Predicting Proper Exercise Technique From Personal Wareable Device Data

Hadrien Dykiel

Septempber 24, 2016

Executive Summary

Data collected from wearable devices like fitbit can be used to predict if the owner of the device is using proper form during weight-lifting exercises. The ability to determine when someone is doing an exercise incorrectly and subsequently notify them can help prevent serious injury, bettering the health of individuals and avoiding costly medical services. This paper includes the model we have developped as well as the methodology for how it was built.

System

This analysis was run with the following system specifications:

```
#Load required packages
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.1
```

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

```
## Loading required package: ggplot2
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.1
```

```
R.Version()
```

```
## $platform
## [1] "x86_64-w64-mingw32"
##
## $arch
## [1] "x86_64"
##
## $os
## [1] "mingw32"
##
## $system
## [1] "x86_64, mingw32"
##
## $status
## [1] ""
```

```
##
## $major
## [1] "3"
##
## $minor
## [1] "3.0"
##
## $year
## [1] "2016"
##
## $month
## [1] "05"
##
## $day
## [1] "03"
##
## $`svn rev`
## [1] "70573"
##
## $language
## [1] "R"
##
## $version.string
## [1] "R version 3.3.0 (2016-05-03)"
##
## $nickname
## [1] "Supposedly Educational"
```

Exploratory Analysis

Our data contains 160 variables, incuding our target variable. Before we begin to build our model, it's a good idea to learn a little bit more about our data. We begin by printing the variable names along with their class and sample values.

```
#Load training & test data sets
df <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", na.strings=c("NA
test <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", na.strings=c("N

#Inspect data
str(df)
```

```
## '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 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ 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      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ 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            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ 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       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell: num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
#Correlation
# cor(df)

#Print possible values for target var
unique(df$classe)
```

```
## [1] A B C D E
## Levels: A B C D E
```

Our target variable is categorical, so a decision tree type model will be a good start.

Methodology

We begin by splitting our data into a training and test set (called validation) so we can cross-validate our model. We also remove variables that contain only NAs since they do not have any predictive power.

```r
#Remove columns in the test table that are entirely NAs
df <- df[ ,colSums(is.na(df)) != nrow(df)]
test <- test[ ,colSums(is.na(test)) != nrow(test)]

#Remove the same columns for the test set
test <- test[ ,which(names(test) %in% names(df))]

#Remove columns in df that are also not in test set
df <- df[ ,which(names(df) %in% c(names(test), "classe"))]

#Data partition
index <- createDataPartition(df$classe, times=1, p=0.5)[[1]]
training <- df[index, ]
validation <- df[-index, ]
```

Next, we create a decision tree model with the default specifications. Because our target variable 'classe' is a factor, we use the method 'class' for building model and a confusion matrix to evaluate it.

```r
#Model
tree <- rpart(classe ~ ., data=training, method="class")

#Predict
pred1 <- predict(tree, validation, type = "class")

#Evaluate the model
confusionMatrix(validation$classe, pred1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2790    0    0    0    0
##          B    1 1897    0    0    0
##          C    0    0 1711    0    0
##          D    0    0    0 1608    0
##          E    0    0    0    1 1802
##
## Overall Statistics
##
##                Accuracy : 0.9998
##                  95% CI : (0.9993, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   1.0000   1.0000   0.9994   1.0000
## Specificity            1.0000   0.9999   1.0000   1.0000   0.9999
## Pos Pred Value         1.0000   0.9995   1.0000   1.0000   0.9994
```

```
## Neg Pred Value         0.9999   1.0000   1.0000   0.9999   1.0000
## Prevalence             0.2845   0.1934   0.1744   0.1640   0.1837
## Detection Rate         0.2844   0.1934   0.1744   0.1639   0.1837
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9998   0.9999   1.0000   0.9997   0.9999
```

Our model proved to be accurate on this validation set, however it might perform differently on other test sets. Cross-validation can be used to get a better estimate of our model's performance. It entails training our model mutliple times, using the same specifications, each time using a different section of our data as our test or validation data set. It is typical to use 5-fold validation, where we'll create 5 different models and average the model performance measures, like accuracy, to give us a better idea of how good our model actually is. The goal is to then tune our model to maximize the average performance measures. In some cases, a blended model can be created but it is out of the scope of this exercise. Instead, our goal is to use cross-validation to simply validate our model's performance. Although some cross-validation functions exist in R, we will write our own function to explicitly show how cross-validation works.

```r
#Set random seed for reproducibility
set.seed(123)

#Initialize the accs vector
accs <- rep(0,6)

#Shuffle the dataframe in case it was sorted in any particular order
n <- nrow(df)
shuffled <- df[sample(n),]

for (i in 1:6) {
  # These indices indicate the interval of the test set
  indices <- (((i-1) * round((1/6)*nrow(shuffled))) + 1):((i*round((1/6) * nrow(shuffled))))

  # Exclude them from the train set
  train <- shuffled[-indices,]

  # Include them in the test set
  validation <- shuffled[indices,]

  # A model is learned using each training set
  tree <- rpart(classe ~ ., train, method = "class")

  # Make a prediction on the test set using tree
 pred <- predict(tree, validation, type="class")

  # Assign the confusion matrix to conf
conf <- table(validation$classe, pred)

  # Assign the accuracy of this model to the ith index in accs
  accs[i] <- sum(diag(conf))/sum(conf)
}

# Print out the mean of accs
meanAccuracy <- mean(accs)
print(meanAccuracy)
```

```
## [1] 0.9997961
```

Our model has a mean accuracy of 0.9997961. Although our model could be further optimized, the current model is sufficient for our purposes. Let's apply it to our test set:

```
predTest <- predict(tree, test)
results <- data.frame(test$user_name, prediction = predTest)
print(results)
```

```
##    test.user_name prediction.A prediction.B prediction.C prediction.D
## 1           pedro            1            0            0            0
## 2          jeremy            1            0            0            0
## 3          jeremy            1            0            0            0
## 4          adelmo            1            0            0            0
## 5          eurico            1            0            0            0
## 6          jeremy            1            0            0            0
## 7          jeremy            1            0            0            0
## 8          jeremy            1            0            0            0
## 9        carlitos            1            0            0            0
## 10        charles            1            0            0            0
## 11       carlitos            1            0            0            0
## 12         jeremy            1            0            0            0
## 13         eurico            1            0            0            0
## 14         jeremy            1            0            0            0
## 15         jeremy            1            0            0            0
## 16         eurico            1            0            0            0
## 17          pedro            1            0            0            0
## 18       carlitos            1            0            0            0
## 19          pedro            1            0            0            0
## 20         eurico            1            0            0            0
##    prediction.E
## 1             0
## 2             0
## 3             0
## 4             0
## 5             0
## 6             0
## 7             0
## 8             0
## 9             0
## 10            0
## 11            0
## 12            0
## 13            0
## 14            0
## 15            0
## 16            0
## 17            0
## 18            0
## 19            0
## 20            0
```