# Practical Machine Learning Course Project

Steve Anderson

Sunday, January 18, 2015

# Practical Machine Learning Course Project

Synopsis:

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).

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set and some of the other variables to predict with. Also this document will describe how I built my model, how I used cross validation, and what I think the expected out of sample error is, and why I made the choices I did. I will also use my prediction model to predict 20 different test cases.

The overal goal of this project is to predict the quality of performing curls by examining the manner of performing unilateral dumbbell biceps curls based on data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The 5 possible methods include:

- A: Doing curls exactly according to the specification
- B: Throwing the elbows to the front while doing curls
- C: lifting the dumbbell only halfway while doing curls
- D: lowering the dumbbell only halfway while doing curls
- E: throwing the hips to the front while doing curls

Load libraries and setup working directory

```
rm(list = ls(all = TRUE))

setwd('C:/Users/Steve/Documents/machinelearning')

library(caret)

trainingRaw <- read.csv(file="https://d396qusza40orc.cloudfront.net/predmachlearn/pm
l-training.csv", header=TRUE, as.is = TRUE, stringsAsFactors = FALSE, sep=',', na.st
rings=c('NA','','#DIV/0!'))
testingRaw <- read.csv(file="https://d396qusza40orc.cloudfront.net/predmachlearn/pml
-testing.csv", header=TRUE, as.is = TRUE, stringsAsFactors = FALSE, sep=',', na.stri
ngs=c('NA','','#DIV/0!'))

trainingRaw$classe <- as.factor(trainingRaw$classe)
```

Examine the data

```
## 'data.frame':  19622 obs. of  160 variables:
##  $ X                       : chr  "1" "2" "3" "4" ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 132
3084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 36829
6 440390 484323 484434 ...
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/20
11 11:23" "05/12/2011 11:23" ...
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ 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      : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt       : chr  "" "" "" "" ...
##  $ skewness_roll_belt      : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" "" "" ...
##  $ skewness_yaw_belt       : chr  "" "" "" "" ...
##  $ 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            : chr  "" "" "" "" ...
##  $ 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            : chr  "" "" "" "" ...
##  $ 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      : chr  "" "" "" "" ...
##  $ 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 -0.0
2 -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 -3
08 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -1
28 ...
##  $ 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 -1
61 ...
##  $ 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 -2
88 ...
##  $ 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 -1
24 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -3
76 ...
##  $ 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       : chr  "" "" "" "" ...
##  $ kurtosis_picth_arm      : chr  "" "" "" "" ...
##  $ kurtosis_yaw_arm        : chr  "" "" "" "" ...
##  $ skewness_roll_arm       : chr  "" "" "" "" ...
##  $ skewness_pitch_arm      : chr  "" "" "" "" ...
##  $ skewness_yaw_arm        : chr  "" "" "" "" ...
##  $ 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  : chr  "" "" "" "" ...
##  $ kurtosis_picth_dumbbell : chr  "" "" "" "" ...
##  $ kurtosis_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ skewness_roll_dumbbell  : chr  "" "" "" "" ...
##  $ skewness_pitch_dumbbell : chr  "" "" "" "" ...
##  $ skewness_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ 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        : chr  "" "" "" "" ...
##  $ 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        : chr  "" "" "" "" ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Cleaning variables

After investigating all the variables of the sets, it’s possible to see that there are a lot of values NA or useless or empty variables for the prediction. It’s request to compute the prediction only on the accelerometers values of belt, forearm, arm and dumbell. So, the non-accelerometer measures are discard with the useless variables.

```
NAindex <- apply(trainingRaw,2,function(x) {sum(is.na(x))})
trainingRaw <- trainingRaw[,which(NAindex == 0)]
NAindex <- apply(testingRaw,2,function(x) {sum(is.na(x))})
testingRaw <- testingRaw[,which(NAindex == 0)]
```

Preprocessing variables

```
v <- which(lapply(trainingRaw, class) %in% "numeric")

preObj <-preProcess(trainingRaw[,v],method=c('knnImpute', 'center', 'scale'))
trainLess1 <- predict(preObj, trainingRaw[,v])
trainLess1$classe <- trainingRaw$classe

testLess1 <-predict(preObj,testingRaw[,v])
```

Removing the non zero variables

Removing the variables with values near zero, that means that they have not so much meaning in the predictions

```
nzv <- nearZeroVar(trainLess1,saveMetrics=TRUE)
trainLess1 <- trainLess1[,nzv$nzv==FALSE]

nzv <- nearZeroVar(testLess1,saveMetrics=TRUE)
testLess1 <- testLess1[,nzv$nzv==FALSE]
```

Create cross validation set

The training set is divided in two parts, one for training and the other for cross validation

```
set.seed(12031987)

inTrain = createDataPartition(trainLess1$classe, p = 3/4, list=FALSE)
training = trainLess1[inTrain,]
crossValidation = trainLess1[-inTrain,]
```

Train model

Train model with random forest due to its highly accuracy rate. The model is build on a training set of 28 variables from the initial 160. Cross validation is used as train control method.

```
modFit <- train(classe ~., method="rf", data=train, trControl=trainControl(method='c
v'), number=5, allowParallel=TRUE )
```

```
## Time difference of 13.05325 mins

## Random Forest
##
## 14718 samples
##    27 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
##
## Summary of sample sizes: 13246, 13245, 13248, 13245, 13247, 13246, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     0.993     0.991  0.00116      0.00147
##   14    0.992     0.99   0.0028       0.00354
##   27    0.989     0.987  0.00353      0.00446
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Accuracy on training set and cross validation set

Following the computation on the accuracy of trainig and cross validation set

Training set:

```
trainingPred <- predict(modFit, training)
confusionMatrix(trainingPred, training$classe)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 4185    0    0    0    0
         B    0 2848    0    0    0
         C    0    0 2567    0    0
         D    0    0    0 2412    0
         E    0    0    0    0 2706

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9997, 1)
    No Information Rate : 0.2843
    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   1.0000
Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Cross validation set

```
cvPred <- predict(modFit, crossValidation)
confusionMatrix(cvPred, crossValidation$classe)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1392    3    0    0    0
         B    2  944    2    0    0
         C    0    2  852    3    0
         D    0    0    1  801    3
         E    1    0    0    0  898

Overall Statistics

               Accuracy : 0.9965
                 95% CI : (0.9945, 0.998)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9956
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9978   0.9947   0.9965   0.9963   0.9967
Specificity            0.9991   0.9990   0.9988   0.9990   0.9998
Pos Pred Value         0.9978   0.9958   0.9942   0.9950   0.9989
Neg Pred Value         0.9991   0.9987   0.9993   0.9993   0.9993
Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
Detection Rate         0.2838   0.1925   0.1737   0.1633   0.1831
Detection Prevalence   0.2845   0.1933   0.1748   0.1642   0.1833
Balanced Accuracy      0.9985   0.9969   0.9976   0.9976   0.9982
```

RESULTS

Predictions on the real testing set

```
testingPred <- predict(modFit, testLess1)
testingPred
```

```
 [1] B A B A A E D B A A B C B A E E A B B B
```