Course Project - Practical Machine Learning

Rinosh Polavarapu

December 22, 2016

Executive Summary

To quantify the workout done by any person, the relevant heart rate data during the activity is required.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. But the quantified data needs to be analyzed and present in a more qualitative manner. The goal of the present project is 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).

Few important things to be addressed in this project are:

1. How the model that predicts the manner in which the individual exercise(**classe** variable) is built?
2. How the cross validation of predicted data with real data was performed?
3. What is the expected Out of Sample error?

1. Loading Data and Required librarites

```
library(caret)
library(randomForest)
library(gbm)
library(plyr)
```

The .csv files related to training and testing are obtained from the coursera site are saved in the working director and assigned to variables

```
### Data from .csv files and assign blanks, and observations divided 0 etc as NA
training_main <- read.csv("pml-training.csv", na.strings = c("NA","","#DIV/0!"))
testing_main <- read.csv("pml-testing.csv", na.strings = c("NA","","#DIV/0!"))

### Cleaning data by getting rid of variables with NA observations
training_main <- training_main[,colSums(is.na(training_main))==0 ]
testing_main <- testing_main[,colSums(is.na(testing_main))==0 ]

### Removing initial 6 columns(variables) which are just identification variables
training_main <- training_main[,-(1:7) ]
testing_main <- testing_main[,-(1:7) ]
```

2. Creating Partition using training data set furtherly into testing and training sets

```
inTrain <- createDataPartition(training_main$classe, p=0.7, list = FALSE)
training_sub <- training_main[inTrain,]
testing_sub <- training_main[-inTrain,]
```

The following columns are used in models bulit in section 3 to predict the **classe** variable.

```
print(names(training_main[,-53]))
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"            
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"        
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"        
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"       
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"           
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"         
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"         
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"        
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"       
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"    
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"    
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"   
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"         
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"     
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"     
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"    
## [52] "magnet_forearm_z"
```

3. Prediction Models

In order to predict the outcomes of classe variable in test data set, two different methods (1. Random Forests & 2. Boosting with Trees (**gbm**)) are employed.

3.1 Random Forest

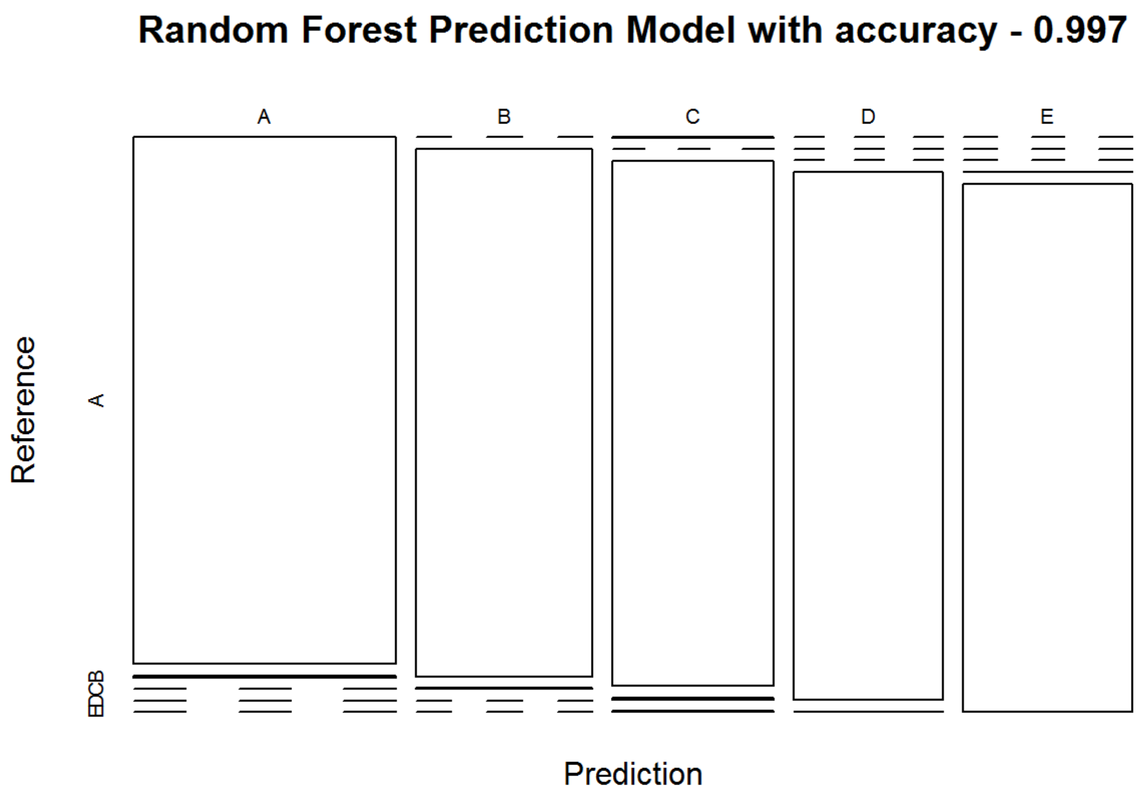
```
set.seed(333)
tr_control_RF <- trainControl(method = "cv",number = 5, verboseIter = FALSE, savePredictions =
"final")
modFit_RF <- train(classe ~ .,data = training_sub, method = "rf", trControl = tr_control_RF)
print(modFit_RF$finalModel)
```

```
## 
## 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: 0.74%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3898    4    3    0    1 0.002048131
## B   18 2623   14    3    0 0.013167795
## C    0   14 2375    6    1 0.008764608
## D    1    0   24 2225    2 0.011989343
## E    0    1    3    7 2514 0.004356436
```

```
pred_RF <- predict(modFit_RF, newdata = testing_sub)
conf_mat_RF <- confusionMatrix(pred_RF,testing_sub$classe)
print(conf_mat_RF, digits = 3)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    0    0
##          B    0 1133    2    0    0
##          C    1    0 1024    4    2
##          D    0    0    0  959    1
##          E    0    0    0    1 1079
## 
## Overall Statistics
##                                           
##                Accuracy : 0.997           
##                  95% CI : (0.995, 0.998)  
##     No Information Rate : 0.284           
##     P-Value [Acc > NIR] : <2e-16          
##                                           
##                   Kappa : 0.996           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.995    0.998    0.995    0.997
## Specificity             0.999    1.000    0.999    1.000    1.000
## Pos Pred Value          0.996    0.998    0.993    0.999    0.999
## Neg Pred Value          1.000    0.999    1.000    0.999    0.999
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.174    0.163    0.183
## Detection Prevalence    0.285    0.193    0.175    0.163    0.184
## Balanced Accuracy       0.999    0.997    0.998    0.997    0.999
```

```
plot(conf_mat_RF$table, col=conf_mat_RF$byClass, main = paste("Random Forest Prediction Model wi
th accuracy -", round(conf_mat_RF$overall['Accuracy'],3)))
```



3.2 Boosting with Trees (Generalized Boosting Model)

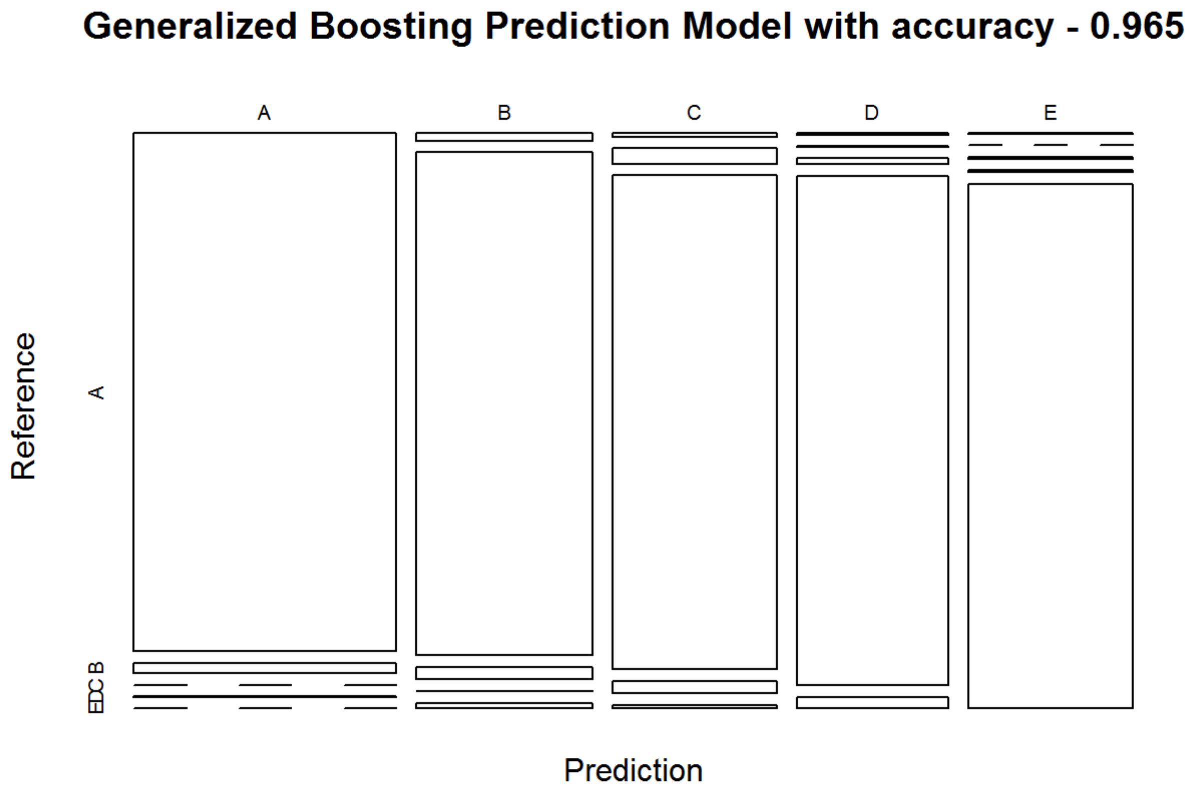
```
set.seed(333)
tr_control_GBM <- trainControl(method = "cv",number = 5, verboseIter = FALSE, savePredictions =
"final")
modFit_GBM <- train(classe ~ .,data = training_sub, method = "gbm", trControl = tr_control_GBM,
verbose = FALSE)
print(modFit_GBM$finalModel)
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 43 had non-zero influence.
```

```
pred_GBM <- predict(modFit_GBM, newdata = testing_sub)
conf_mat_GBM <- confusionMatrix(pred_GBM,testing_sub$classe)
print(conf_mat_GBM, digits = 3)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   33    0    2    0
##          B   16 1074   27    1   12
##          C    7   31  984   25    7
##          D    3    1   11  932   21
##          E    1    0    4    4 1042
## 
## Overall Statistics
##                                         
##                Accuracy : 0.965         
##                  95% CI : (0.96, 0.97)  
##     No Information Rate : 0.284         
##     P-Value [Acc > NIR] : < 2e-16       
##                                         
##                   Kappa : 0.956         
##  Mcnemar's Test P-Value : 3.03e-06      
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.984    0.943    0.959    0.967    0.963
## Specificity             0.992    0.988    0.986    0.993    0.998
## Pos Pred Value          0.979    0.950    0.934    0.963    0.991
## Neg Pred Value          0.994    0.986    0.991    0.993    0.992
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.280    0.182    0.167    0.158    0.177
## Detection Prevalence    0.286    0.192    0.179    0.164    0.179
## Balanced Accuracy       0.988    0.966    0.972    0.980    0.981
```

```
plot(conf_mat_GBM$table, col=conf_mat_GBM$byClass, main = paste("Generalized Boosting Prediction
 Model with accuracy -", round(conf_mat_GBM$overall['Accuracy'],3)))
```



4. Out of Sample error

Out of Sample error rate is the error rate on new data sets. In this case, it is nothing but 1-Accuracy of predicted outcomes.

Therefore, Out of Sample error for prediction models using -
a) Random Forest method: 1-0.992 = 0.008
b) Generalized Boosting method : 1 - 0.956 = 0.044

5. Conclusion with Main testing set

The classe variable for the testing set with 20 observations is to be predicted with the best possible model. It is obvious that the prediction model with Random Forest method is better of the two models as the accuracy is comparitively higher.

Hence, using model with Random Forest method, the predictions for 20 observations is shown below:

```
predict_finalTesting <- predict(modFit_RF ,newdata =testing_main )
print(predict_finalTesting)
```

```
##  [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
```