Practical Machine Learning Project

Pradeep Pal

May 22, 2018

I. Overview

The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the “classe” variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

II. Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX (http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX)

III. Data Loading and Exploratory Analysis

a) Dataset Overview

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Full source:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. “Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13)”. Stuttgart, Germany: ACM SIGCHI, 2013.

My special thanks to the above mentioned authors for being so generous in allowing their data to be used for this kind of assignment.

A short description of the datasets content from the authors’ website:

“Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg)."

b) Environment Preparation

Here is the R libraries that are necessary for the complete analysis.

- library(knitr)
- library(caret)
- library(rpart)
- library(rpart.plot)
- library(rattle)
- library(randomForest)
- library(corrplot)
- set.seed(12345)

c) Data Loading and Cleaning

The next step is loading the dataset from the URL provided above. The training dataset is then partinioned in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations. The testing dataset is not changed and will only be used for the quiz results generation.

```
#Upload the R libraries

library(knitr)
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(tidyr)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:rattle':
##
##     xgboost
```

```
library(Rtsne)
library(stats)
library(ggplot2)
library(data.table)
library(curl)
set.seed(12345)
```

```
# Setting path of training and testing dataset.
train_data = read.csv("C:/Users/Pradeep/Documents/Coursera/ML_Project/pml-training.cs
v")
test_data = read.csv("C:/Users/Pradeep/Documents/Coursera/ML_Project/pml-testing.csv")

# create a partition with the training dataset
inTrain  <- createDataPartition(train_data$classe, p=0.7, list=FALSE)
TrainSet <- train_data[inTrain, ]
TestSet  <- train_data[-inTrain, ]
dim(TrainSet)
```

```
## [1] 13737   160
```

```
dim(TestSet)
```

```
## [1] 5885  160
```

```
names(TrainSet)
```

```
##   [1] "X"                       "user_name"
##   [3] "raw_timestamp_part_1"    "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"          "new_window"
##   [7] "num_window"              "roll_belt"
##   [9] "pitch_belt"              "yaw_belt"
##  [11] "total_accel_belt"        "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"     "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"      "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"       "max_roll_belt"
##  [19] "max_picth_belt"          "max_yaw_belt"
##  [21] "min_roll_belt"           "min_pitch_belt"
##  [23] "min_yaw_belt"            "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"    "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"    "avg_roll_belt"
##  [29] "stddev_roll_belt"        "var_roll_belt"
##  [31] "avg_pitch_belt"          "stddev_pitch_belt"
##  [33] "var_pitch_belt"          "avg_yaw_belt"
##  [35] "stddev_yaw_belt"         "var_yaw_belt"
##  [37] "gyros_belt_x"            "gyros_belt_y"
##  [39] "gyros_belt_z"            "accel_belt_x"
##  [41] "accel_belt_y"            "accel_belt_z"
##  [43] "magnet_belt_x"           "magnet_belt_y"
##  [45] "magnet_belt_z"           "roll_arm"
##  [47] "pitch_arm"               "yaw_arm"
##  [49] "total_accel_arm"         "var_accel_arm"
##  [51] "avg_roll_arm"            "stddev_roll_arm"
##  [53] "var_roll_arm"            "avg_pitch_arm"
##  [55] "stddev_pitch_arm"        "var_pitch_arm"
##  [57] "avg_yaw_arm"             "stddev_yaw_arm"
##  [59] "var_yaw_arm"             "gyros_arm_x"
##  [61] "gyros_arm_y"             "gyros_arm_z"
##  [63] "accel_arm_x"             "accel_arm_y"
##  [65] "accel_arm_z"             "magnet_arm_x"
##  [67] "magnet_arm_y"            "magnet_arm_z"
##  [69] "kurtosis_roll_arm"       "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"        "skewness_roll_arm"
##  [73] "skewness_pitch_arm"      "skewness_yaw_arm"
##  [75] "max_roll_arm"            "max_picth_arm"
##  [77] "max_yaw_arm"             "min_roll_arm"
##  [79] "min_pitch_arm"           "min_yaw_arm"
##  [81] "amplitude_roll_arm"      "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"       "roll_dumbbell"
##  [85] "pitch_dumbbell"          "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"  "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"   "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell" "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"       "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"        "min_roll_dumbbell"
```

```
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

```
# remove variables with Nearly Zero Variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TrainSet)
```

```
## [1] 13737   106
```

```
dim(TestSet)
```

```
## [1] 5885  106
```

```
# remove variables that are mostly NA
AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]
dim(TrainSet)
```

```
## [1] 13737    59
```

```
dim(TestSet)
```

```
## [1] 5885   59
```

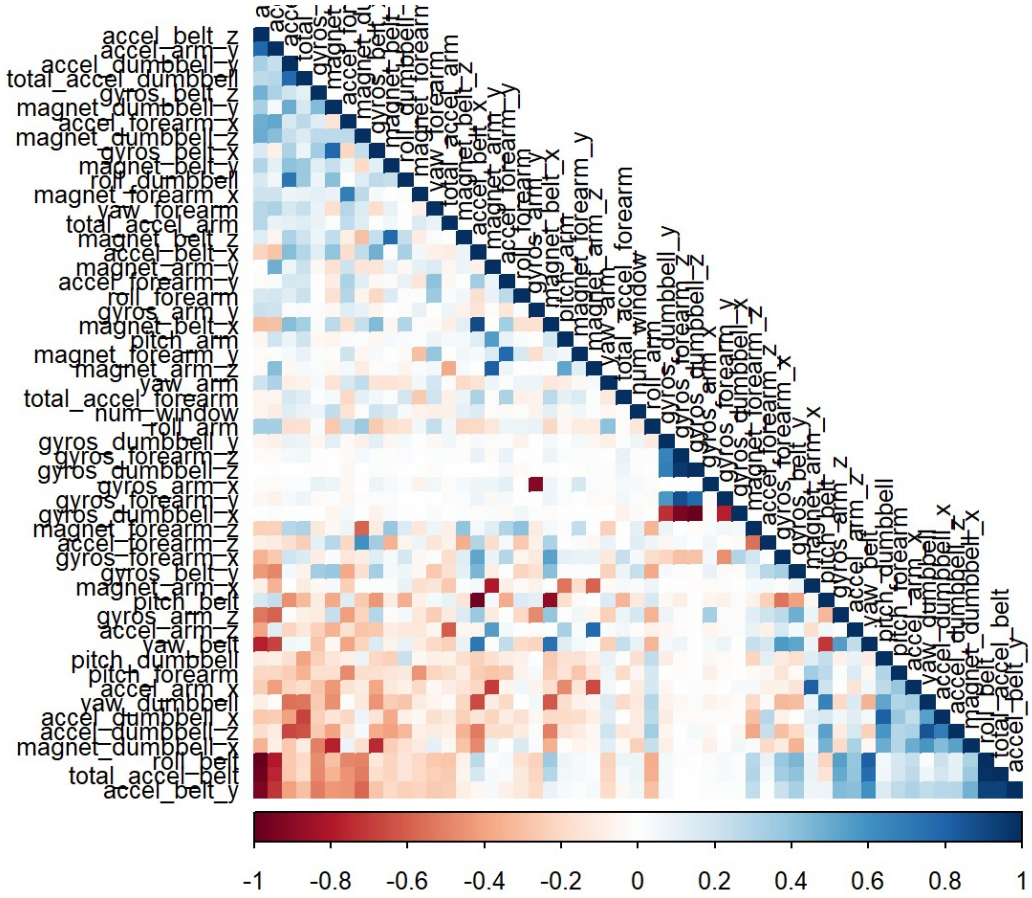
```
# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

```
## [1] 13737    54
```

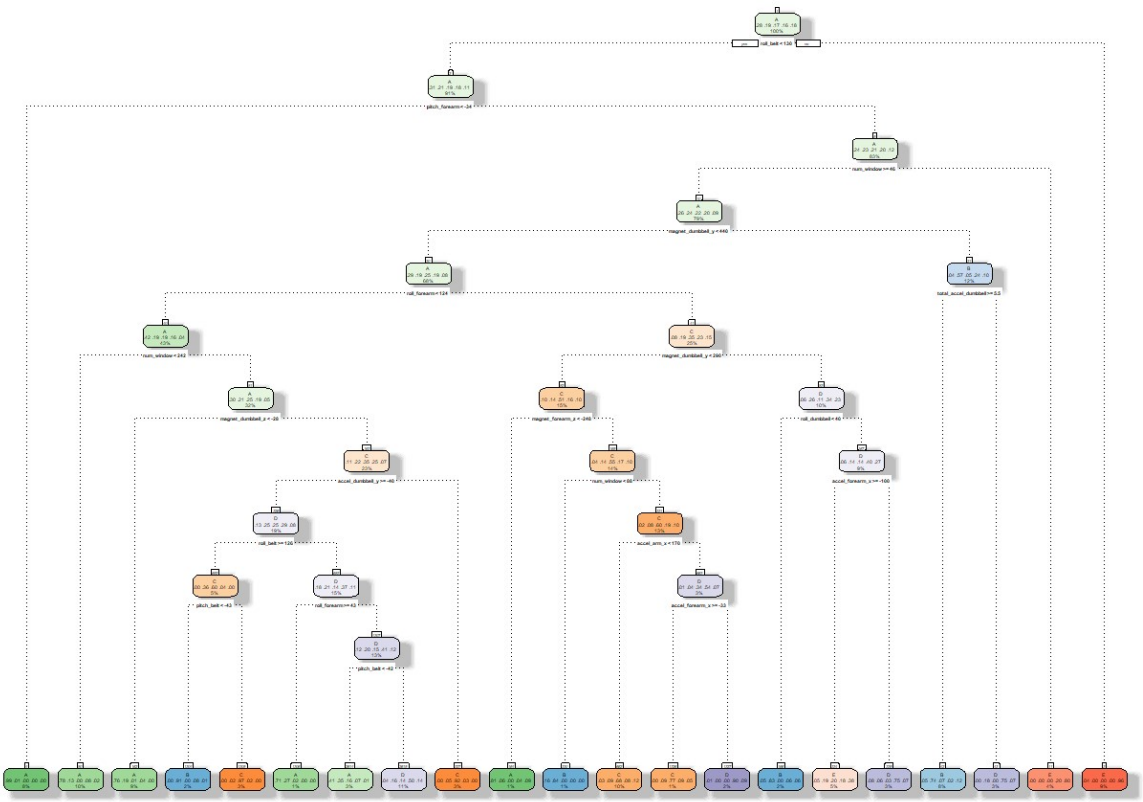
```
dim(TestSet)
```

```
## [1] 5885   54
```

```
# correlation analysis
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```




```
# model fit
set.seed(12345)
modFitA1 <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitA1)
```

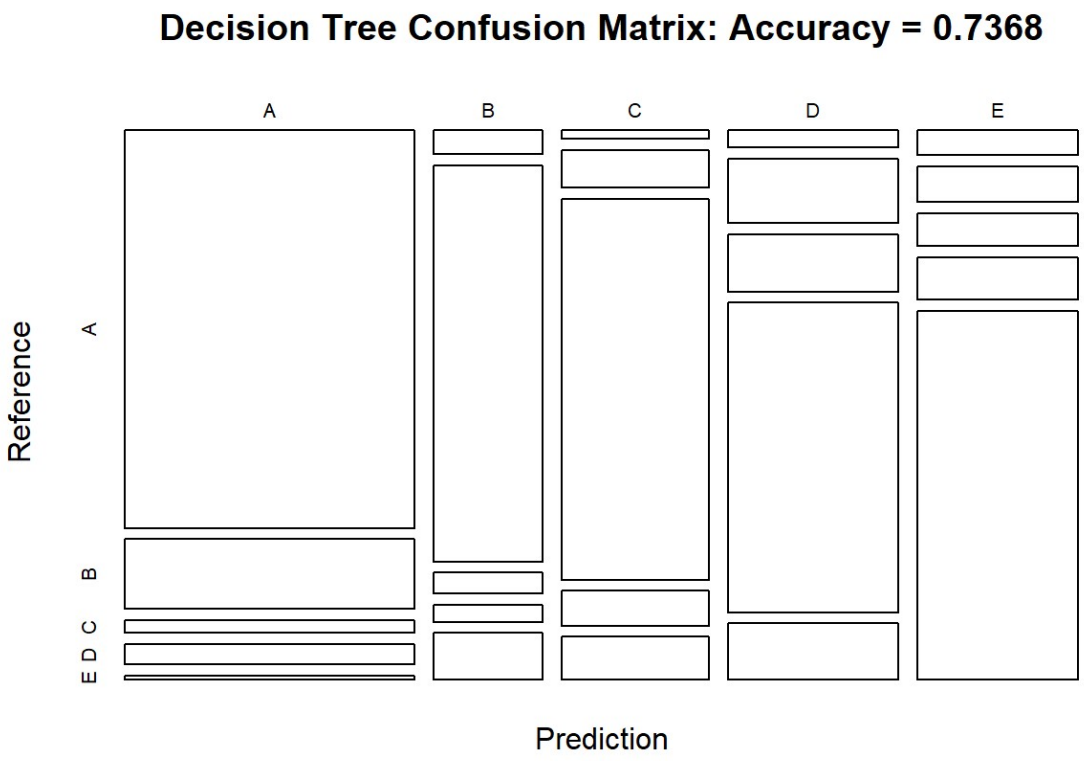


Rattle 2018-May-22 18:01:48 Pradeep

```
predictionsA1 <- predict(modFitA1, TestSet, type = "class")
cmtree <- confusionMatrix(predictionsA1, TestSet$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  269   51   79   16
##          B   35  575   31   25   68
##          C   17   73  743   68   84
##          D   39  146  130  702  128
##          E   53   76   71   90  786
##
## Overall Statistics
##
##                Accuracy : 0.7368
##                  95% CI : (0.7253, 0.748)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6656
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.50483   0.7242   0.7282   0.7264
## Specificity            0.9014  0.96650   0.9502   0.9100   0.9396
## Pos Pred Value         0.7866  0.78338   0.7543   0.6131   0.7305
## Neg Pred Value         0.9635  0.89051   0.9422   0.9447   0.9384
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2600  0.09771   0.1263   0.1193   0.1336
## Detection Prevalence   0.3305  0.12472   0.1674   0.1946   0.1828
## Balanced Accuracy      0.9077  0.73566   0.8372   0.8191   0.8330
```

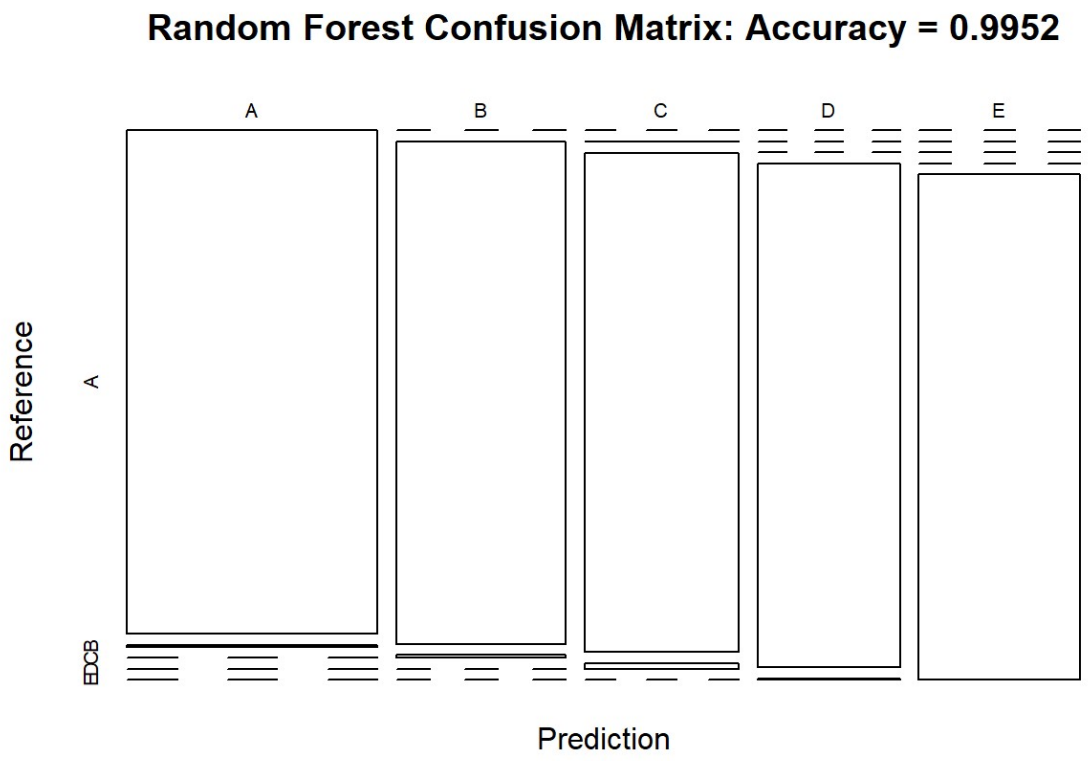
```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree Confusion Matrix:
Accuracy =", round(cmtree$overall['Accuracy'], 4)))
```




```
#Prediction with Random Forests
set.seed(12345)
modFitB1 <- randomForest(classe ~ ., data=TrainSet)
predictionB1 <- predict(modFitB1, TestSet, type = "class")
cmrf <- confusionMatrix(predictionB1, TestSet$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1132    6    0    0
##          C    0    1 1020   12    0
##          D    0    0    0  952    3
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9931, 0.9968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9939   0.9942   0.9876   0.9972
## Specificity            0.9986   0.9987   0.9973   0.9994   1.0000
## Pos Pred Value         0.9964   0.9947   0.9874   0.9969   1.0000
## Neg Pred Value         1.0000   0.9985   0.9988   0.9976   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1733   0.1618   0.1833
## Detection Prevalence   0.2855   0.1934   0.1755   0.1623   0.1833
## Balanced Accuracy      0.9993   0.9963   0.9957   0.9935   0.9986
```

```
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest Confusion Matrix: A
ccuracy =", round(cmrf$overall['Accuracy'], 4)))
```



IV. Applying the Selected Model to the Test Data

The accuracy of the 2 regression modeling methods above are:

a. Decision Tree : 0.7368
b. Random Forest : 0.9952

Predicting Results on the Test Data

Random Forests gave an Accuracy in the myTesting dataset of 99.52%, which was more accurate that what I got from the Decision Trees . The expected out-of-sample error is 100-99.52 = 0.48%.

In that case, the Random Forest model will be applied to predict the 20 quiz results (testing dataset) as shown below.