the statistical analysis. After removing variables with near zero variance, I obtained 105 columns, hence, 55 variables have been removed due to the variance near to zero.

```
# Remove variables with Nearly Zero Variance

NZV <- nearZeroVar(TrainDataSet)

TrainDataSet <- TrainDataSet[, -NZV]

TestDataSet  <- TestDataSet[, -NZV]

dim(TrainDataSet)

dim(TestDataSet)
```

```
> # Remove variables with Nearly Zero Variance
> NZV <- nearZeroVar(TrainDataSet)
> TrainDataSet <- TrainDataSet[, -NZV]
> TestDataSet  <- TestDataSet[, -NZV]
> dim(TrainDataSet)
[1] 13737   105
> dim(TestDataSet)
[1] 5885  105
```

I removed also columns which contains mostly "not available/missing" value (columns with a number of rows with "not available/missing" higher or equal than 95% of total are excluded). After removing variables with non-relevant information, I obtained 59 columns.

```
# Remove variables which contain mostly missing values
```

```
ColIndex <- colSums(is.na(TrainDataSet))/nrow(TrainDataSet) < 0.95

TrainDataSet <- TrainDataSet[,ColIndex]

TestDataSet <- TestDataSet[,ColIndex]

dim(TrainDataSet)

dim(TestDataSet)
```

```
> # Remove variables which contain mostly missing value
>
> ColIndex <- colSums(is.na(TrainDataSet))/nrow(TrainDataSet) < 0.95
> TrainDataSet <- TrainDataSet[,ColIndex]
> TestDataSet <- TestDataSet[,ColIndex]
> dim(TrainDataSet)
[1] 13737    59
> dim(TestDataSet)
[1] 5885   59
```

I completed the data cleaning phase by removing also columns related to information not relevant for this analysis (time stamp, name). After removing these latter variables, I obtained 52 columns.

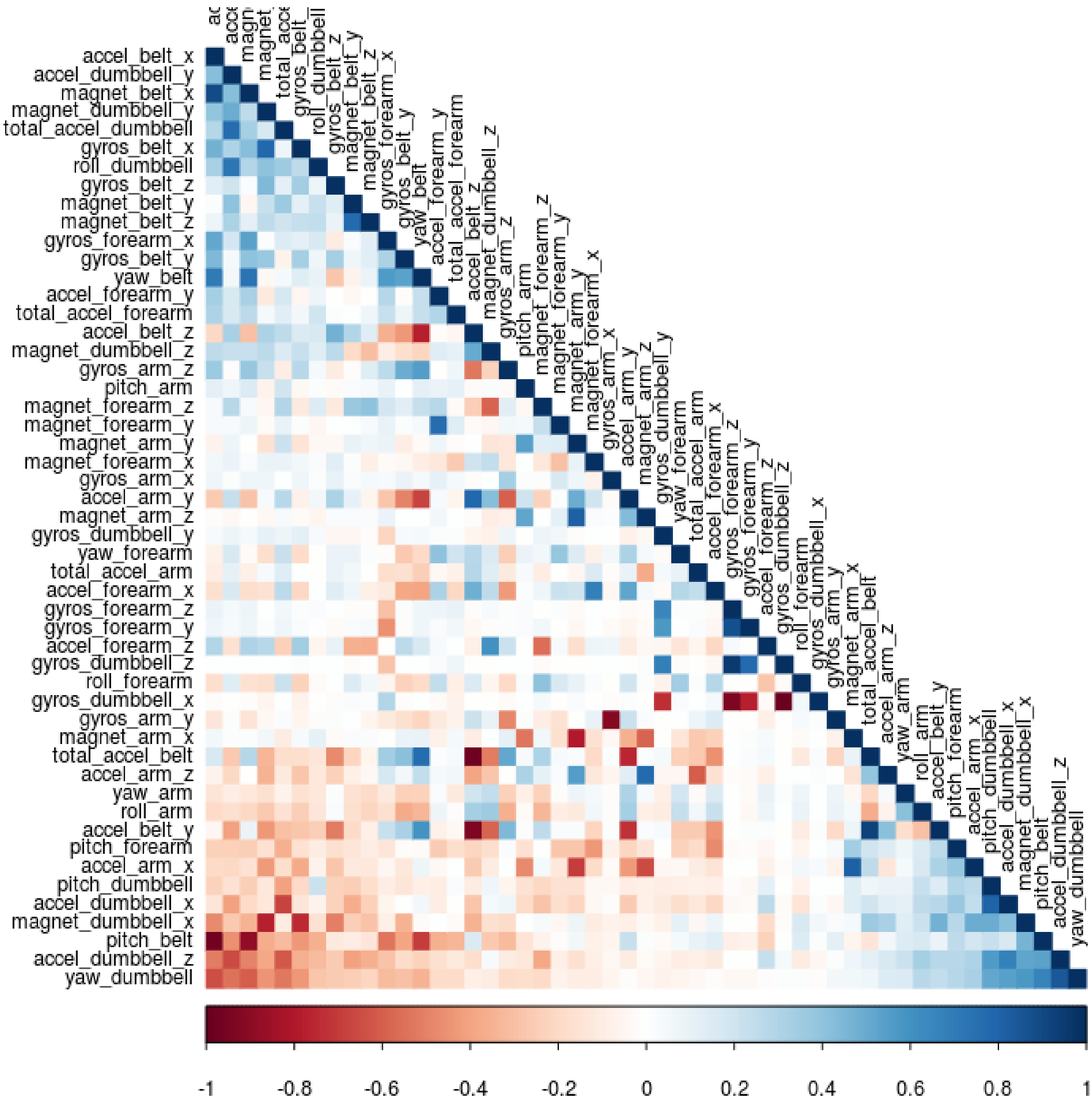
```
# Remove identification only variables (columns 1 to 7)

TrainDataSet <- TrainDataSet[, -(1:7)]

TestDataSet  <- TestDataSet[, -(1:7)]

dim(TrainDataSet)

dim(TestDataSet)
```

```
> # Remove first seven variables (columns 1 to 7)
> TrainDataSet <- TrainDataSet[, -(1:7)]
> TestDataSet  <- TestDataSet[, -(1:7)]
> dim(TrainDataSet)
[1] 13737    52
> dim(TestDataSet)
[1] 5885   52
```

CORRELATION ANALYSIS

I performed a correlation analysis to find the highly correlated variables and plotting the overall correlation analysis. Darker colours in the graph below represents the highly correlated variables.

```
# Correlation analysis

corMat <- cor(TrainDataSet[, -52])

corrplot(corMat, order = "FPC", method = "color", type = "lower",

         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



Below the highly correlated variables using a cutoff of 90%.

```
highlyCorr = findCorrelation(corMat, cutoff=0.9)
```

```
names(TrainDataSet)[highlyCorr]
```

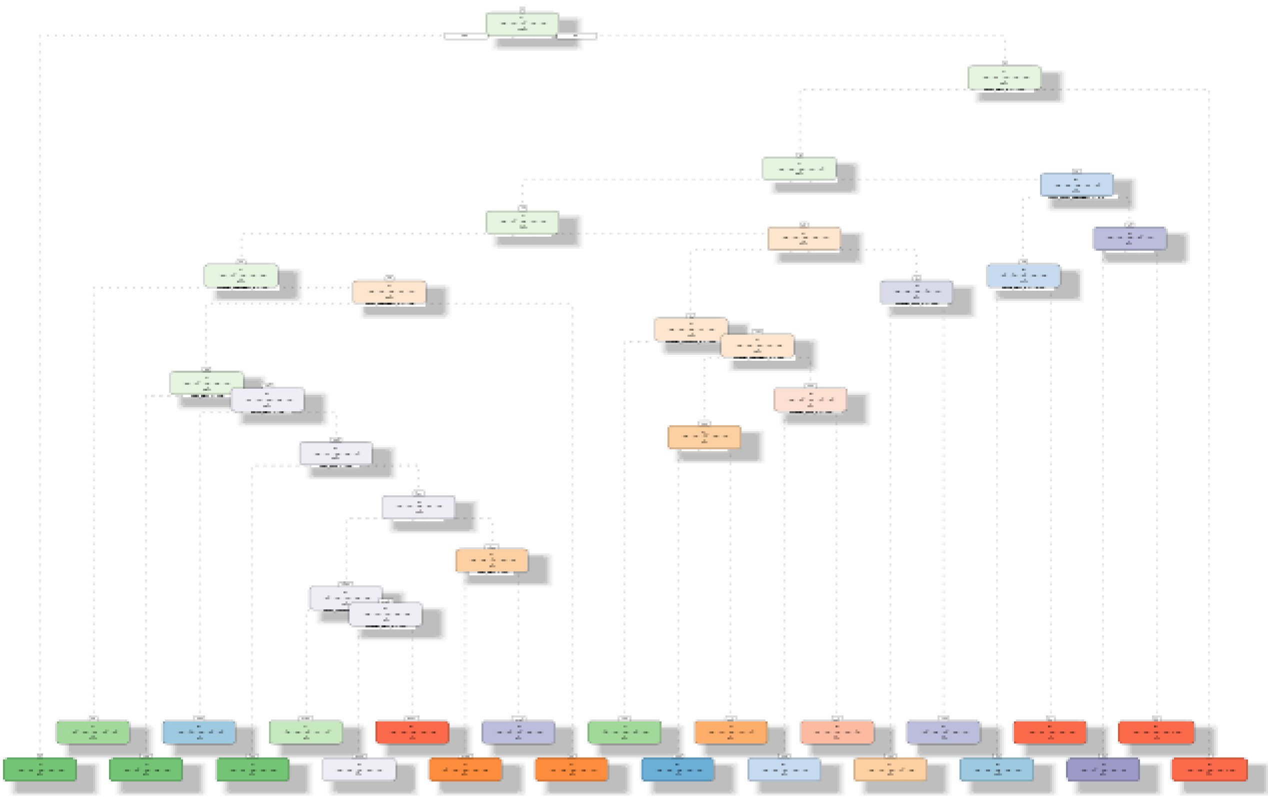
```
> names(TrainDataSet)[highlyCorr]
[1] "accel_belt_z"     "accel_belt_y"     "accel_belt_x"     "gyros_dumbbell_x"
[5] "gyros_dumbbell_z" "gyros_arm_x"
```

MODELS DESIGN

I performed two different analysis using different model methods to evaluate which model has best performance on this data.

As first model, I performed a decision tree model by applying the training dataset created previously.

```
#Decision Tree Model

set.seed(14042020)

decisionTreeModel <- rpart(classe ~ ., data=TrainDataSet, method="class")

fancyRpartPlot(decisionTreeModel)
```



Rattle 2020-Apr-18 15:48:05 rstudio-user

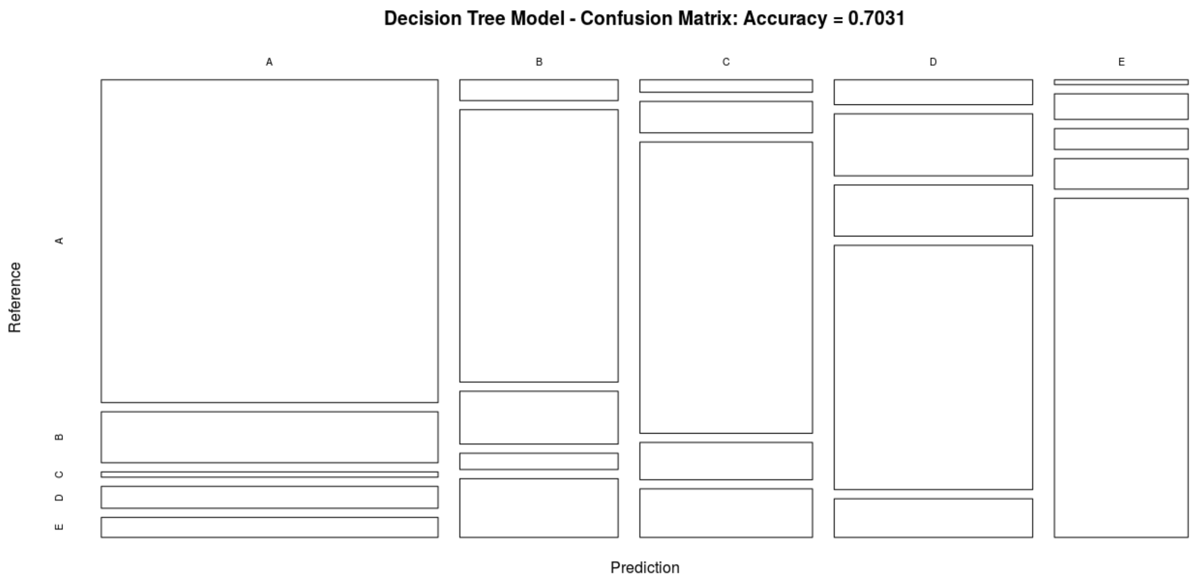
I evaluated the performance of the model obtained using the test data set created previously and checking the accuracy through a confusion matrix applied to the prediction.

```
predictTreeModel <- predict(decisionTreeModel, TestDataSet, type = "class")

ConfMatrixTree <- confusionMatrix(predictTreeModel, TestDataSet$classe)

ConfMatrixTree
```

```
> predictTreeModel <- predict(decisionTreeModel, TestDataSet, type = "class")
> ConfMatrixTree <- confusionMatrix(predictTreeModel, TestDataSet$classe)
> ConfMatrixTree
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1520  240   25  103   94
         B   46  603  117   36  130
         C   30   76  703   90  117
```

```
         D   69  172  142  678  107
         E    9   48   39   57  634

Overall Statistics

               Accuracy : 0.7031
                 95% CI : (0.6913, 0.7148)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6225

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9080   0.5294   0.6852   0.7033   0.5860
Specificity            0.8903   0.9307   0.9356   0.9004   0.9681
Pos Pred Value         0.7669   0.6470   0.6919   0.5805   0.8056
Neg Pred Value         0.9605   0.8918   0.9337   0.9394   0.9121
Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
Detection Rate         0.2583   0.1025   0.1195   0.1152   0.1077
Detection Prevalence   0.3368   0.1584   0.1726   0.1985   0.1337
Balanced Accuracy      0.8991   0.7300   0.8104   0.8019   0.7770
```

```
plot(ConfMatrixTree$table, col = ConfMatrixTree$byClass, main = paste("Decision Tree
Model - Confusion Matrix: Accuracy =", round(ConfMatrixTree$overall['Accuracy'], 4)))
```



According to the outcomes obtained above, the decision tree model has an **accuracy of 70.31%.**

I performed a random forest model using the same data set to get a better accuracy.

```
#random forest model

set.seed(14042020)
```

```
TrainControlRF <- trainControl(method = "cv",

                        number = 3,

                        allowParallel = TRUE,

                        verboseIter = TRUE)

ModelFitRandomForest <- train(classe ~ ., data=TrainDataSet, method="rf",

                          trControl=TrainControlRF,ntree=100,importance=TRUE)

ModelFitRandomForest$finalModel

varImp(ModelFitRandomForest)
```

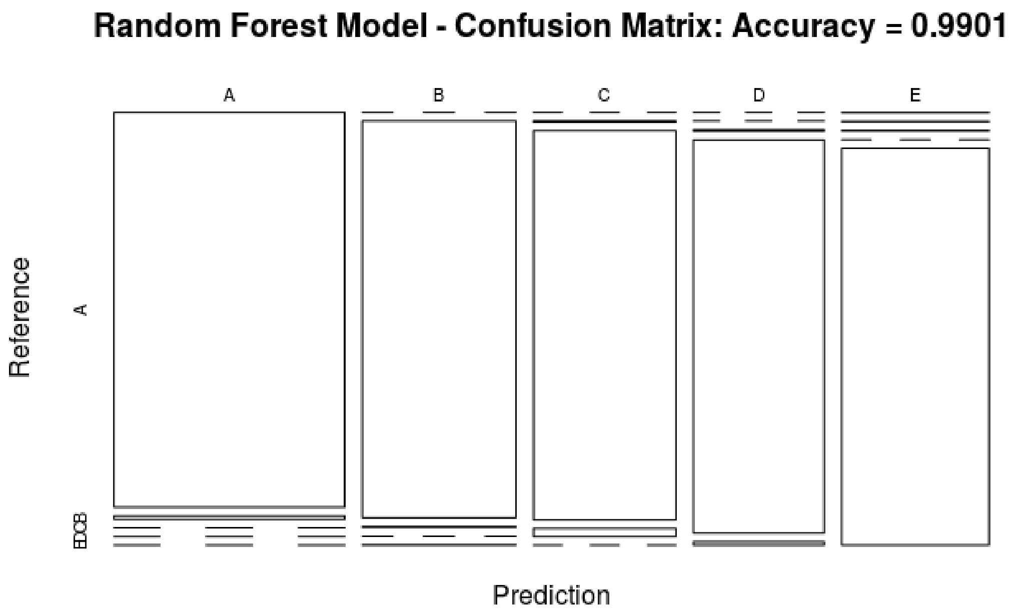
```
> ModelFitRandomForest$finalModel

Call:
 randomForest(x = x, y = y, ntree = 100, mtry = param$mtry, importance = TRUE) 
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 26

        OOB estimate of  error rate: 0.77%
Confusion matrix:
     A    B    C    D    E class.error
A 3898    6    1    0    1 0.002048131
B   24 2625    7    0    2 0.012415350
C    0   17 2375    4    0 0.008764608
D    1    1   25 2223    2 0.012877442
E    0    1    4   10 2510 0.005940594
> 
> varImp(ModelFitRandomForest)
rf variable importance

  variables are sorted by maximum importance across the classes
  only 20 most important variables shown (out of 51)

                          A     B     C     D     E
yaw_belt             100.00 81.37 64.96 95.70 78.80
pitch_belt            24.94 90.82 57.60 50.22 46.71
pitch_forearm         63.10 76.91 88.80 48.58 71.56
magnet_dumbbell_z     82.70 63.58 79.38 66.22 73.04
magnet_dumbbell_y     69.02 59.21 77.45 52.65 52.58
gyros_belt_z          28.72 47.78 33.10 26.28 42.89
accel_belt_z          34.68 41.41 41.90 45.62 27.63
accel_forearm_x       21.98 36.35 32.10 37.61 32.49
roll_forearm          37.03 35.35 35.71 23.38 32.04
magnet_belt_x         15.94 33.83 24.40 18.30 35.46
gyros_arm_y           24.35 33.33 20.37 28.17 24.30
yaw_arm               33.30 29.16 24.11 26.61 24.06
accel_dumbbell_z      22.71 32.51 18.00 25.04 26.45
gyros_dumbbell_y      31.83 15.14 27.81 18.72 13.52
accel_dumbbell_y      23.25 24.44 31.79 23.31 30.54
roll_dumbbell         18.16 31.51 23.25 27.19 24.45
magnet_arm_z          14.38 29.72 22.10 16.70 15.54
total_accel_dumbbell  17.33 24.47 17.55 21.83 28.94
gyros_forearm_y       11.26 19.32 28.77 11.66 14.83
magnet_belt_z         21.10 28.07 23.92 28.57 27.21
```

I evaluated the performance of the model obtained using the test data set created previously and checking the accuracy through a confusion matrix applied to the prediction.

```
PredictRF <- predict(ModelFitRandomForest, newdata=TestSet)

ConfMatrixRF <- confusionMatrix(PredictRF, TestSet$classe)

ConfMatrixRF
```

```
> PredictRF <- predict(ModelFitRandomForest, newdata=TestDataSet)
> ConfMatrixRF <- confusionMatrix(PredictRF, TestDataSet$classe)
> ConfMatrixRF
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1673   14    0    0    0
         B    0 1121    3    0    1
         C    0    3 1017   21    0
         D    0    0    4  943    8
         E    1    1    2    0 1073

Overall Statistics
                                          
               Accuracy : 0.9901          
                 95% CI : (0.9873, 0.9925)
    No Information Rate : 0.2845          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.9875          
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9994   0.9842   0.9912   0.9782   0.9917
Specificity            0.9967   0.9992   0.9951   0.9976   0.9992
Pos Pred Value         0.9917   0.9964   0.9769   0.9874   0.9963
Neg Pred Value         0.9998   0.9962   0.9981   0.9957   0.9981
Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
Detection Rate         0.2843   0.1905   0.1728   0.1602   0.1823
Detection Prevalence   0.2867   0.1912   0.1769   0.1623   0.1830
Balanced Accuracy      0.9980   0.9917   0.9931   0.9879   0.9954
```

```
plot(ConfMatrixRF$table, col = ConfMatrixRF$byClass, main = paste("Random Forest Model - Confusion Matrix: Accuracy =", round(ConfMatrixRF$overall['Accuracy'], 4)))
```



According to the outcomes obtained above, the random forest model has an **accuracy of 99.01%;** The accuracy ratio of random forest model is significantly higher than the one of the decision tree model (70.31%).

I can assume that the random forest approach is the best method to predict the target variable of the dataset provided for this project.

Finally, I submitted the final prediction applying the random forest model to the new data provided (test data). Below is showed the outcome of the prediction performed.

```
#Final prediction on Test data

predict(ModelFitRandomForest, newdata = testing_data)
```

```
> #Final prediction on Test data
> predict(ModelFitRandomForest, newdata = testing_data)
 [1] B A B A A E D B A A B C B A E E A B B B
Levels: A B C D E
```