Practical Machine Learning Final Project

Malachi Jones

April 18, 2021

Executive Summary

This report attempts to predict what technique is used for a lift. It uses data collected from wearable fitness devices like FitBit and Nike FuelBand. A few different machine learning algorithms are applied to the data including decision trees, random forests and a generalized boosting model. Using cross-validation and the testing/training split method, random forests was determined to be the best algorithm for our problem.

Introduction

Wearable fitness devices like Nike FuelBand and FitBit has made personal activity data easier and cheaper to collect. A lot of research using this data has focused on how much of a certain activity is performed. In this report, the focus will be on the quality of a certain activity, in this case dumbbell bicep curls.

Data

Below is an excerpt from the website where the data is found:

"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)."

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz6sUu7q3Of

Citation of Data:

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.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz6sUuG89pV

Data Processing and Cleaning

First, the data set was split into a training set and test set. About 70% of the data was put into the training set while 30% of the data is set aside for the test set.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.1
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.3
```

```
## Loading required package: tibble
```

```
## Warning: package 'tibble' was built under R version 3.6.3
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
training_set <- read.csv("C:\\Users\\mjone\\OneDrive\\Documents\\JHU Machine Learning\\pml-training.csv

set.seed(1000)

inTrain <- createDataPartition(y=training_set$classe, p = 0.7, list = FALSE)

myTrain <- training_set[inTrain,]
myTest <- training_set[-inTrain,]

dim(myTrain)
```

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

```r
dim(myTest)
```

```
## [1] 5885  160
```

Next, the data was cleaned to get rid of unnecessary or unusable variables. Variables like time stamp and username was removed. Also, variables with near zero variane were also removed. Also, columns with over 70% of their values missing were removed. After cleaning, there are 52 explanatory variables.

```
myTrain <- myTrain[c(-1, -2, -3, -4, -5, -6, -7)]
myTest <- myTest[c(-1, -2, -3, -4, -5, -6, -7)]

nzvVariables <- names(myTrain)[nearZeroVar(myTrain)]
nzvInd <- match(nzvVariables, names(myTrain))

myTrain <- myTrain[-nzvInd]
myTest <- myTest[-nzvInd]

emptyColumns <- which(colSums(is.na(myTrain) | myTrain == "") >= 0.7*dim(myTrain)[1])
myTrain <- myTrain[,-emptyColumns]
myTest <- myTest[,-emptyColumns]

dim(myTrain)
```

```
## [1] 13737    53
```

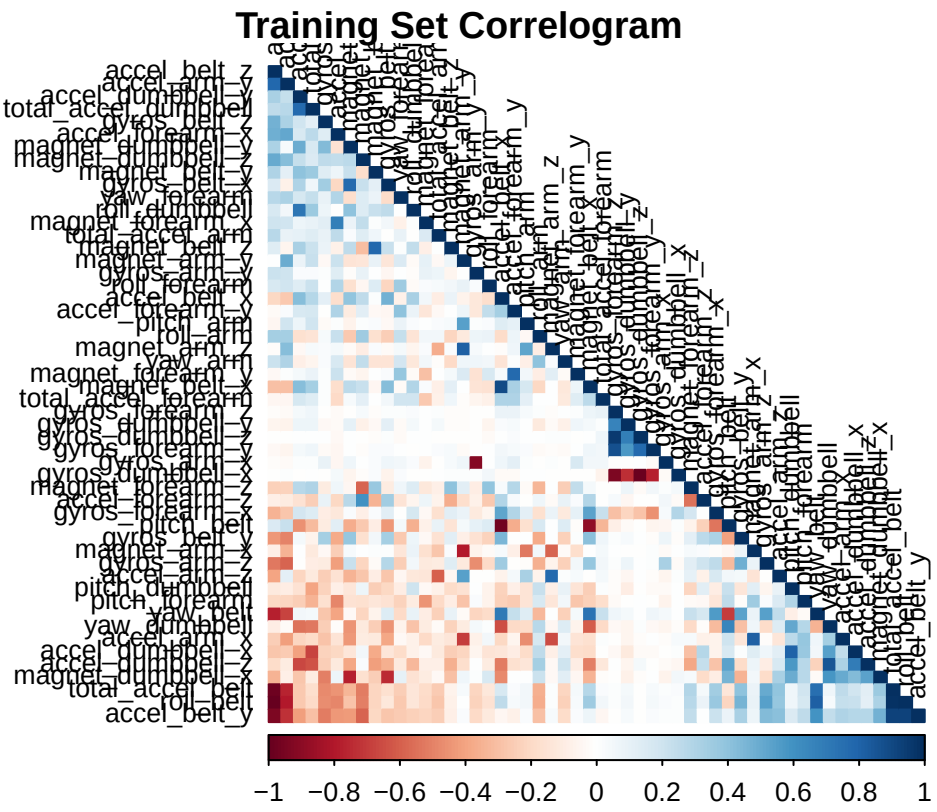
```
dim(myTest)
```

```
## [1] 5885   53
```

Exploratory Analysis

Next, the the relationships between our independent variables is explored using the correlogram below. We see a fair amount of variable are highly correlated.

```
corMat <- cor(myTrain[,-53])
corrplot(corMat, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0,0,0), mar = c(1,1,1,1), title = "Training Set Correlogram")
```

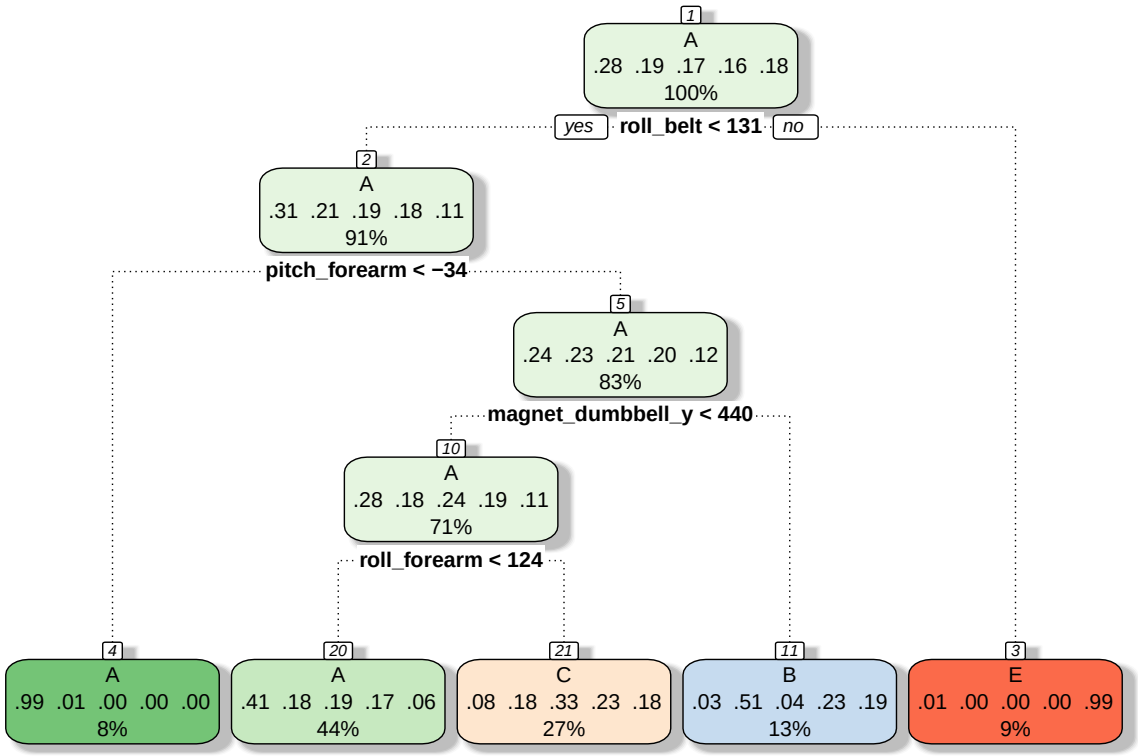


Model Building

Three models, a decision tree, a random forests, and a generalized boosted model, will be created to make predictions on how the lifts were performed. Each model will be created using 5-fold cross-validation. Cross-validation is used to help against overfitting our model to the training data, therefore making it worse when making predictions on data outside the sample.

```
crossValid <- trainControl(method = 'cv', number = 5)

DT_Model <- train(classe ~ ., data = myTrain, method = "rpart", trControl = crossValid)
fancyRpartPlot(DT_Model$finalModel)
```




```
RF_Model <- train(classe ~ ., data = myTrain, method = "rf", trControl = crossValid, verbose = FALSE)
```

```
GBM_Model <- train(classe ~ ., data = myTrain, method = "gbm", trControl = crossValid, verbose = FALSE)
```

Now that the models have been created, they will be used to make predictions on the test data that was set aside in the beginning and was not used to train the data. These predictions are then compared to the actual values. The accuracy for the decision tree model is 0.493. This means our out of sample error can be expected to be 0.507 for the decision tree. The accuracy for the random forests model and generalized boosted model are 0.993 and 0.964, respectively. Therefore, random forests appears to be the best model with an expected out of sample error of 0.007.

```
DT_Predict <- predict(DT_Model, newdata = myTest)
RF_Predict <- predict(RF_Model, newdata = myTest)
GBM_Predict <- predict(GBM_Model, newdata = myTest)

DT_acc <- confusionMatrix(myTest$classe, DT_Predict)$overall[1]
RF_acc <- confusionMatrix(myTest$classe, RF_Predict)$overall[1]
GBM_acc <- confusionMatrix(myTest$classe, GBM_Predict)$overall[1]

DT_acc
```

```
##  Accuracy
## 0.4929482
```

```
RF_acc
```

```
##  Accuracy
## 0.9926933
```

```
GBM_acc
```

```
##  Accuracy
## 0.9641461
```

Applying Random Forests Model to Validation Data

Next, the random forests model was used to make predictions on 20 observations.

```
validData <- read.csv("C:\\Users\\mjone\\OneDrive\\Documents\\JHU Machine Learning\\pml-testing.csv")
validData <- validData[c(-1, -2, -3, -4, -5, -6, -7)]
validData <- validData[-nzvInd]
validData <- validData[,-emptyColumns]
dim(validData)
```

```
## [1] 20 53
```

```
predictions <- predict(RF_Model, newdata = validData)
predictions
```

```
##  [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
```