Predicting the Manner of Exercising

Wei Hao Khoong

25 May 2018

Abstract

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

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 report, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise.

Data

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Retrieving & Preparing Data

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.3
```

```r
training <- read.csv("pml-training.csv", na.strings=c("","NA", "#DIV/0!"))
testing <- read.csv("pml-testing.csv", na.strings=c("","NA", "#DIV/0!"))

# Remove columns that are not predictors, where the first 5 columns include non-predicting variables
trainingPs <- training[,-(1:5)]

# Remove predictors with data that does not vary
trainingPs <- trainingPs[,-nearZeroVar(trainingPs, saveMetrics = FALSE)]
```

```r
# Remove columns that have NA values
rem.columns <- names(which(colSums(is.na(trainingPs))>0))
trainingPs <- trainingPs[, !(names(trainingPs) %in% rem.columns)]
```

Cross Validation

We now split the cleaned data into a training and validation set:

```r
inTrain <- createDataPartition(y=trainingPs$classe, p=.7, list= FALSE)
trainingSet <- trainingPs[inTrain,]
validationSet <- trainingPs[-inTrain,]
```

Summary of training and validation datasets

```r
CrossValSummary <- rbind(Original_data = dim(trainingPs), training_subset = dim(trainingSet), validatio
colnames(CrossValSummary) <- c("Observations", "Predictors")
CrossValSummary
```

```
##                   Observations Predictors
## Original_data            19622         54
## training_subset          13737         54
## validation_subset         5885         54
```

Our Models

Next, we construct two models: random forest (RF) and generalized boosting model (GBM). Note that both are tree-based classification models from the caret package.

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.3.3

## Loading required package: survival
```

```
##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```r
set.seed(111)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFit <- train(classe~., data=trainingSet, method="rf", trControl=controlRF)
modFit$finalModel
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFit_gbm <- train(classe~., data=trainingSet, method="gbm", trControl = controlGBM, verbose = FALSE)
modFit_gbm$finalModel
```

The accuracies of our models are: 0.9976 for RF and 0.987 for GBM. In particular, the RF model offers an out-of sample error rate of 0.18%.

Model Accuracy

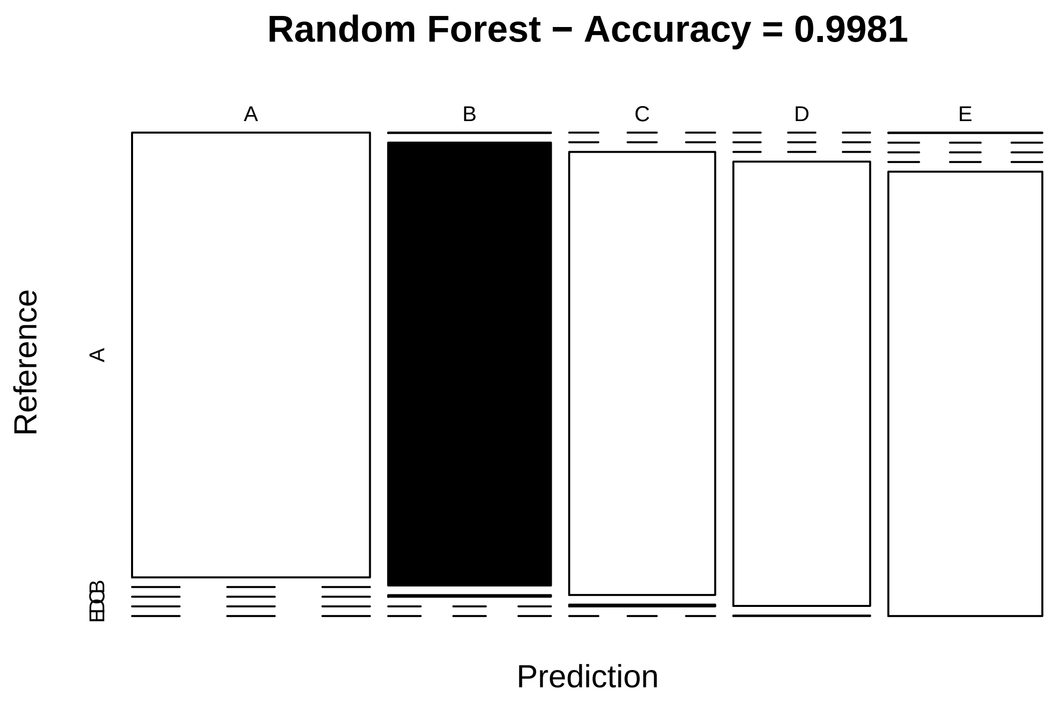
To evaluate the two models, we will use the validation data subset to predict the classification and compare the predicted classification with the true classification.

Random Forest Model Prediction Accuracy

```r
Predict_rf <- predict(modFit, validationSet)
CM_RF <- confusionMatrix(Predict_rf, validationSet$classe)
CM_RF$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9981308      0.9976358      0.9966580      0.9990666      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

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



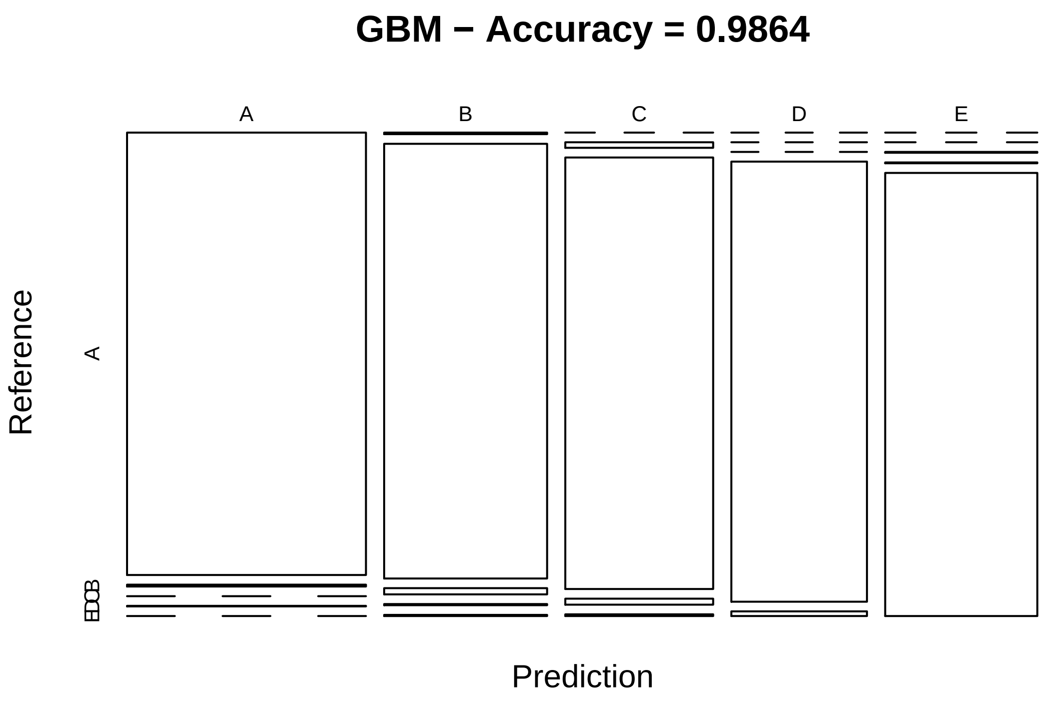
We note that the RF model has the best accuracy of the 2 models.

GBM Model Prediction Accuracy

```
Predict_gbm <- predict(modFit_gbm, validationSet)
CM_GBM <- confusionMatrix(Predict_gbm, validationSet$classe)
CM_GBM$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9864061      0.9828031      0.9831094      0.9892066      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

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



The GBM model is not as accurate as the RF model, though sufficiently high.

We compare the predictions across models in the following table:

```
CM_RF$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    0    0    0    0
##          B    1 1139    4    0    0
##          C    0    0 1022    4    0
##          D    0    0    0  960    1
##          E    1    0    0    0 1081
```

```
CM_GBM$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    7    0    2    0
##          B    4 1119   16    3    3
##          C    0   13 1008   14    4
##          D    0    0    0  943   10
##          E    0    0    2    2 1065
```

From this, we find that the RF method was better across all classifications.

Testing Dataset Result Prediction

Lastly, we proceed with predicting using the RF method:

```
modelPredictions <- predict(modFit, testing)
cbind(testing[,1:2], classe = modelPredictions)
```

```
##     X user_name classe
## 1   1     pedro      B
## 2   2    jeremy      A
## 3   3    jeremy      B
## 4   4    adelmo      A
## 5   5    eurico      A
## 6   6    jeremy      E
## 7   7    jeremy      D
## 8   8    jeremy      B
## 9   9  carlitos      A
## 10 10   charles      A
## 11 11  carlitos      B
## 12 12    jeremy      C
## 13 13    eurico      B
## 14 14    jeremy      A
## 15 15    jeremy      E
## 16 16    eurico      E
## 17 17     pedro      A
## 18 18  carlitos      B
## 19 19     pedro      B
## 20 20    eurico      B
```

```
modelPredictions
```

```
##  [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
```