HAR Weight Lifting Exercise

Tomas Rodriguez

January 27, 2019

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).

Introduction/Scope

The goal of you this analysis is to predict the manner in which participants did the exercise. This is the “classe” variable in the training set. Here we describe how we built your model, how we used cross validation, what we think the expected out of sample error is, and why we made the choices we did. We will also use our prediction model to predict 20 different test cases.

Experiment Description

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). Sensors were placed at four locations: belt, arm, dumbbell, and forearm. The output of the sensor data become the input files to our analysis.

Data URL

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

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

Analysis Repository

URL: https://github.com/tomyr95/har (https://github.com/tomyr95/har)

Environment Setup

```
library(caret); library(dplyr); library(Hmisc); library(car); library(corrplot)
library(rattle); library(rpart); library(parallel); library(doParallel)
library(e1071); library(ggplot2); library(klaR); library(stringr)
library(lattice); library(beepr); library(grid); library(gridExtra)
```

Download/Read Data

1. We downloaded the training/testing data into our working directory.
2. We assined it to objects dat1 and dat2, respectively.

```
url1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url1, "training.csv")
download.file(url2, "testing.csv")

dat1 <- read.csv("training.csv")
dat2 <- read.csv("testing.csv")
```

Inspect/Cleanup/Partition Data

1. In reviewing the data we find many zero, nezr-zero, and columns containing only NA.
2. In addition we see a number of columns that are not pertinent to our analysis.
3. We removed such columns for both the trainig (dat1) and testing (dat2) data.
4. We split original trainig data into training(TRIAN)/validation(VAL) sets.
5. We assign testing data and remove dat, dat2 placeholders.

```
str(dat1)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2
...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323
084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484
323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9
...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4
...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1
...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
```

```
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
dat1 <- dat1[,-c(1:7)] #Remove 1st columns with identifying data, not for processing
dat1 <- dat1[, -nearZeroVar(dat1)] #Remove all zero and near-zero values
dat1 <- dat1[ , colSums(is.na(dat1)) == 0] #Remove all only NA columns
dat1$id <- c(1:nrow(dat1)) #Added simple 'id' for reference purposes.

#Apply same transformations to dat2(testing data)
dat2 <- dat2[,-c(1:7)]
dat2 <- dat2[, -nearZeroVar(dat2)]
dat2 <- dat2[ , colSums(is.na(dat2)) == 0]

#Divide training (dat1) into training(TRAIN)/validation(VAL) datasets
set.seed(333)
inTrain <- createDataPartition(y = dat1$classe, p=0.75)[[1]]
TRAIN <- dat1[inTrain,]
VAL <- dat1[-inTrain,]

#Verify outcome (classe) proportionally distributed (expected)
table(TRAIN$classe)
```

```
## 
##    A    B    C    D    E 
## 4185 2848 2567 2412 2706
```

```
table(VAL$classe)
```

```
## 
##    A    B    C    D    E 
## 1395  949  855  804  901
```

```
testing <-dat2

remove(dat1)
remove(dat2)
```

Review Data

1. We note columns called ‘total_accel’ and suspect they are functions of other predictors.
2. We will use ‘featureplot’ function to review for potential outliers in the data.
3. We will leave review of potential important predictors to the selected model.
4. One can get a sense of important features “pairs” and“density plot types.
5. Saved feature plots in: har-featureset1.png (2, 3, 4).
6. Removed four (4) points as suspected outliers.

```
names(TRAIN)
```

```
##  [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"     "classe"               "id"
```

```
plot1 <- featurePlot(x=TRAIN[, 1:3], y=TRAIN$classe, plot="box")   # BELT rpy       (r p y)
plot2 <- featurePlot(x=TRAIN[, 5:7], y=TRAIN$classe, plot="box")   # BELT gyrow     (LOW)
plot3 <- featurePlot(x=TRAIN[, 8:10], y=TRAIN$classe, plot="box")  # BELT accel     (x y z)
plot4 <- featurePlot(x=TRAIN[, 11:13], y=TRAIN$classe, plot="box") # BELT magnet    (x)
png('har-featureset1.png', width = 10, height = 7.5, units = 'in', res = 300)
grid.arrange(plot1, plot2, plot3, plot4, ncol=2, nrow=2)
dev.off()
```

```
## png
##   2
```

```
plot1 <- featurePlot(x=TRAIN[, 14:16], y=TRAIN$classe, plot="box", no.axes=TRUE) # ARM rpy
   (r)
plot2 <- featurePlot(x=TRAIN[, 18:20], y=TRAIN$classe, plot="box", no.axes=TRUE) # ARM gyros
   (LOW)
plot3 <- featurePlot(x=TRAIN[, 21:23], y=TRAIN$classe, plot="box") # ARM accel      (z)
plot4 <- featurePlot(x=TRAIN[, 24:26], y=TRAIN$classe, plot="box") # ARM magnet     (LOW)
png('har-featureset2.png', width = 10, height = 7.5, units = 'in', res = 300)
grid.arrange(plot1, plot2, plot3, plot4, ncol=2, nrow=2)
dev.off()
```

```
## png
##   2
```

```
plot1 <- featurePlot(x=TRAIN[, 27:29], y=TRAIN$classe, plot="box") # DUMBBELL rpy    (LOW)
plot2 <- featurePlot(x=TRAIN[, 31:33], y=TRAIN$classe, plot="box") # DUMBBELL gyros  (LOW)
plot3 <- featurePlot(x=TRAIN[, 34:36], y=TRAIN$classe, plot="box") # DUMBBELL accel  (y)
plot4 <- featurePlot(x=TRAIN[, 37:39], y=TRAIN$classe, plot="box") # DUMBELL magnet  (y z)
png('har-featureset3.png', width = 10, height = 7.5, units = 'in', res = 300)
grid.arrange(plot1, plot2, plot3, plot4, ncol=2, nrow=2)
dev.off()
```

```
## png
##   2
```

```
plot1 <- featurePlot(x=TRAIN[, 40:42], y=TRAIN$classe, plot="box") # FOREARM rpy     (LOW)
plot2 <- featurePlot(x=TRAIN[, 44:46], y=TRAIN$classe, plot="box") # FOREARM gyros   (LOW)
plot3 <- featurePlot(x=TRAIN[, 47:49], y=TRAIN$classe, plot="box") # FOREARM Accel   (z)
plot4 <- featurePlot(x=TRAIN[, c(4, 17, 30, 43)], y=TRAIN$classe, plot="box")           # f(pred
ictors)
png('har-featureset4.png', width = 10, height = 7.5, units = 'in', res = 300)
grid.arrange(plot1, plot2, plot3, plot4, ncol=2, nrow=2)
dev.off()
```

```
## png
##   2
```

Remove Outliers (Based on extreme observations ONLY in this rather large data set)

```
TRAIN <- TRAIN[TRAIN$id!=5373,]
TRAIN <- TRAIN[TRAIN$id!=9274,]
TRAIN <- TRAIN[TRAIN$id!=152,]
TRAIN <- TRAIN[TRAIN$id!=9941,]

#Remove id column for rest of analysis
TRAIN <- TRAIN[, -ncol(TRAIN)]
VAL <- VAL[, -ncol(VAL)]
```

Sample Smaller Dataset for Model Spot Check

```
  set.seed(333)
  sampleTRAIN <- createDataPartition(y = TRAIN$classe, p=0.05)[[1]]
  sampleVAL <- createDataPartition(y = VAL$classe, p=0.05)[[1]]
  smlTr <- TRAIN[sampleTRAIN,]
  smlVal <- VAL[sampleVAL,]
  table(smlTr$classe)
```

```
##
##   A   B   C   D   E
## 210 143 129 121 136
```

```
table(smlVal$classe)
```

```
##
##  A  B  C  D  E
## 70 48 43 41 46
```

Model Spot Check

1- Model Spot Check/Selection performed for a selection of models discussed in class 2- Class: Data Science/Practical Machine Learning, John's Hopkins University, Bis-Statistics. 3- Random Forrest was found to be the most accurate model to fit the training data set. 4- Top performing models highly correlated, so don't consider stacking.

```
  control <- trainControl(method="repeatedcv", number=5, repeats=3)

  # Linear Discriminant Analysis
  set.seed(333)
  start <- Sys.time()
  fit.lda <- train(classe~., data=smlTr, method="lda", metric="Accuracy", preProc=c("center", "s
cale"), trControl=control)
  end <- Sys.time()
  prediction <-predict(fit.lda, smlTr[, -ncol(smlTr)])
  time.lda <- end - start
  p.lda<-prediction

  # CART
  set.seed(333)
  start <- Sys.time()
  fit.rpart <- train(classe~., data=smlTr, method="rpart", metric="Accuracy", trControl=control)
  end <- Sys.time()
  prediction <-predict(fit.lda, smlTr[, -ncol(smlTr)])
  time.rpart <- end - start
  p.rpart<-prediction

  # Bagging
  set.seed(333)
  start <- Sys.time()
  fit.treebag <- train(classe~., data=smlTr, method="treebag", metric="Accuracy", trControl=cont
rol, verbose=FALSE)
  end <- Sys.time()
  prediction <-predict(fit.treebag, smlTr[, -ncol(smlTr)])
  time.treebag <- end - start
  p.treebag<-prediction

  # Random Forest
  set.seed(333)
  start <- Sys.time()
  fit.rf <- train(classe~., data=smlTr, method="rf", metric="Accuracy", trControl=control)
  end <- Sys.time()
  prediction <-predict(fit.rf, smlTr[, -ncol(smlTr)])
  time.rf <- end - start
  p.rf<-prediction

  # Boosting
  set.seed(333)
  start <- Sys.time()
  fit.gbm <- train(classe~., data=smlTr, method="gbm", metric="Accuracy", trControl=control, ver
bose=FALSE)
  end <- Sys.time()
  prediction <-predict(fit.gbm, smlTr[, -ncol(smlTr)])
  time.gbm <- end - start
  p.gbm<-prediction

  # Summarize Results
  results <- resamples(list(lda=fit.lda, rpart=fit.rpart, treebag=fit.treebag, rf=fit.rf, gbm=fi
t.gbm))
```

```
sum_df <- summary(results)
times <- data.frame("names"=c("lda", "rpart", "treebag", "rf", "gbm"),
          "times"=c(time.lda, time.rpart, time.treebag, time.rf, time.gbm))
print(sum_df)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, rpart, treebag, rf, gbm
## Number of resamples: 15
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda     0.6510067 0.6790541 0.7006803 0.6982023 0.7084712 0.7567568    0
## rpart   0.3493151 0.4142481 0.5135135 0.4812041 0.5457575 0.5918367    0
## treebag 0.8053691 0.8282922 0.8424658 0.8502487 0.8682432 0.9127517    0
## rf      0.8287671 0.8644052 0.8657718 0.8713961 0.8775454 0.9121622    0
## gbm     0.8219178 0.8469388 0.8724832 0.8668945 0.8851351 0.8993289    0
##
## Kappa
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lda     0.55979774 0.5952985 0.6215331 0.6183161 0.6316898 0.6919163    0
## rpart   0.09458842 0.2236695 0.3707334 0.3211274 0.4192747 0.4757489    0
## treebag 0.75264755 0.7822194 0.8001904 0.8103907 0.8335176 0.8898430    0
## rf      0.78389580 0.8279657 0.8304313 0.8371203 0.8452043 0.8886896    0
## gbm     0.77554399 0.8060610 0.8394670 0.8316022 0.8545664 0.8726786    0
```

```
print(times)
```

```
##     names           times
## 1     lda  0.7701921 secs
## 2   rpart  0.8768082 secs
## 3 treebag  6.9227722 secs
## 4      rf 42.2010620 secs
## 5     gbm 27.9692690 secs
```

```
#Access correlation between Top performing models
confusionMatrix(p.rf, p.gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 210   0   0   0   0
##          B   0 143   0   0   0
##          C   0   0 129   0   0
##          D   0   0   0 121   0
##          E   0   0   0   0 136
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.995, 1)
##     No Information Rate : 0.2842
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000    1.000
## Specificity            1.0000   1.0000   1.0000   1.0000    1.000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Prevalence             0.2842   0.1935   0.1746   0.1637    0.184
## Detection Rate         0.2842   0.1935   0.1746   0.1637    0.184
## Detection Prevalence   0.2842   0.1935   0.1746   0.1637    0.184
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000    1.000
```

```
confusionMatrix(p.rf, p.treebag)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   A   B   C   D   E
##          A 210   0   0   0   0
##          B   0 143   0   0   0
##          C   0   0 129   0   0
##          D   0   0   0 121   0
##          E   0   0   0   0 136
## 
## Overall Statistics
##                                      
##                Accuracy : 1          
##                  95% CI : (0.995, 1)
##     No Information Rate : 0.2842     
##     P-Value [Acc > NIR] : < 2.2e-16  
##                                      
##                   Kappa : 1          
##  Mcnemar's Test P-Value : NA         
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000    1.000
## Specificity            1.0000   1.0000   1.0000   1.0000    1.000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Prevalence             0.2842   0.1935   0.1746   0.1637    0.184
## Detection Rate         0.2842   0.1935   0.1746   0.1637    0.184
## Detection Prevalence   0.2842   0.1935   0.1746   0.1637    0.184
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000    1.000
```

Run Ramdom Forrest, All Predictors

1- Run RF model with all predictors and full TRAIN data to assess predictor importance (varImp).

```
  set.seed(333)
  training <- TRAIN
  validation <- VAL
  fitControl <- trainControl(method="repeatedcv", number=5, repeats=3)
  start <- Sys.time()
  fit <- train(classe ~., method = "rf", data = training, trControl = fitControl, importance=TRU
E)
  end <- Sys.time()
  timeRF <- end - start
  pTr <-predict(fit, training[, -ncol(training)])
  pVal <-predict(fit, validation[, -ncol(validation)])
  accTr <-confusionMatrix(pTr, training[, ncol(training)])$overall['Accuracy']
  accVal <-confusionMatrix(pVal, validation[, ncol(validation)])$overall['Accuracy']
  fitRF <- fit
  Imp <- varImp(fitRF)$importance
  Imp$names <- rownames(Imp)
  features <- Imp %>% mutate(id = row_number()) %>% mutate(sum = A+B+C+D+E) %>% arrange(desc(su
m))
  cols_all <- features$id
  print(features)
```

```
##             A        B        C         D        E                names id
## 1  92.142513 91.79959 73.67040 100.00000 49.68146             yaw_belt  3
## 2  64.714729 91.16680 93.54129  91.39563 59.73162            roll_belt  1
## 3  64.893533 96.28356 77.44490  70.01059 63.82285           pitch_belt  2
## 4  71.154332 68.37430 90.74185  69.87000 56.76221    magnet_dumbbell_z 39
## 5  47.460311 78.58983 68.21261  58.40571 47.96653             roll_arm 14
## 6  45.664579 70.00607 59.00961  64.56742 54.39833     accel_dumbbell_y 35
## 7  44.158528 72.13908 60.32763  54.33492 50.97969         gyros_belt_z  7
## 8  49.470478 55.03912 79.08787  54.25863 41.64644    magnet_dumbbell_y 38
## 9  38.911396 59.82930 51.81152  61.22643 52.05880        pitch_forearm 41
## 10 42.321806 60.37897 56.49821  54.29630 35.74546         accel_belt_z 10
## 11 39.119267 58.42879 59.41469  49.42798 41.97304     gyros_dumbbell_y 32
## 12 45.010735 53.48937 51.50906  47.58077 47.77110     accel_dumbbell_z 36
## 13 43.616975 50.96397 48.48822  52.26668 38.59675        magnet_belt_y 12
## 14 25.564560 55.44133 53.10733  39.03143 60.44291        magnet_belt_x 11
## 15 28.402784 59.40465 51.15313  55.52887 34.46269              yaw_arm 16
## 16 35.167375 55.05881 48.30806  49.97035 36.00389     magnet_forearm_z 52
## 17 29.442608 65.87876 44.17627  43.11397 38.70674          gyros_arm_y 19
## 18 28.782293 54.13018 52.74973  39.81120 40.15225     accel_dumbbell_x 34
## 19 43.647937 40.01139 40.97414  42.93783 43.35103 total_accel_dumbbell 30
## 20 36.556664 43.27525 41.45298  48.20416 38.32418        magnet_belt_z 13
## 21 25.821633 44.37850 55.09205  40.62182 38.90226      accel_forearm_z 49
## 22 28.444237 58.16420 41.68401  40.28820 35.54178     gyros_dumbbell_z 33
## 23 34.862962 38.38450 52.00992  43.51937 33.68536        roll_dumbbell 27
## 24 30.162259 45.88555 50.16669  40.81346 32.96438     magnet_forearm_y 51
## 25 27.692753 39.57091 49.37865  35.15981 43.91669          yaw_forearm 42
## 26 27.086587 49.67516 38.40367  38.58680 34.44458          accel_arm_y 22
## 27 23.501910 42.53011 44.16420  47.92486 30.01851      accel_forearm_x 47
## 28 31.844452 42.12551 54.35134  32.01845 23.64720    magnet_dumbbell_x 37
## 29 34.247796 33.65955 50.17245  36.94938 27.75463         roll_forearm 40
## 30 33.017185 45.98480 42.63863  32.68894 28.09933         magnet_arm_z 26
## 31 25.360768 51.55269 46.73639  35.75854 21.73939     gyros_dumbbell_x 31
## 32 20.801886 38.85686 43.80809  38.67721 38.65228         yaw_dumbbell 29
## 33 31.094258 40.19395 45.91036  32.77973 29.85501      accel_forearm_y 48
## 34 32.275425 36.93979 35.90322  34.34344 27.90494  total_accel_forearm 43
## 35 17.396454 56.30104 31.02552  30.34131 27.60987      gyros_forearm_z 46
## 36 26.626184 43.41368 34.95545  24.07174 32.23642         gyros_belt_x  5
## 37 25.511547 36.71736 36.00076  29.85547 30.58137      gyros_forearm_x 44
## 38 21.195198 44.54628 35.04581  22.96525 30.70178         accel_belt_x  8
## 39 21.065464 39.06837 29.17010  22.50470 32.72251     total_accel_belt  4
## 40 21.580364 29.18373 27.28209  42.69338 17.75067          accel_arm_x 21
## 41 14.764947 40.42534 29.93413  29.37335 22.70572      gyros_forearm_y 45
## 42 13.658476 28.65514 36.34744  32.57685 24.29840          accel_arm_z 23
## 43 12.408951 25.29227 34.39927  26.25510 26.74736     magnet_forearm_x 50
## 44 13.849138 33.36053 19.41539  25.75130 25.32229         accel_belt_y  9
## 45 16.404947 34.17671 19.78303  24.79716 20.99247          gyros_arm_x 18
## 46 11.008801 22.97363 29.00141  31.11851 17.32895         magnet_arm_y 25
## 47  6.538288 33.38958 32.45140  24.13394 12.40788            pitch_arm 15
## 48  6.278925 34.66664 30.19699  18.50348 14.89759       pitch_dumbbell 28
## 49  7.994172 22.79642 21.48650  22.39762 28.80971         gyros_belt_y  6
## 50 18.884825 18.29863 28.10860  21.37809 11.34542         magnet_arm_x 24
## 51 19.430798 26.37408 21.04459  13.21313 15.83780          gyros_arm_z 20
## 52  0.000000 17.32774 18.02980  21.33332 12.73004      total_accel_arm 17
```

```
##          sum
## 1  407.29396
## 2  400.55008
## 3  372.45544
## 4  356.90270
## 5  300.63499
## 6  293.64600
## 7  281.93985
## 8  279.50254
## 9  263.83745
## 10 249.24074
## 11 248.36376
## 12 245.36104
## 13 233.93259
## 14 233.58756
## 15 228.95212
## 16 224.50848
## 17 221.31835
## 18 215.62566
## 19 210.92232
## 20 207.81324
## 21 204.81625
## 22 204.12242
## 23 202.46211
## 24 199.99234
## 25 195.71881
## 26 188.19680
## 27 188.13960
## 28 183.98695
## 29 182.78380
## 30 182.42889
## 31 181.14778
## 32 180.79634
## 33 179.83330
## 34 167.36682
## 35 162.67419
## 36 161.30348
## 37 158.66650
## 38 154.45433
## 39 144.53114
## 40 138.49023
## 41 137.20349
## 42 135.53631
## 43 125.10294
## 44 117.69865
## 45 116.15431
## 46 111.43130
## 47 108.92109
## 48 104.54362
## 49 103.48443
## 50  98.01556
## 51  95.90040
## 52  69.42090
```

```
  df <- data.frame(c("features", "timeRF", "accTr", "accVal"), c(ncol(training)-1, timeRF, accT
r, accVal))
  df
```

```
##   c..features....timeRF....accTr....accVal..
## 1                                  features
## 2                                    timeRF
## 3                                     accTr
## 4                                    accVal
##   c.ncol.training....1..timeRF..accTr..accVal.
## 1                                    52.000000
## 2                                    45.208954
## 3                                     1.000000
## 4                                     0.995106
```

```
  cols_all
```

```
##  [1]  3  1  2 39 14 35  7 38 41 10 32 36 12 11 16 52 19 34 30 13 49 33 27
## [24] 51 42 22 47 37 40 26 31 29 48 43 46  5 44  8  4 21 45 23 50  9 18 25
## [47] 15 28  6 24 20 17
```

Create Dataframe for Fit Results by Predictor Quantity

```
  pQty <- data.frame(matrix(ncol=3, nrow=0))
  colnames(pQty) <- c("Pred", "Time", "OOS")
```

Optimize Predictor Selection

1- To prevent underfitting or overfitting the model we will use out-of-sample (OOS) error estimation. 2- The goal is to find the number of predictors that minimize the OOS error rate. 3- We wil use the validation (VAL) dataset predictions to assess OOS error. 4- This was done manually and stored in object “Multi_Parameter.rds”. 5- OOS error (1-accuracy) will not reduce predictor quantity.

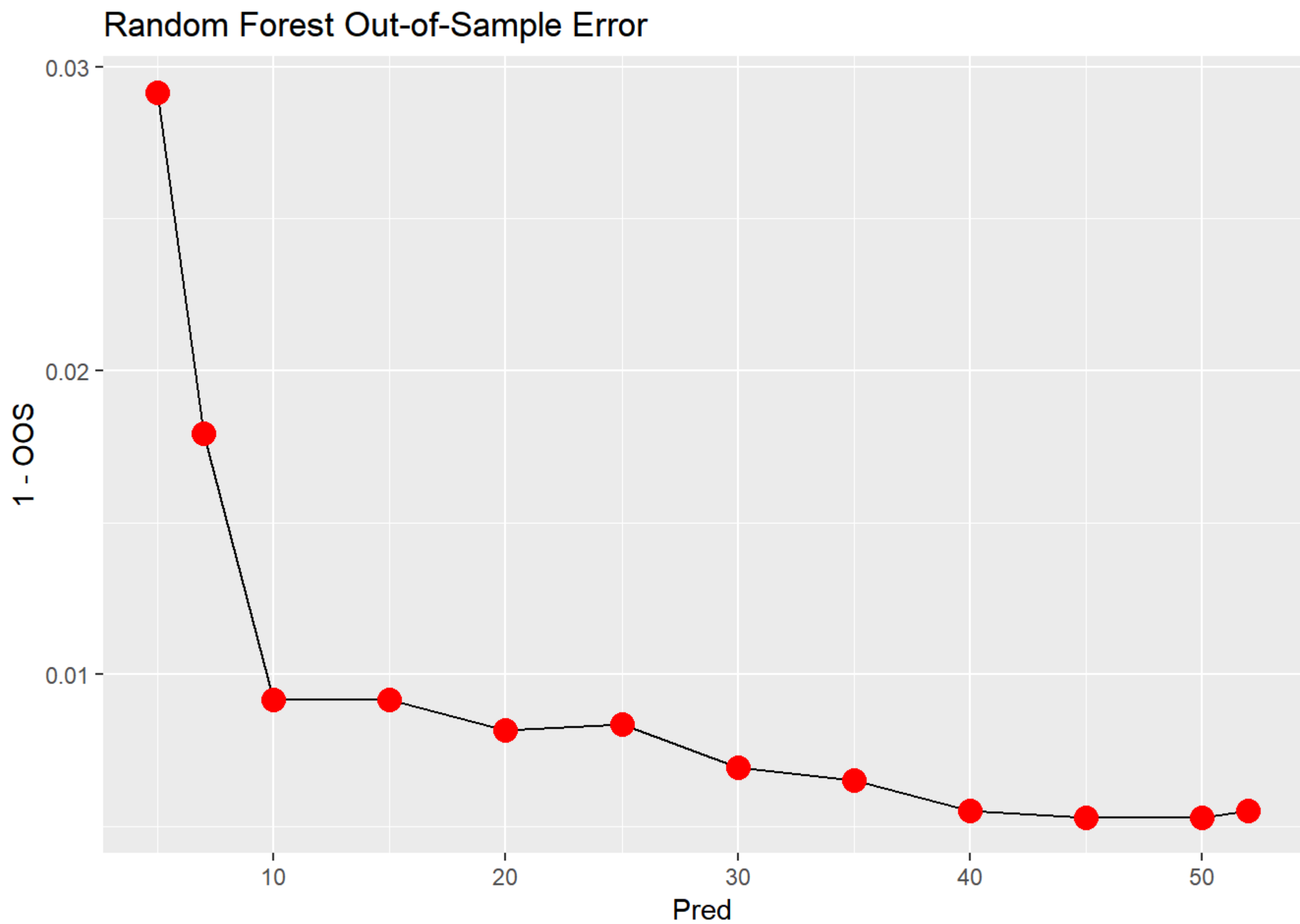
```
  set.seed(333)
  redNum <- 5
  redFeatures <- head(features$id, redNum)
  cols <- append(redFeatures, ncol(TRAIN), after = length(redFeatures))
  training <- TRAIN[, cols]
  fitcontrol <- trainControl(method="repeatedcv", number=5, repeats=3)
  start <- Sys.time()
  fit <- train(classe ~., method = "rf", data = training, trControl = fitControl, importance=TRU
E)
  end <- Sys.time()
  pVal <-predict(fit, validation[, -ncol(validation)])
  fitRF <- fit
  timeRF <- end - start
  accVal <-confusionMatrix(pVal, validation[, ncol(validation)])$overall['Accuracy']
  pQty[nrow(pQty) + 1,] = list(redNum, timeRF, accVal)
  save(pQty, file = "Multi_Parameter.rds")
```

Plot Predictors vs. OOS Error

```
load("Multi_Parameter.rds")
print(pQty)
```

```
##    Pred      Time       OOS
## 1     5  4.037030 0.9708401
## 2    10  6.982997 0.9908238
## 3    15 10.259109 0.9908238
## 4    20 14.318595 0.9918434
## 5    25 18.555994 0.9916395
## 6     7  5.294556 0.9820555
## 7    35 23.467394 0.9934747
## 8    30 19.846859 0.9930669
## 9    40 27.086994 0.9944943
## 10   50 35.079192 0.9946982
## 11   45 32.006012 0.9946982
## 13   52 37.131802 0.9944943
```

```
g = ggplot(pQty, aes(y=1-OOS, x=Pred))
    g = g + geom_line()
    g = g + geom_point(color="RED", size=4)
    g = g + ggtitle("Random Forest Out-of-Sample Error")
    g
```



Run Test Data On Selected Model

1- We run the test data for the selected model and have the following results. 2- These results will be submitted to the Course Project Prediction Quiz.

```
predict(fit, testing)
```

```
##  [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
```

Conclusions

1- A model can be adequately done -in this case- random forrest to predict exercise manner. 2- Our predicted results: B A B A A E D B A A B C B A E E A B B B.

References

1- “How to Evaluate Machine Learning Algorithms with R” by Jason Brownlee on February 1, 2016 in R Machine Learning. 2- 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.