Effects Proper Ways to Lift Weights have on the Human Body

William Moore

December 29, 2019

Introduction

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, my 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).

Steps Used to During our Investigation

We will utilize the below steps to devise results of our investigation:

- Process the data (training and test data).
- Explore the data, primarily focusing on the two paramaters we wish to investigate.
- Evaluate each model, where we try different models to help us answer our questions and select the best model from the group.
- A Conclusion where we answer the questions based on the data.
- Predict the classification of the model on test set data.

```r
URLTraining <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URLTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

TrainingSet <- read.csv(url(URLTraining))
TestSet <- read.csv(url(URLTest))
```

Data Analysis

The below code is to investigate the training and test data sets from the accelerometers used during data collection. We will take a look at the collected data in the training data set.

As the training set data contains a lot of NA values, we will use the below code to clean the training and test data of these values.

```r
maxNAPercent = 20
maxNACount <- nrow(TrainingSet) / 100 * maxNAPercent
removeColumns <- which(colSums(is.na(TrainingSet) | TrainingSet=="") > maxNACount)
training.clean <- TrainingSet[,-removeColumns]
test.clean <- TestSet[,-removeColumns]
```

Since we do not use time related data, the below code is to remove this data from both the Training and Test data sets.

```r
removeColumns <- grep("timestamp", names(training.clean))
training.clean2 <- training.clean[,-c(1, removeColumns )]
test.clean2 <- test.clean[,-c(1, removeColumns )]
```

Now we will use the below code to convert the class of the training and test data to the integer class to help during our investigation.

```r
classeLevels <- levels(training.clean2$classe)
training.clean3 <- data.frame(data.matrix(training.clean2))
training.clean3$classe <- factor(training.clean3$classe, labels=classeLevels)
test.clean3 <- data.frame(data.matrix(test.clean2))
```

The below code is to set the final data set to be used in our investigation.

```r
training.final <- training.clean3
test.final <- test.clean3
```

Since the test set we are using is for the purpose of being the validation set, we will split the current training into a test and training set with the below code.

```r
library(caret)

set.seed(19791108)
classeIndex <- which(names(training.final) == "classe")
partition <- createDataPartition(y=training.final$classe, p=0.75, list=FALSE)
training.subSetTrain <- training.final[partition, ]
training.subSetTest <- training.final[-partition, ]
```

We will now see which variables in our data set are highly correlated with one another in the Training Subset.

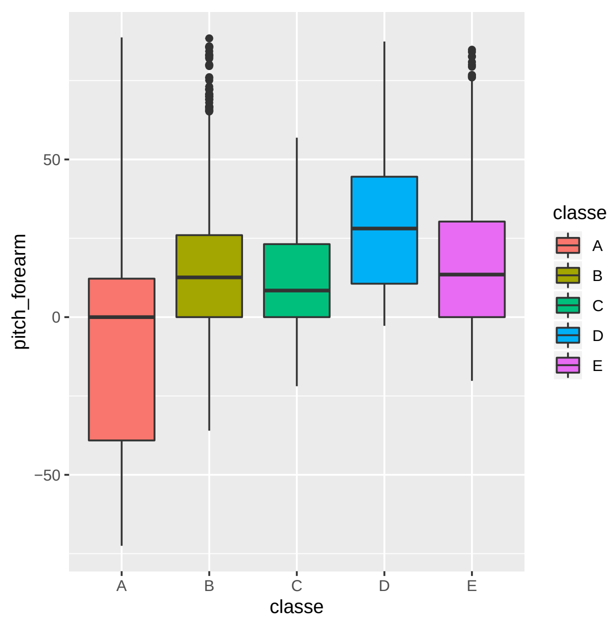
```r
correlations <- cor(training.subSetTrain[, -classeIndex], as.numeric(training.subSetTrain$classe))
bestCorrelations <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.3)
bestCorrelations
```

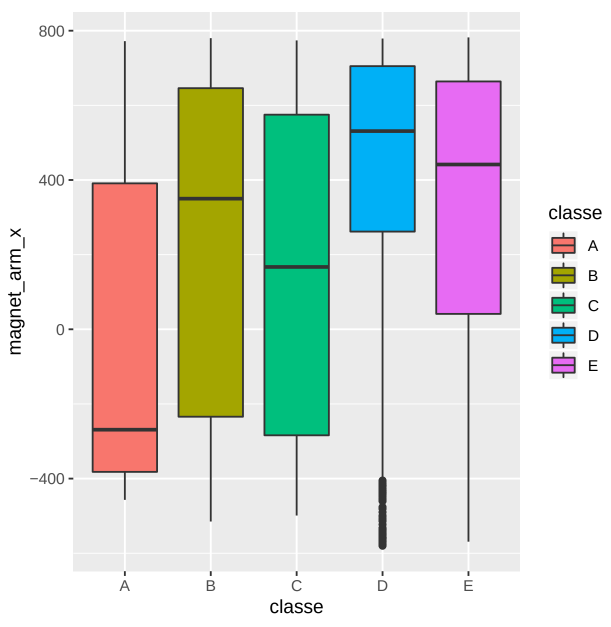
```
##             Var1 Var2      Freq
## 27  magnet_arm_x    A 0.3027579
## 44 pitch_forearm    A 0.3471068
```

We can see that the two variables, "magnet_arm_x" and "pitch_forearm" have the best correlations of the data, but their correlations are at 0.30 and 0.35 respectively.

We will look at the plots of the data to see if it is hard to use these 2 variables as possible simple linear predictors.

```r
library(Rmisc)
library(ggplot2)

p1 <- ggplot(training.subSetTrain, aes(classe,pitch_forearm)) +
  geom_boxplot(aes(fill=classe))
p2 <- ggplot(training.subSetTrain, aes(classe, magnet_arm_x)) +
  geom_boxplot(aes(fill=classe))
multiplot(p1,p2,cols=2)
```



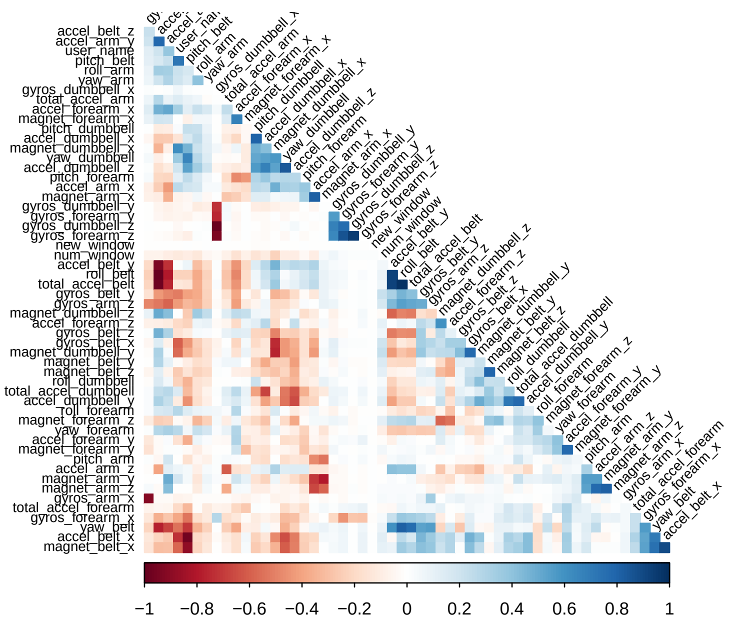


As we can see, there is no distinct seperation of classes possible using only these features that are highly correlated with one another. We will not create other models from this data to get closer to a way of predicting these classe's.

Model Selection

We are now going to identify variables with high correlations amongst each other in our set, so we can possibly exclude them from the pca or training. We will check afterwards to see if these modifications to the dataset make the model more accurate (and perhaps even faster).

```r
library(corrplot)

correlationMatrix <- cor(training.subSetTrain[, -classeIndex])
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.9, exact=TRUE)
excludeColumns <- c(highlyCorrelated, classeIndex)
corrplot(correlationMatrix, method="color", type="lower", order="hclust", tl.cex=0.70, tl.col="black",
```



From the above correlation plot, we can see that there are some variables that are quite correlated with each other.

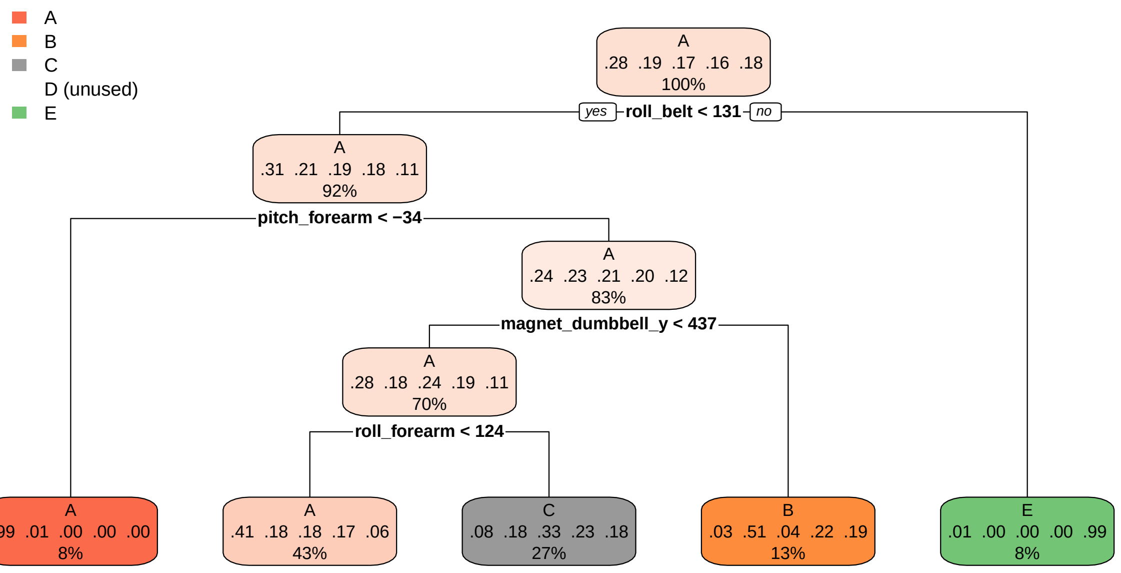
We will further investigate by creating a Decision Tree Model of the training and test data subsets.

```
DT_modfit <- train(classe ~ ., data = training.subSetTrain, method="rpart")

##Below are the prediction in terms of the decision tree model
DT_prediction <- predict(DT_modfit, training.subSetTest)
confusionMatrix(DT_prediction, training.subSetTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1268  410  410  375  137
##          B   32  315   29  145  121
##          C   92  224  416  284  239
##          D    0    0    0    0    0
##          E    3    0    0    0  404
##
## Overall Statistics
##
##                Accuracy : 0.49
##                  95% CI : (0.4759, 0.5041)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.3325
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9090  0.33193  0.48655   0.0000  0.44839
## Specificity            0.6204  0.91732  0.79279   1.0000  0.99925
## Pos Pred Value         0.4877  0.49065  0.33147      NaN  0.99263
## Neg Pred Value         0.9449  0.85124  0.87969   0.8361  0.88948
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2586  0.06423  0.08483   0.0000  0.08238
## Detection Prevalence   0.5302  0.13091  0.25591   0.0000  0.08299
## Balanced Accuracy      0.7647  0.62462  0.63967   0.5000  0.72382
```

```r
rpart.plot(DT_modfit$finalModel, roundint=FALSE)
```



🟥 A
🟧 B
⬜ C
D (unused)
🟩 E
A .28 .19 .17 .16 .18 100%
yes roll_belt < 131 no
A .31 .21 .19 .18 .11 92%
pitch_forearm < −34
A .24 .23 .21 .20 .12 83%
magnet_dumbbell_y < 437
A .28 .18 .24 .19 .11 70%
roll_forearm < 124
A .99 .01 .00 .00 .00 8%
A .41 .18 .18 .17 .06 43%
C .08 .18 .33 .23 .18 27%
B .03 .51 .04 .22 .19 13%
E .01 .00 .00 .00 .99 8%

Since the accuracy of the above Decision Tree Model is 0.49, this is not up to desired level of accuracy we need.

We will now create random forests to determine which model would be best to use with our data.

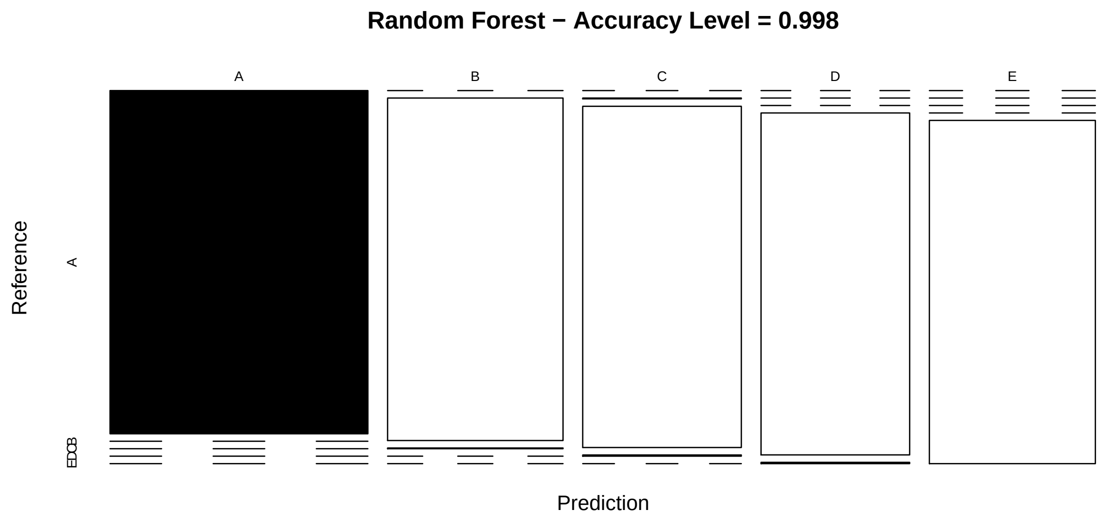
```r
RF_modfit <- randomForest(classe ~ ., data = training.subSetTrain, ntree = 100)

##Below are predictions in terms of the Random Forest Model
```

```r
RF_prediction <- predict(RF_modfit, training.subSetTest)
RF_pred_conf <- confusionMatrix(RF_prediction, training.subSetTest$classe)
RF_pred_conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  947    2    0    0
##          C    0    2  853    3    0
##          D    0    0    0  801    3
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.9963, 0.999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9977   0.9963   0.9967
## Specificity            1.0000   0.9995   0.9988   0.9993   1.0000
## Pos Pred Value         1.0000   0.9979   0.9942   0.9963   1.0000
## Neg Pred Value         1.0000   0.9995   0.9995   0.9993   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1739   0.1633   0.1831
## Detection Prevalence   0.2845   0.1935   0.1750   0.1639   0.1831
## Balanced Accuracy      1.0000   0.9987   0.9982   0.9978   0.9983
```

Below is the code for the graph of the random forest model.

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



Per the confusion matrix, we can see that the Prediction accuracy of the Random Forest Model is 99%, which satisfies our model.

Results and Conclusion

From the Overall Statistics data, we can conclude that the Random Forest model has definitely more accuracy than Data tree Model. Hence we will be selecting Random Forest model for final prediction.

Final Prediction Model

Below is the code for our final to apply to the original comparison test data.

```r
Final_RF_prediction <- predict(RF_modfit, test.final)
Final_RF_prediction
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```