

Predicting quality of exercise movements using data from accelerometers

Andy Vallebuena

February 12, 2019

Executive summary

This report has the objective of fitting a model that predicts the quality of an activity performed at a specific point in time. It uses the Weight Lifting Exercises Dataset, which investigates how well an activity was performed by the wearer of accelerometers. For this dataset, six 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).” While Class A identifies the correct, specified execution of the exercise, the other four classes capture common mistakes made. (Velloso, 2013)

More information on this dataset can be found at the following webpage: <http://groupware.les.inf.puc-rio.br/har#dataset#ixzz5fKXfT5aO>

Exploratory Analysis

We begin by reading the training and testing data into R and observe that the training dataset consists of 19,622 observations of 160 variables. These belong to six subjects, as per the `user_name` variable, producing five different movements (A, B, C, D or E), as per the `classe` variable. This type of movement is the variable that we shall seek to predict with our model. Below is a summary of the first 20 variables.

```
training<- read.csv("pml-training.csv")
testing<- read.csv("pml-testing.csv")
summary(training[, 1:20])
```

```
##           X           user_name  raw_timestamp_part_1 raw_timestamp_part_2
## Min.      :    1    adelmo   :3892    Min.      :1.322e+09    Min.      :   294
## 1st Qu.: 4906    carlitos :3112    1st Qu.:1.323e+09    1st Qu.:252912
## Median : 9812    charles  :3536    Median :1.323e+09    Median :496380
## Mean    : 9812    eurico   :3070    Mean   :1.323e+09    Mean   :500656
## 3rd Qu.:14717    jeremy   :3402    3rd Qu.:1.323e+09    3rd Qu.:751891
## Max.    :19622    pedro    :2610    Max.    :1.323e+09    Max.    :998801
##
##           cvtd_timestamp  new_window  num_window  roll_belt
## 28/11/2011 14:14: 1498    no :19216    Min.      : 1.0    Min.      : -28.90
## 05/12/2011 11:24: 1497    yes:  406    1st Qu.:222.0    1st Qu.:  1.10
## 30/11/2011 17:11: 1440                      Median :424.0    Median :113.00
## 05/12/2011 11:25: 1425                      Mean   :430.6    Mean   : 64.41
## 02/12/2011 14:57: 1380                      3rd Qu.:644.0    3rd Qu.:123.00
## 02/12/2011 13:34: 1375                      Max.    :864.0    Max.    :162.00
## (Other)           :11007
##           pitch_belt           yaw_belt  total_accel_belt kurtosis_roll_belt
## Min.      : -55.8000    Min.      : -180.00    Min.      : 0.00           :19216
## 1st Qu.:  1.7600    1st Qu.:  -88.30    1st Qu.:  3.00    #DIV/0! :   10
## Median :  5.2800    Median :  -13.00    Median :17.00    -1.908453:    2
## Mean     :  0.3053    Mean     : -11.21    Mean     :11.31    -0.016850:    1
```

```
## 3rd Qu.: 14.9000 3rd Qu.: 12.90 3rd Qu.:18.00 -0.021024: 1
## Max. : 60.3000 Max. : 179.00 Max. :29.00 -0.025513: 1
## (Other) : 391
## kurtosis_picth_belt kurtosis_yaw_belt skewness_roll_belt
## :19216 :19216 :19216
## #DIV/0! : 32 #DIV/0!: 406 #DIV/0! : 9
## 47.000000: 4 0.000000 : 4
## -0.150950: 3 0.422463 : 2
## -0.684748: 3 -0.003095: 1
## -1.750749: 3 -0.010002: 1
## (Other) : 361 (Other) : 389
## skewness_roll_belt.1 skewness_yaw_belt max_roll_belt max_picth_belt
## :19216 :19216 Min. : -94.300 Min. : 3.00
## #DIV/0! : 32 #DIV/0!: 406 1st Qu.: -88.000 1st Qu.: 5.00
## 0.000000 : 4 Median : -5.100 Median :18.00
## -2.156553: 3 Mean : -6.667 Mean :12.92
## -3.072669: 3 3rd Qu.: 18.500 3rd Qu.:19.00
## -6.324555: 3 Max. :180.000 Max. :30.00
## (Other) : 361 NA's :19216 NA's :19216
## max_yaw_belt
## :19216
## -1.1 : 30
## -1.4 : 29
## -1.2 : 26
## -0.9 : 24
## -1.3 : 22
## (Other): 275
```

Given the large size of the training set, we will create a validation set and then proceed to the exploratory analysis on our resulting training set.

```
set.seed(411)
inTrain<- createDataPartition(y = training$classe, p = 0.8, list = FALSE)
trainingset<- training[inTrain,]; validationset<- training[-inTrain,]
```

We notice that there are several NA values and proceed to count them in order to identify which variables are useful for prediction. We observe that 67 variables have a missing value rate of 97.9%, indicating that these may not be so useful for prediction. Moreover, we observe that in the cases where these variables have a value of "NA", 34 other variables have a blank value. From here on, we will only focus on the remaining variables as there is not information on the previously mentioned variables to use these for prediction.

```
NAvalues<-NULL
for (i in 1:160) {
  if (mean(is.na(trainingset[,i])) != 0) {
    NAvalues[i]<-i
  } else {NAvalues[i]<-0}
}

Blankvalues<-NULL
for (i in 1:160) {
  if (class(trainingset[1, i]) == "factor" & trainingset[1, i] == "") {
    Blankvalues[i]<-i
  } else {
    Blankvalues[i]<-0
  }
}
```

```

}
combined<-c(NAvalues, Blankvalues)
combined<-combined[combined != 0]
newtrainingset<- trainingset[, -combined]

```

We check that we have maintained variables with enough variability with the following zero covariate analysis.

```

nsv<-nearZeroVar(newtrainingset, saveMetrics = TRUE)
nsv

```

##		freqRatio	percentUnique	zeroVar	nzv
## X		1.000000	100.00000000	FALSE	FALSE
## user_name		1.117773	0.03821899	FALSE	FALSE
## raw_timestamp_part_1		1.032258	5.33154978	FALSE	FALSE
## raw_timestamp_part_2		1.250000	88.17122110	FALSE	FALSE
## cvtd_timestamp		1.002498	0.12739665	FALSE	FALSE
## new_window		47.453704	0.01273966	FALSE	TRUE
## num_window		1.032258	5.45257660	FALSE	FALSE
## roll_belt		1.086957	7.50366265	FALSE	FALSE
## pitch_belt		1.006452	11.06439901	FALSE	FALSE
## yaw_belt		1.124365	11.79692974	FALSE	FALSE
## total_accel_belt		1.079528	0.18472514	FALSE	FALSE
## gyros_belt_x		1.061510	0.80896872	FALSE	FALSE
## gyros_belt_y		1.166108	0.43314861	FALSE	FALSE
## gyros_belt_z		1.059901	1.07013186	FALSE	FALSE
## accel_belt_x		1.078947	1.03828269	FALSE	FALSE
## accel_belt_y		1.146040	0.89814638	FALSE	FALSE
## accel_belt_z		1.085631	1.89184024	FALSE	FALSE
## magnet_belt_x		1.112281	1.94916874	FALSE	FALSE
## magnet_belt_y		1.092157	1.88547041	FALSE	FALSE
## magnet_belt_z		1.002646	2.84094528	FALSE	FALSE
## roll_arm		52.423077	15.81629403	FALSE	FALSE
## pitch_arm		90.900000	18.52984267	FALSE	FALSE
## yaw_arm		34.948718	17.28135550	FALSE	FALSE
## total_accel_arm		1.016598	0.42040894	FALSE	FALSE
## gyros_arm_x		1.053922	4.05758329	FALSE	FALSE
## gyros_arm_y		1.504902	2.35683802	FALSE	FALSE
## gyros_arm_z		1.140811	1.51602013	FALSE	FALSE
## accel_arm_x		1.114504	4.91751067	FALSE	FALSE
## accel_arm_y		1.142857	3.37601121	FALSE	FALSE
## accel_arm_z		1.138614	4.90477100	FALSE	FALSE
## magnet_arm_x		1.044776	8.48461685	FALSE	FALSE
## magnet_arm_y		1.071429	5.49716542	FALSE	FALSE
## magnet_arm_z		1.094118	8.01961908	FALSE	FALSE
## roll_dumbbell		1.026316	85.57869928	FALSE	FALSE
## pitch_dumbbell		2.153846	83.41932607	FALSE	FALSE
## yaw_dumbbell		1.271739	85.08185235	FALSE	FALSE
## total_accel_dumbbell		1.079890	0.27390280	FALSE	FALSE
## gyros_dumbbell_x		1.010040	1.49054080	FALSE	FALSE
## gyros_dumbbell_y		1.266667	1.73259443	FALSE	FALSE
## gyros_dumbbell_z		1.042945	1.26122683	FALSE	FALSE
## accel_dumbbell_x		1.014440	2.66258997	FALSE	FALSE
## accel_dumbbell_y		1.019802	2.93012294	FALSE	FALSE
## accel_dumbbell_z		1.169399	2.57341232	FALSE	FALSE
## magnet_dumbbell_x		1.013889	6.96222689	FALSE	FALSE

```
## magnet_dumbbell_y      1.186207      5.33791961      FALSE FALSE
## magnet_dumbbell_z      1.013514      4.25504809      FALSE FALSE
## roll_forearm           10.881944     12.72692528      FALSE FALSE
## pitch_forearm          65.270833     17.42786165      FALSE FALSE
## yaw_forearm            15.660000     11.82240907      FALSE FALSE
## total_accel_forearm    1.135406      0.43951844      FALSE FALSE
## gyros_forearm_x        1.062350      1.85362125      FALSE FALSE
## gyros_forearm_y        1.033003      4.61812854      FALSE FALSE
## gyros_forearm_z        1.181579      1.84725142      FALSE FALSE
## accel_forearm_x        1.082192      4.97483916      FALSE FALSE
## accel_forearm_y        1.023529      6.25517549      FALSE FALSE
## accel_forearm_z        1.098361      3.61169501      FALSE FALSE
## magnet_forearm_x       1.033898      9.44646156      FALSE FALSE
## magnet_forearm_y       1.188406     11.75234091      FALSE FALSE
## magnet_forearm_z       1.020408     10.47837442      FALSE FALSE
## classe                 1.469388      0.03184916      FALSE FALSE
```

Given the number of variables, we calculate a correlation matrix on the numeric variables. We observe that several variables have a correlation higher than 0.80.

```
correlations<- abs(cor(newtrainingset[, -c(1, 2 ,3 , 4 ,5 , 6, 60)]))
diag(correlations)<-0
correlations<- as.data.frame(correlations)
subcor<-correlations[correlations > 0.8]
subcor
```

```
## [1] 0.8163596 0.9808708 0.9237196 0.9919226 0.9657924 0.8875411 0.8163596
## [8] 0.9808708 0.9270899 0.9745665 0.9657924 0.8919023 0.9237196 0.9270899
## [15] 0.9322608 0.9919226 0.9745665 0.9322608 0.8875411 0.8919023 0.9177242
## [22] 0.9177242 0.8160718 0.8160718 0.8140338 0.8140338 0.8098581 0.8479521
## [29] 0.9831733 0.9308973 0.9831733 0.9456068 0.8098581 0.8479521 0.8636651
## [36] 0.9308973 0.9456068 0.8636651
```

Model selection

Since our objective is to predict a factor variable with 5 levels, we will focus on non-linear models. We will start with a classification tree to get an idea of accuracy metrics. We note that this first model (Model 1) has 15,699 nodes and a relatively low accuracy of 66%. As detailed in the confusion matrix, the model correctly classifies A, B and E classes, but incorrectly classifies as E all those observations from the C and D classes.

```
set.seed(411)
modell1<- train(classe ~ . , data = newtrainingset, method = "rpart")
print(modell1$finalModel)

## n= 15699
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 15699 11235 A (0.28 0.19 0.17 0.16 0.18)
##   2) X< 5580.5 4464      0 A (1 0 0 0 0) *
##   3) X>=5580.5 11235  8197 B (0 0.27 0.24 0.23 0.26)
##     6) X< 9377.5 3038      0 B (0 1 0 0 0) *
##     7) X>=9377.5 8197  5311 E (0 0 0.33 0.31 0.35) *

modell1predictions<- predict(modell1, newdata = validationset)
confusionMatrix(validationset$classe, modell1predictions)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##           A 1116    0    0    0    0
##           B    0  759    0    0    0
##           C    0    0    0    0  684
##           D    0    0    0    0  643
##           E    0    0    0    0  721
##
```

```
## Overall Statistics
##
##           Accuracy : 0.6617
##           95% CI : (0.6467, 0.6765)
##           No Information Rate : 0.522
##           P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##           Kappa : 0.5695
## Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
```

```
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      1.0000  1.0000      NA      NA  0.3521
## Specificity      1.0000  1.0000  0.8256  0.8361  1.0000
## Pos Pred Value   1.0000  1.0000      NA      NA  1.0000
## Neg Pred Value   1.0000  1.0000      NA      NA  0.5856
## Prevalence       0.2845  0.1935  0.0000  0.0000  0.5220
## Detection Rate   0.2845  0.1935  0.0000  0.0000  0.1838
## Detection Prevalence 0.2845  0.1935  0.1744  0.1639  0.1838
## Balanced Accuracy 1.0000  1.0000      NA      NA  0.6760
```

We now fit a second model, a random forest model, to compare. We note that there is an important trade-off here in terms of accuracy and speed compared to our first model. Accuracy for Model 2 increases to 99.9%, although it is much more computationally demanding and subject to overfitting. We note that in this section we are using cross validation as the resampling method in the trainControl function, and changing to 5 the number that specifies the quantity of folds for k-fold cross validation.

```
cluster<-makeCluster(detectCores() - 1)
registerDoParallel(cluster)
fitControl<- trainControl(method = "cv", number = 5, allowParallel = TRUE)
model2<- train(classe ~ . , data = newtrainingset[, 2:60], method = "rf", trControl = fitControl)
stopCluster(cluster)
registerDoSEQ()

print(model2$finalModel)
```

```
##
## Call:
## randomForest(x = x, y = y, mtry = param$mtry)
##           Type of random forest: classification
##           Number of trees: 500
## No. of variables tried at each split: 41
##
```

```
##          OOB estimate of  error rate: 0.05%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4464    0    0    0    0 0.0000000000
## B    1 3036    1    0    0 0.0006583278
## C    0    3 2735    0    0 0.0010956903
## D    0    0    1 2571    1 0.0007773028
## E    0    0    0    1 2885 0.0003465003

model2predictions<- predict(model2, newdata = validationset)
confusionMatrix(validationset$classe, model2predictions)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    A    B    C    D    E
##              A 1116    0    0    0    0
##              B    1  758    0    0    0
##              C    0    0  684    0    0
##              D    0    0    0  643    0
##              E    0    0    0    1  720
##
## Overall Statistics
##
##              Accuracy : 0.9995
##              95% CI : (0.9982, 0.9999)
##      No Information Rate : 0.2847
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.9994
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.9991  1.0000  1.0000  0.9984  1.0000
## Specificity          1.0000  0.9997  1.0000  1.0000  0.9997
## Pos Pred Value       1.0000  0.9987  1.0000  1.0000  0.9986
## Neg Pred Value       0.9996  1.0000  1.0000  0.9997  1.0000
## Prevalence           0.2847  0.1932  0.1744  0.1642  0.1835
## Detection Rate       0.2845  0.1932  0.1744  0.1639  0.1835
## Detection Prevalence 0.2845  0.1935  0.1744  0.1639  0.1838
## Balanced Accuracy     0.9996  0.9998  1.0000  0.9992  0.9998
```

For purely visual purposes, we now graph one of the trees from our model to get an idea of how the variables are interacting. We have chosen to plot tree k= 1. Note that the code for this graph, which is purely to illustrate our model, has been sourced from Shirin's playgRound and can be found at this webpage: https://shiring.github.io/machine_learning/2017/03/16/rf_plot_ggraph. Please find full reference in the Sources section.

```
tree_func(final_model = model2$finalModel, 1)
```

[illegible]

Following an exploratory analysis and model selection process, we have fitted a random forest model with strong accuracy metrics on our validation data set which classifies the quality of a particular activity using 60 variables. This model will be used on a testing set to predict the class or quality of movement of 20

different observations. Although our in sample error is relatively low, we know that the out of sample error or generalization error will be slightly higher, particularly due to overfitting in random forest models. However, we expect an adequate performance.

Sources

Glander, Shirin. Plotting trees from Random Forest models with ggraph. Shirin's playgRound. 2019. URL: https://shiring.github.io/machine_learning/2017/03/16/rf_plot_ggraph

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.