Practical Machine Learning

Arushi Agarwal

2/2/2018

Overview

This is the final report of the Peer Assessment project from Coursera's course Practical Machine Learning, as part of the Specialization in Data Science.

Aim : 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.

Background Information

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Read more: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Data Loading and Processing

First, we import the data by downloading the training and test data sets using the given URLs.During my exploratory data analysis, I saw that blank values, "NA", and "#DIV/0!" often show up in data columns so I have decided to treat all of these values as NA.

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Loading the required libraries :

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 5.0.14 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'default/Asia/
## Kolkata'
```

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.4.3
```

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

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.4.2

## corrplot 0.84 loaded
```

```r
set.seed(2345)
```

Data Loading and Cleaning

```r
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download data sets
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

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

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

Both created datasets have 160 variables. Removing the NA

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

```
## [1] 5885  129
```

```r
dim(TestSet)
```

```
## [1] 5885  129
```

```r
# 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
```

```r
dim(TestSet)
```

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

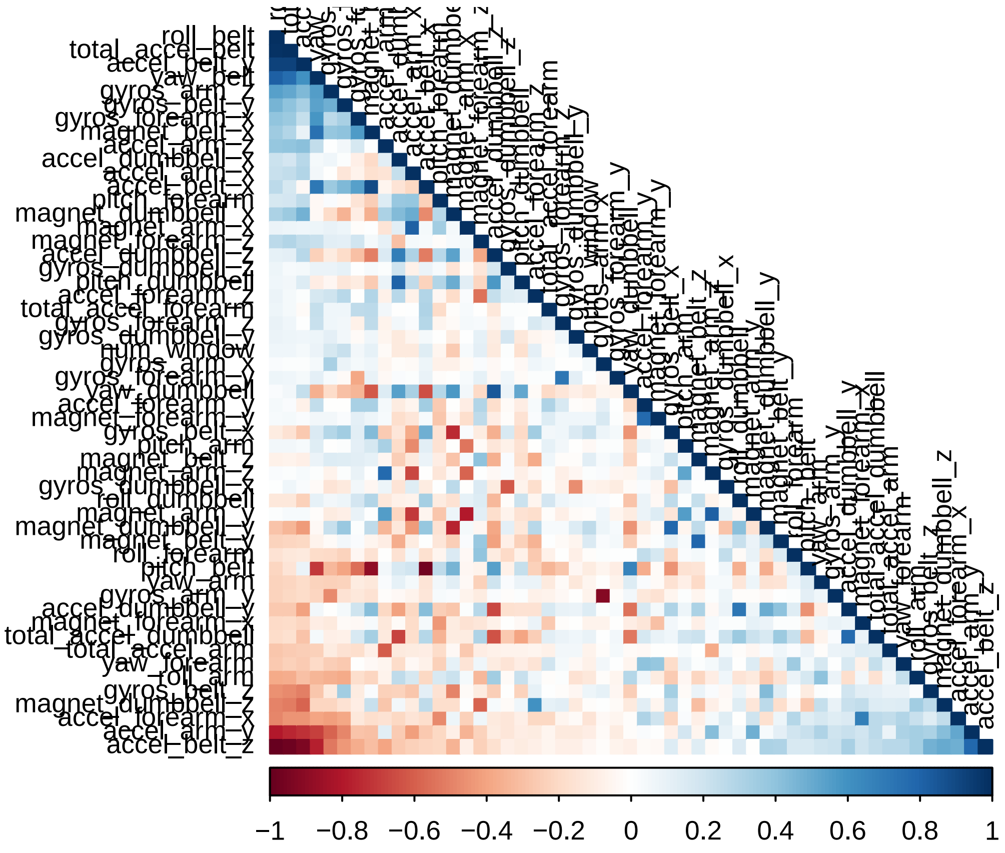
```r
# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

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

With the cleaning process above, the number of variables for the analysis has been reduced to 54 only.

Correlation Analysis

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



Note : The highly correlated variables are shown in dark colors in the graph above. To make an even more compact analysis, a PCA (Principal Components Analysis) could be performed as pre-processing step to the datasets. Nevertheless, as the correlations are quite few, this step will not be applied for this assignment.

Model Selection

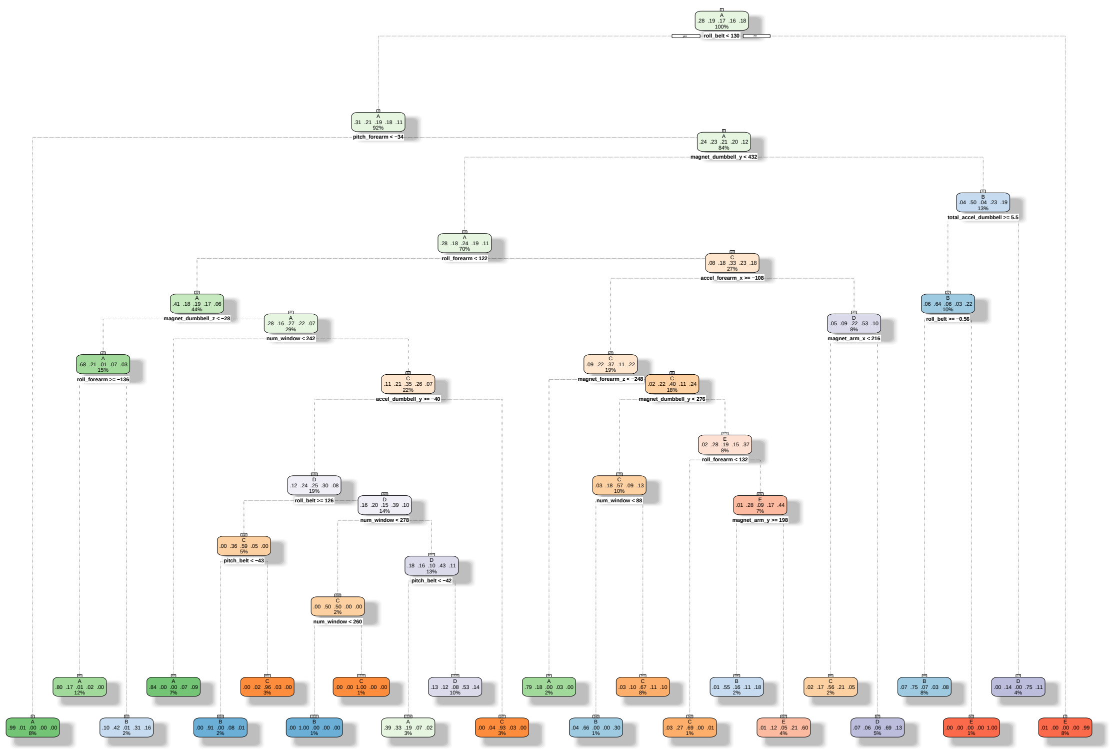
Three methods will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset) will be used for the quiz predictions.

The methods are: Random Forests, Decision Tree and Generalized Boosted Model, as described below. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

Decision Tree :

```
# model fit
set.seed(2345)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitDecTree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



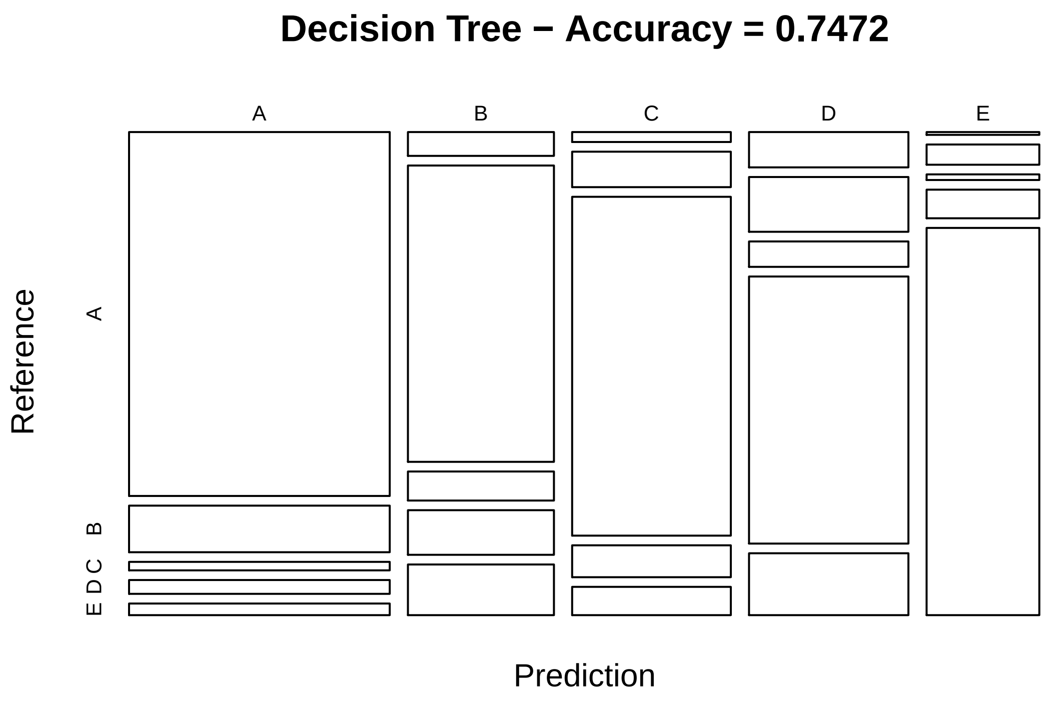
Rattle 2018−Feb−06 04:40:04 arushiagarwal

```r
# 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 1500  192   35   57   48
##          B   55  684   67  103  117
##          C   25   89  850   80   71
##          D   89  138   64  673  156
##          E    5   36   10   51  690
##
## Overall Statistics
##
##                Accuracy : 0.7472
##                  95% CI : (0.7358, 0.7582)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6794
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8961   0.6005   0.8285   0.6981   0.6377
## Specificity            0.9212   0.9279   0.9455   0.9092   0.9788
## Pos Pred Value         0.8188   0.6667   0.7623   0.6009   0.8712
## Neg Pred Value         0.9571   0.9064   0.9631   0.9389   0.9230
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2549   0.1162   0.1444   0.1144   0.1172
## Detection Prevalence   0.3113   0.1743   0.1895   0.1903   0.1346
## Balanced Accuracy      0.9086   0.7642   0.8870   0.8036   0.8082
```

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



Random Forest :

```
# model fit
set.seed(2345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                          trControl=controlRF)
modFitRandForest$finalModel
```

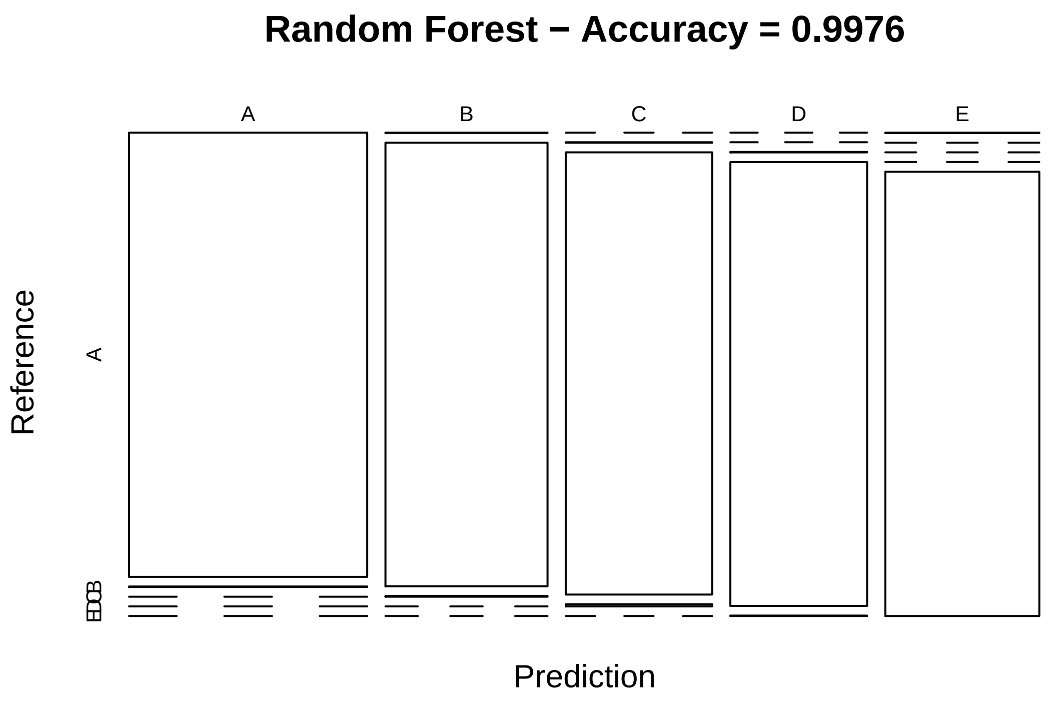
```
##
## Call:
```

```
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.22%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    6 2649    2    1    0 0.003386005
## C    0    4 2390    2    0 0.002504174
## D    0    0    9 2242    1 0.004440497
## E    0    1    0    4 2520 0.001980198
```

```
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    1 1136    2    0    0
##          C    0    1 1023    5    0
##          D    0    0    1  959    1
##          E    1    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9974   0.9971   0.9948   0.9991
## Specificity            0.9995   0.9994   0.9988   0.9996   0.9998
## Pos Pred Value         0.9988   0.9974   0.9942   0.9979   0.9991
## Neg Pred Value         0.9995   0.9994   0.9994   0.9990   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1930   0.1738   0.1630   0.1837
## Detection Prevalence   0.2845   0.1935   0.1749   0.1633   0.1839
## Balanced Accuracy      0.9992   0.9984   0.9979   0.9972   0.9994
```

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



GBM :

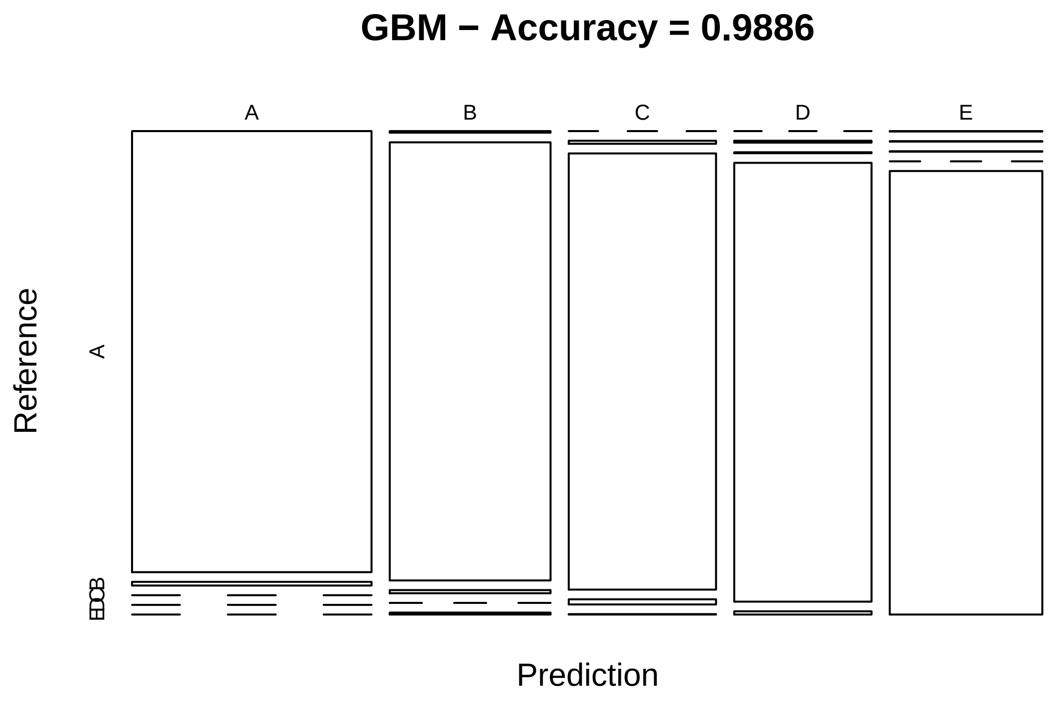
```
# model fit
set.seed(2345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=TrainSet, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)
modFitGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 40 had non-zero influence.
```

```
# 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 1669   14    0    0    0
##            B    4 1113    8    0    5
##            C    0    7 1015   12    1
##            D    0    4    2  952    7
##          E    1    1    1    0 1069
##
## Overall Statistics
##
##                Accuracy : 0.9886
##                  95% CI : (0.9856, 0.9912)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9856
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9772   0.9893   0.9876   0.9880
## Specificity            0.9967   0.9964   0.9959   0.9974   0.9994
## Pos Pred Value         0.9917   0.9850   0.9807   0.9865   0.9972
## Neg Pred Value         0.9988   0.9945   0.9977   0.9976   0.9973
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1891   0.1725   0.1618   0.1816
## Detection Prevalence   0.2860   0.1920   0.1759   0.1640   0.1822
## Balanced Accuracy      0.9968   0.9868   0.9926   0.9925   0.9937
```

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



Conclusions

The accuracy of the 3 regression modeling methods above are:

1. Decision Tree : 0.7472
2. Random Forest : 0.9976
3. GBM : 0.9886

In that case, the Random Forest model will be applied to predict the 20 quiz results.

```
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
```