Prediction Assignment Writeup

author: Micha Bouts
date: February 21, 2016

Executive Summary

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. 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, we use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the **quality** of the physical exercise. Participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways, defined as the *classe* outcome.

Both a training and test set were made available. While exploring the dataset we reduced the number of variables from 160 to merely 45 on which we trained a couple of models. The *Random Forests Model* turned out giving the highest accuracy with an out of sample error of 0.8%. This performance was sufficiently good for use as a **prediction** tool in assessing the **quality** of the Unilateral Dumbbell Biceps Curl on 20 test cases.

Initialization

As a starter we set the working directory and load a number of libraries which we'll use during the data analysis.

```
setwd("~/R/Coursera/Practical_Machine_Learning"); rm(list = ls())
library(knitr);library(dplyr);library(corrplot);library(rattle); library(randomForest)
library(rpart);library(caret);library(parallel);library(doParallel)
```

Getting Data from Open Source

A Weight Lifting Exercise Dataset is public available at http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). Both a training and a test set are provided.

Courtesy to

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.

After downloading the datasets we load them in the R environment.

```
if (!file.exists("data")) {

        dir.create("data")

        fileUrlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        fileUrlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

        download.file(fileUrlTrain, destfile = "./data/pml-training.csv")
        download.file(fileUrlTest, destfile = "./data/pml-testing.csv")

        }

list.files("./data")
```

```
## [1] "pml-testing.csv"  "pml-training.csv"
```

```
training <- read.csv("./data/pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("./data/pml-testing.csv", na.strings = c("NA", ""))
```

Exploratory Data Analysis

```
dim(training); dim(testing)
```

```
## [1] 19622   160
```

```
## [1]  20 160
```

```
str(training, list.len = 10)
```

```
## '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 132308423
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4
##  $ 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 ...
##   [list output truncated]
```

```
table(training$classe)
```

```
## 
##    A    B    C    D    E 
## 5580 3797 3422 3216 3607
```

The training and test datasets contain 19622 and 20 observations respectively. Both datasets have 160 variables. The outcome variable of interest *classe* is a qualitative index of how well the Unilateral Dumbbell Biceps Curl is performed. The table above shows the number of training data per type of *classe*.

- Class A is according to the specification
- Class B is throwing the elbows to the front
- Class C is lifting the dumbbell only halfway
- Class D is lowering the dumbbell only halfway
- Class E is throwing the hips to the front

The intention of this data analysis is to predict the *classe* for each observation in the given test set. We don't look at the test set for now. We get back to that at the very end after having selected the prediction model.

The first 7 variables *X*, *user_name*, *raw_timestamp_part_1*, *raw_timestamp_part_2*, *cvtd_timestamp*, *new_window* and *num_window* represent general background information. It might be there is a time related effect in our data. To avoid unnecessary complexity we remove these qualifiers for now from our scope of interest.

As pointed out below 100 out of 160 variables have large quantities of missing values (NA's). Removing these variables dramatically reduces the dataset complexity without missing precious information.

```
table(colSums(!is.na(training)))
```

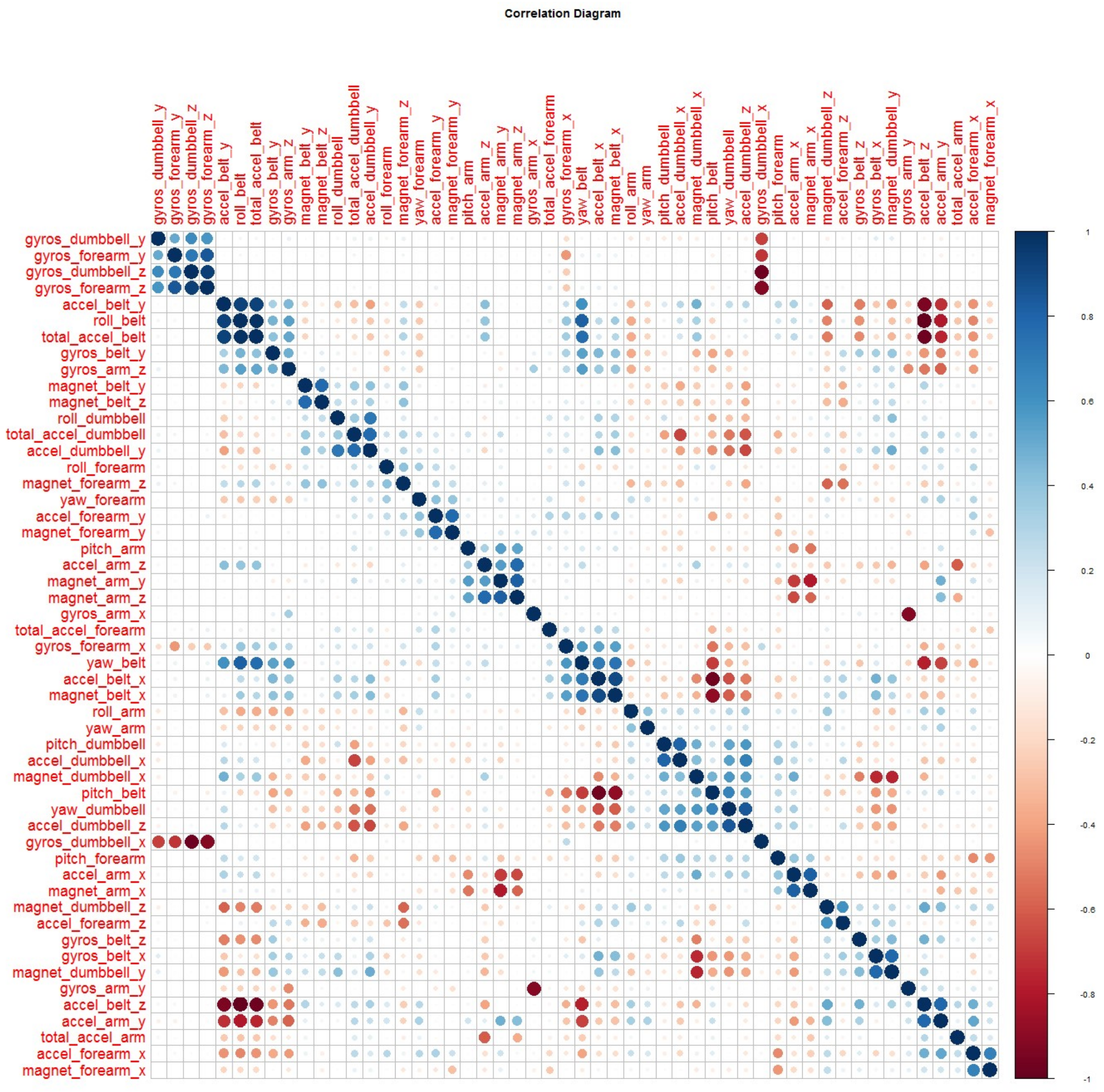
```
## 
##   406 19622 
##   100    60
```

```
dftraining1 <- select(training, -c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_ti
dftraining1 <- select(dftraining1, which(colSums(!is.na(dftraining1)) == 19622))

dim(dftraining1)
```

```
## [1] 19622    53
```

The downsized dataset has only 53 variables. Furthermore, predictor variables which are strongly correlated with each other don't give extra information, yet do pile up the actual error. Hence we better filter them out. A correlation diagram is meaningful for identifying highy correlated regressors. We cluster them by correlation index. Dark blue and dark red bullets on the diagram below represent strongly related regressors which we remove at a cutoff correlation value of 85% or higher. This level is arbitrary chosen.

```
trainCor1 <- cor(dftraining1[-53])
corrplot(trainCor1, order = "hclust", tl.cex = 1.5, mar = c(5, 4, 2, 2), title = "Correlation Diagra
```



```
highlyCorTraining85 <- findCorrelation(trainCor1, cutoff = 0.85, names = FALSE)
dftraining2 <- dftraining1[ ,-highlyCorTraining85]

dim(dftraining2)
```

```
## [1] 19622    45
```

As such we reach a final dataset with 45 variables. Before we start model building let's first split this dataset in a smaller training set and a validation set. We sacrifice part of the initial training data in order to create validation data on which we assess the out of sample error rate.

```
inTrain <- createDataPartition(y = dftraining2$classe, p = 0.75, list = FALSE)
trainingSet <- dftraining2[ inTrain, ]
validationSet <- dftraining2[-inTrain, ]

dim(trainingSet); dim(validationSet)
```

```
## [1] 14718    45
```

```
## [1] 4904   45
```

Model Building

Because the outcome *classe* is a categorical variable we'll examine models which can deal with this type of outcome. Therefore we select algorithms based on a *Tree* and a *Random Forests Model*.

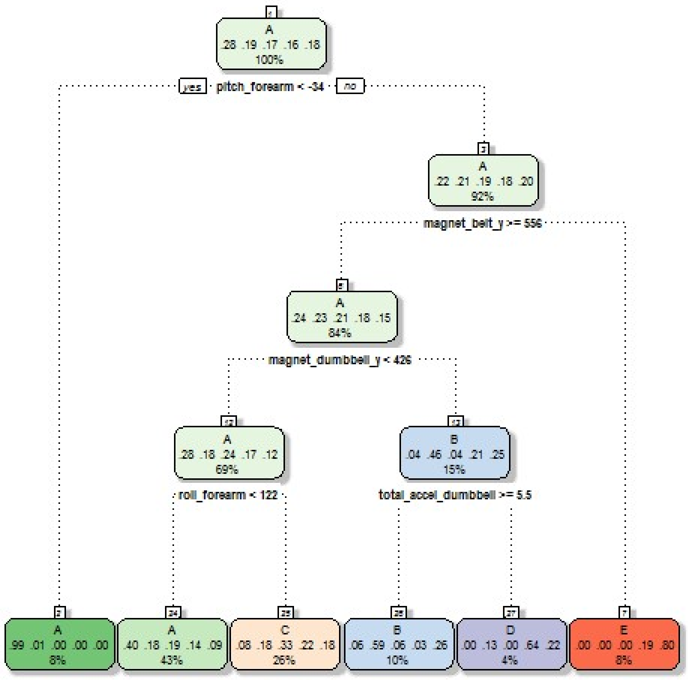
Tree Model

This classification tree finds the subsequent variables that best separates the outcome. This continues until the group is small enough or the classification is reasonable pure, as such mimimizing classification errors.

```
set.seed(333)

modFitTree <- train(classe ~ ., method = "rpart", data = trainingSet)
print(modFitTree$finalModel)
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##    2) pitch_forearm< -33.95 1214     9 A (0.99 0.0074 0 0 0) *
##    3) pitch_forearm>=-33.95 13504 10524 A (0.22 0.21 0.19 0.18 0.2)
##      6) magnet_belt_y>=555.5 12384  9406 A (0.24 0.23 0.21 0.18 0.15)
##       12) magnet_dumbbell_y< 426.5 10213  7332 A (0.28 0.18 0.24 0.17 0.12)
##         24) roll_forearm< 121.5 6392  3822 A (0.4 0.18 0.19 0.14 0.089) *
##         25) roll_forearm>=121.5 3821  2541 C (0.081 0.18 0.33 0.22 0.18) *
##       13) magnet_dumbbell_y>=426.5 2171  1178 B (0.045 0.46 0.043 0.21 0.25)
##         26) total_accel_dumbbell>=5.5 1535   627 B (0.063 0.59 0.06 0.026 0.26) *
##         27) total_accel_dumbbell< 5.5 636   227 D (0 0.13 0.0031 0.64 0.22) *
##      7) magnet_belt_y< 555.5 1120   221 E (0.0018 0.0036 0.00089 0.19 0.8) *
```

```
fancyRpartPlot(modFitTree$finalModel, sub = "Classification Tree with 6 Final Node Classe Outcomes")
```



Classification Tree with 6 Final Node Classe Outcomes

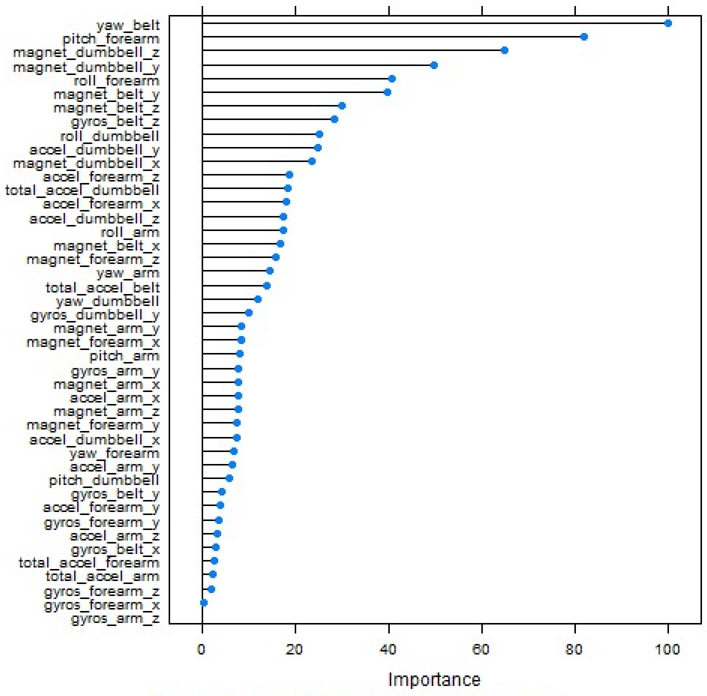
Random Forests Model

We make use of multi-core parallel processing to circumvent the limitation of a 6GB RAM memory at use. Furthermore we control the *train* function by selection a 5-fold *cv* cross validation. Also this will keep the processor time in check.

```
# configure parallel processing

cluster <- makeCluster(detectCores() -1)
registerDoParallel(cluster)

set.seed(123)

modFitRFcv <- train(classe ~ ., method = "rf", data = trainingSet,
                    trControl = trainControl(method = "cv", number = 5, allowParallel = TRUE))

# De-register parallel processing cluster

stopCluster(cluster)

modFitRFcv
```

```
## Random Forest
##
## 14718 samples
##    44 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11773, 11775, 11775, 11775
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9899444  0.9872784  0.002868633  0.003629907
##   23    0.9901481  0.9875372  0.001906859  0.002412797
##   44    0.9848484  0.9808314  0.002410690  0.003050962
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 23.
```

```
plot(varImp(modFitRFcv), sub = "Random Forests Model Most Important Variables")
```



Random Forests Model Most Important Variables

Cross Validation & Expected Out of Sample Error

In order to calculate the expected out of sample error we perform a cross validation with the *validationSet*. This dataset was parked till now, so both models haven't seen it yet.

Tree Model Out of Sample Error

```
predTree <- predict(modFitTree, validationSet)

cmTree <- confusionMatrix(predTree, validationSet$classe)
cmTree$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1273  358  372  302  187
##          B   26  324   34   21  148
##          C   95  236  447  283  227
##          D    0   31    1  146   48
##          E    1    0    1   52  291
```

```
accuracyTree <- cmTree$overall[c(1, 3, 4)]
accuracyTree
```

```
##      Accuracy AccuracyLower AccuracyUpper
##     0.5059135     0.4918185     0.5200015
```

```
outofsampleerrorTree <- 1 - accuracyTree[1]
outofsampleerrorTree[[1]]
```

```
## [1] 0.4940865
```

The confusion matrix shows a siginificant number of misclassification errors where the *Prediction* doesn't match up with the *Reference*. An accuracy of 50.6% or put differently an out of sample error rate of 49.4% is pretty poor.

Random Forests Model Out of Sample Error

```
predRF <- predict(modFitRFcv, validationSet)

cmRF <- confusionMatrix(predRF, validationSet$classe)
cmRF$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    3    0    0    0
##          B    1  936    3    0    0
##          C    0    9  847   13    0
##          D    0    0    5  788    3
##          E    0    1    0    3  898
```

```
accuracyRF <- cmRF$overall[c(1, 3, 4)]
accuracyRF
```

```
##      Accuracy AccuracyLower AccuracyUpper
##     0.9916395     0.9886749     0.9939939
```

```
outofsampleerrorRF <- 1 - accuracyRF[1]
outofsampleerrorRF[[1]]
```

```
## [1] 0.008360522
```

This time the confusion matrix only has a small number of misclassifiers. With an accuracy of 99.2% and a respective out of sample error of 0.8% we proceed with the random forests model as the chosen one for our prediction.

Reasoning for Choices

Because the outcome *classe* is a categorical variable we started off with the *Tree Model* with its good interpretability. However, with such a poor *Tree Model* accuracy we moved on to the *Random Forests Model* to average the trees. This with the expectation to pump-up the accuracy.

Initially, our first attempt was to use a default bootstrapping method within the *Random Forests Model*. This took way too much processor time and with a limit of 6GB RAM memory we switched to parallel processing combined with a 5-fold cross validation. This significanlty reduced the processing time to a couple of minutes while maintaining a sufficient high accuracy.

We also considered running a *Boosting Model*, yet due to *Random Forests Model* processing issues we refrained from doing so.

Predicting Test Cases

We select the best model, which is the *Random Forests Model*. This final model is used to predict the *classe* outcome on the test cases.

```
predTest <- predict(modFitRFcv, testing)
predTest
```

```
##  [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

For predicting the quality of performing the Unilateral Dumbbell Biceps Curls we apply a *Random Forests Model* with a prediction accuracy of 99.2%. This model is subsequently verified on 20 test cases with the goal of getting a quality label on the exercise. Ultimately this information can be used in a feedback loop towards the practitioner.