PML_course_project

Huiwu Zhao

7/1/2019

PML Course Project: Prediction Assignment Writeup

I. 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.
* Exactly according to the specification (Class A)
* Throwing the elbows to the front (Class B) - mistake
* Lifting the dumbbell only halfway (Class C) - mistake
* Lowering the dumbbell only halfway (Class D) - mistake
* Throwing the hips to the front (Class E) - mistake

More information is available from the website here:
http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

II. Goals

To predict the manner in which they did the exercise.
To create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did.
You will also use your prediction model to predict 20 different test cases.

III. Loading Data and exploratory analysis

1. Datasets 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. 2)R environment preparation

```
rm(list=ls())
setwd("~/Desktop/coursera/practicalML")
packages<-c("caret","rpart","rpart.plot","rattle","randomForest","corrplot")
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
     install.packages(setdiff(packages, rownames(installed.packages())),repos = "http://
cran.us.r-project.org")
 }
lapply(packages,library,character.only=TRUE)
```

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

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

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## 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
```

```
## corrplot 0.84 loaded
```

```
## [[1]]
##  [1] "caret"     "ggplot2"   "lattice"   "stats"     "graphics"
##  [6] "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[2]]
##  [1] "rpart"     "caret"     "ggplot2"   "lattice"   "stats"
##  [6] "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [11] "base"
##
## [[3]]
##  [1] "rpart.plot" "rpart"      "caret"      "ggplot2"    "lattice"
##  [6] "stats"      "graphics"   "grDevices"  "utils"      "datasets"
## [11] "methods"    "base"
##
## [[4]]
##  [1] "rattle"     "rpart.plot" "rpart"      "caret"      "ggplot2"
##  [6] "lattice"    "stats"      "graphics"   "grDevices"  "utils"
## [11] "datasets"   "methods"    "base"
##
## [[5]]
##  [1] "randomForest" "rattle"       "rpart.plot"   "rpart"
##  [5] "caret"        "ggplot2"      "lattice"      "stats"
##  [9] "graphics"     "grDevices"    "utils"        "datasets"
## [13] "methods"      "base"
##
## [[6]]
##  [1] "corrplot"     "randomForest" "rattle"       "rpart.plot"
##  [5] "rpart"        "caret"        "ggplot2"      "lattice"
##  [9] "stats"        "graphics"     "grDevices"    "utils"
## [13] "datasets"     "methods"      "base"
```

3)Loading and cleaning data

The traing and test datasets will be loaded 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.

```
TrainUrl<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
TestUrl<-"http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(TrainUrl))
testing  <- read.csv(url(TestUrl))
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
dim(TrainSet)
```

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

```
dim(TestSet)
```

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

```
### remove variables with zero variance
ZeroVar <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -ZeroVar]
TestSet  <- TestSet[, -ZeroVar]
dim(TrainSet)
```

```
## [1] 13737   109
```

```
dim(TestSet)
```

```
## [1] 5885  109
```

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

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

```
dim(TestSet)
```

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

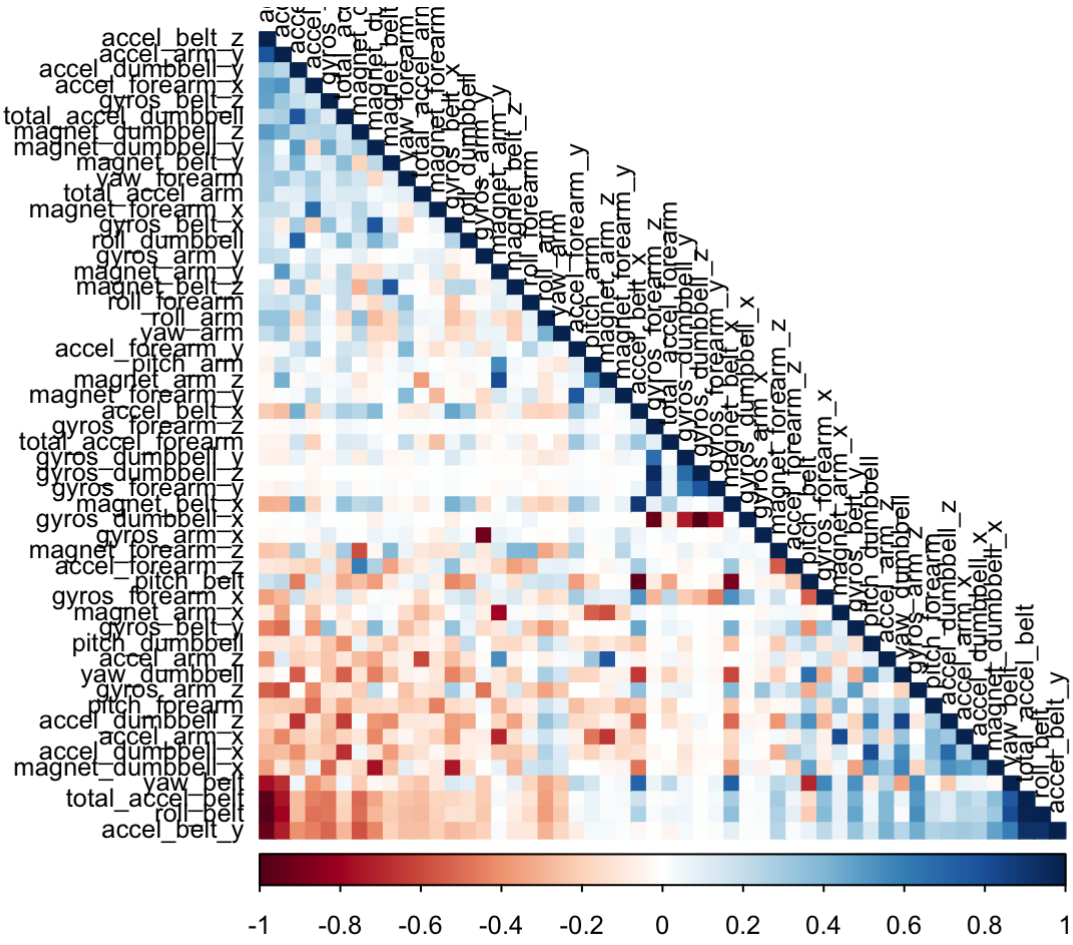
```
### Remove the non-predictors from the datasets, such as the index, subject name, time a
nd widow variables.
TrainSet<-TrainSet[,-c(1:6)]
TestSet<-TestSet[,-c(1:6)]
dim(TrainSet)
```

```
## [1] 13737    53
```

```
### After the cleaning, the variables have been reduced to 53.
```

4. Correlation analysis

```
corMatrix <- cor(TrainSet[, -53])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```




```
### The highly correlated variables are shown in dark colors in the graph above.As the c
orrelations are quite few, PCA will not be applied for this analysis.
```

IV Build prediction model

Three methods (Random Forests, Decision Tree and Generalized Boosted Model) will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset) will be used for the quiz predictions. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models. 1) Random Forests

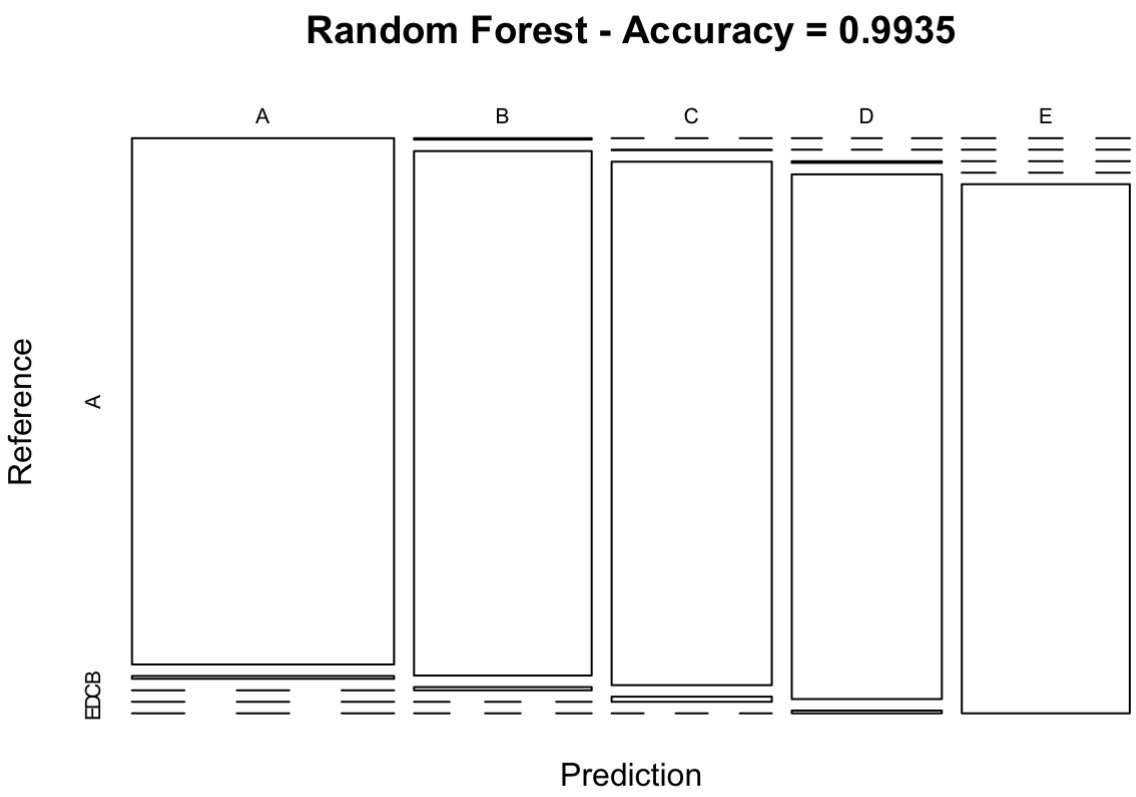
```
set.seed(10000)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                          trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    1    1    0    1 0.0007680492
## B   11 2636   11    0    0 0.0082768999
## C    0   20 2372    4    0 0.0100166945
## D    0    0   40 2211    1 0.0182060391
## E    0    0    0    6 2519 0.0023762376
```

```
### Prediction
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)
confMatRandForest
```

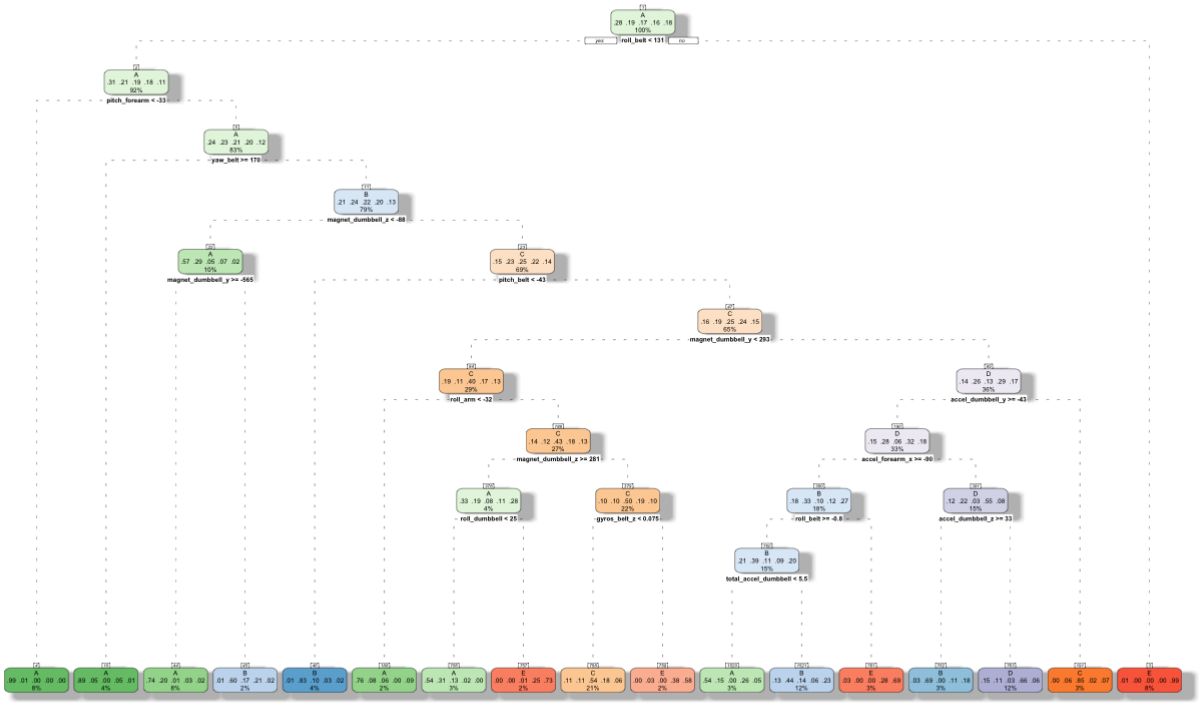
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    9    0    0    0
##          B    3 1129    7    0    0
##          C    0    1 1016   10    0
##          D    0    0    3  954    5
##          E    0    0    0    0 1077
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9912   0.9903   0.9896   0.9954
## Specificity            0.9979   0.9979   0.9977   0.9984   1.0000
## Pos Pred Value         0.9946   0.9912   0.9893   0.9917   1.0000
## Neg Pred Value         0.9993   0.9979   0.9979   0.9980   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1918   0.1726   0.1621   0.1830
## Detection Prevalence   0.2855   0.1935   0.1745   0.1635   0.1830
## Balanced Accuracy      0.9980   0.9946   0.9940   0.9940   0.9977
```

```
### plot matrix results
plot(confMatRandForest$table, col = confMatRandForest$byClass,
     main = paste("Random Forest - Accuracy =",
                  round(confMatRandForest$overall['Accuracy'], 4)))
```



2)Decision tree

```
set.seed(10000)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitDecTree)
```

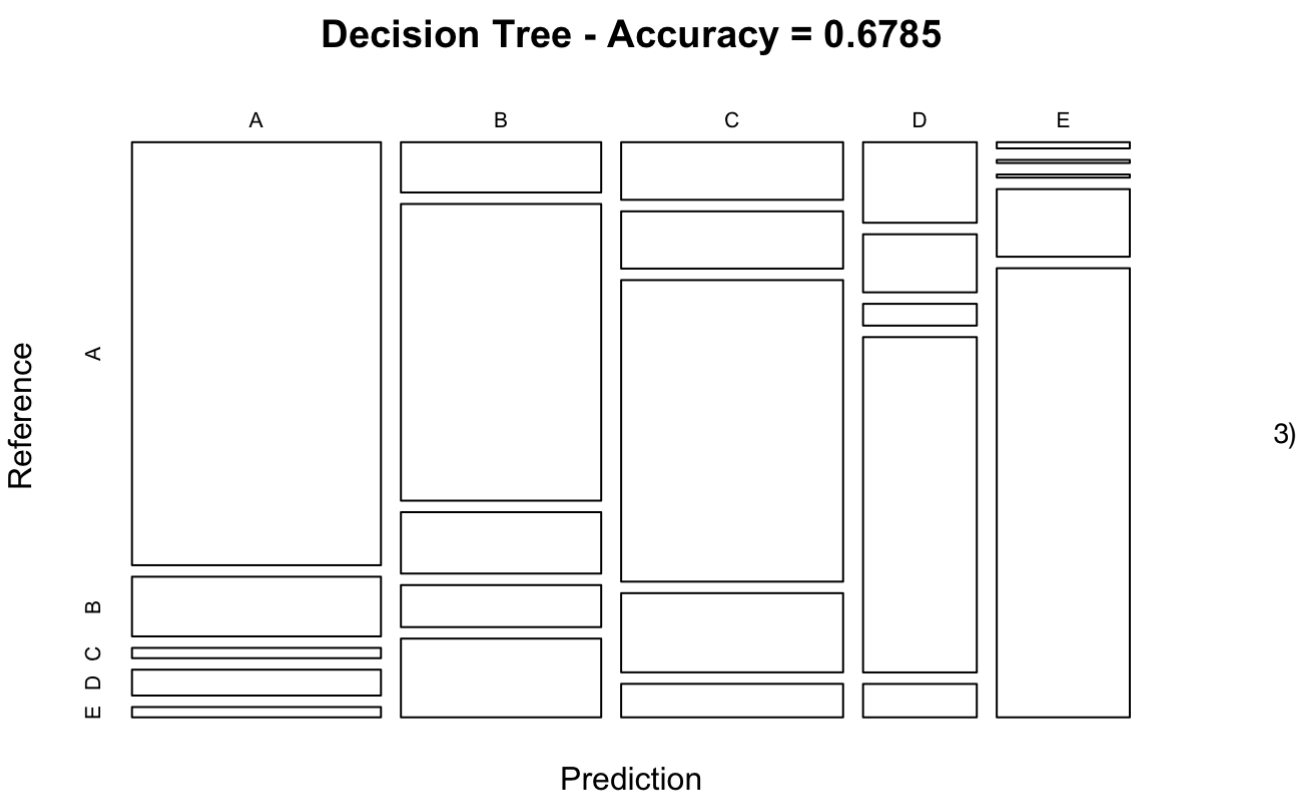


Rattle 2019-Jul-02 00:50:07 huiwuzhao

```
# prediction on Test dataset
predictDecTree <- predict(modFitDecTree, newdata=TestSet, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, TestSet$classe)
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1276  180   31   78   31
##          B  122  720  149  102  191
##          C  155  154  811  213   90
##          D  111   80   30  462   46
##          E   10    5    5  109  724
##
## Overall Statistics
##
##                Accuracy : 0.6785
##                  95% CI : (0.6664, 0.6904)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5938
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7622   0.6321   0.7904   0.4793   0.6691
## Specificity            0.9240   0.8812   0.8740   0.9457   0.9731
## Pos Pred Value         0.7995   0.5607   0.5699   0.6337   0.8488
## Neg Pred Value         0.9072   0.9089   0.9518   0.9026   0.9289
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2168   0.1223   0.1378   0.0785   0.1230
## Detection Prevalence   0.2712   0.2182   0.2418   0.1239   0.1449
## Balanced Accuracy      0.8431   0.7566   0.8322   0.7125   0.8211
```

```
# plot matrix results
plot(confMatDecTree$table, col = confMatDecTree$byClass,
     main = paste("Decision Tree - Accuracy =",
                  round(confMatDecTree$overall['Accuracy'], 4)))
```



3)

GBM model

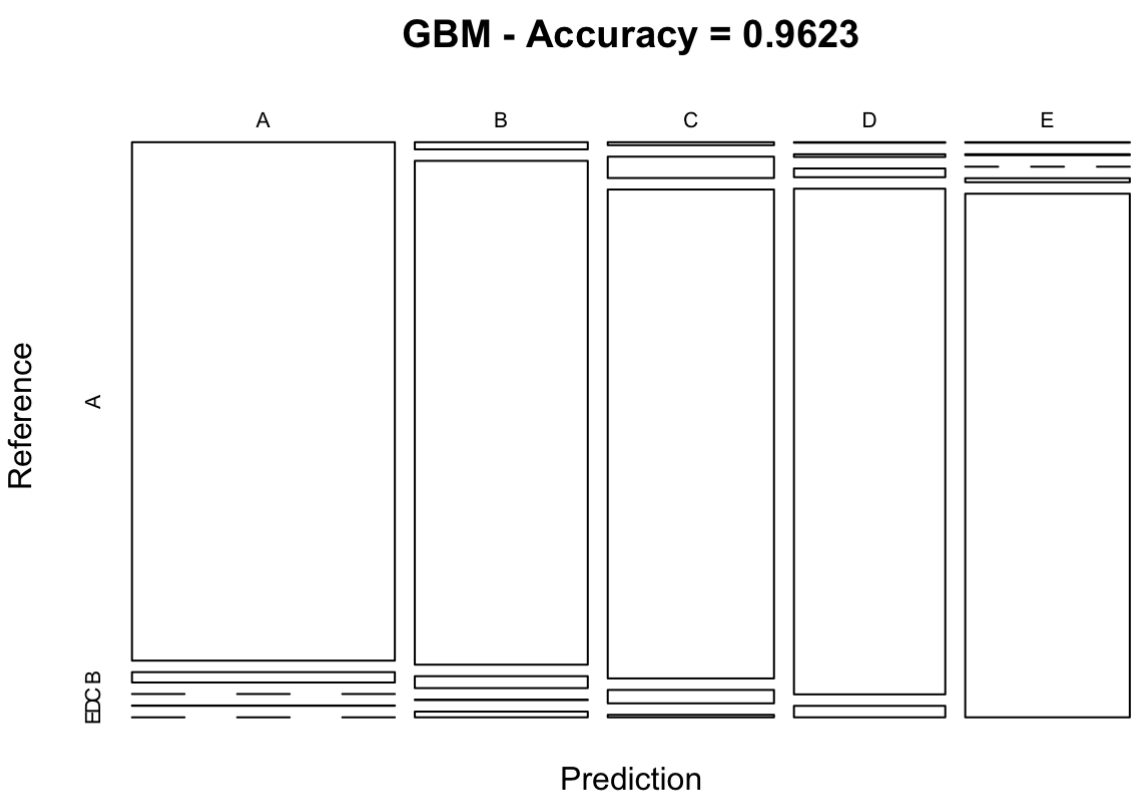
```
set.seed(10000)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=TrainSet, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)
modFitGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

```
# prediction on Test dataset
# prediction on Test dataset
predictGBM <- predict(modFitGBM, newdata=TestSet)
confMatGBM <- confusionMatrix(predictGBM, TestSet$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1651   33    0    1    0
##          B   15 1056   25    1   12
##          C    6   43  985   27    5
##          D    1    5   16  927   21
##          E    1    2    0    8 1044
##
## Overall Statistics
##
##                Accuracy : 0.9623
##                  95% CI : (0.9571, 0.967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9523
##
##  Mcnemar's Test P-Value : 7.605e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9863   0.9271   0.9600   0.9616   0.9649
## Specificity            0.9919   0.9888   0.9833   0.9913   0.9977
## Pos Pred Value         0.9798   0.9522   0.9240   0.9557   0.9896
## Neg Pred Value         0.9945   0.9826   0.9915   0.9925   0.9921
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2805   0.1794   0.1674   0.1575   0.1774
## Detection Prevalence   0.2863   0.1884   0.1811   0.1648   0.1793
## Balanced Accuracy      0.9891   0.9580   0.9717   0.9764   0.9813
```

```
# plot matrix results
plot(confMatGBM$table, col = confMatGBM$byClass,
     main = paste("GBM - Accuracy =", round(confMatGBM$overall['Accuracy'], 4)))
```



V Apply data to the test data.

RandomForest has the best accuracy among the three models. It will be applied to the test data.

```
predictTEST <- predict(modFitRandForest, newdata=testing)
predictTEST
```

```
##  [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
```