# Prediction models for barbell lift exercise performance

*T.Roelofs*

*28 May 2017*

## Introduction

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. The data from the Weight Lifting Exercises Dataset from Velloso et al. has been used to model and predict the way exercises are performed. Class A corresponds to a correct execution of the exercise, class B-E point towards an incorrect execution of the exercise.

## Synopsis

The accuracy of the developed Graded Boosting Regression Model tp predict the correct class of activity (A-E) as determined on the independent validation set is 0.9621, with a 95% confidence interval of (0.9569, 0.9668). The Out of Sample error is therefore estimated at 3.8% (+/- 0.8%). This model was selected as a Random Forest model fit tuned out to be computationally too intensive for the computer infrastructure at hand.

## Data Processing

### Data loading

```
knitr::opts_chunk$set(verbose = FALSE, message = FALSE, warning = FALSE)

library(ggplot2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

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

```
setwd("~/Datasciencecoursera/Module 8 Practical Machine Learning/Week 4 assignment")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "traini
training <- read.csv("trainingdata")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "testdat
testing <- read.csv("testdata")
```

## Data preprocessing

Now we check the quality of the training data set. On casual manual inspection, it was visible that a considerable number of data fields was either NA, #DIV/0! or empty. The columns that contain too much of these odd values need to be removed to optimize the model fit. So let's check the values and do some more preprocessing by removing the covariates of the remaining set that correlate. The first 7 columns seem to be metadata on the measurements (person, timestamps etc) which we will leave out.

```
check <- data.frame(nrow=1, ncol=160)
quality <- data.frame(nrow=1, ncol=160)
for (i in 1:160){
  check[i] <- sum(is.na(training[,i])|
                      training[,i]=="#DIV/0!" |
                      training[,i]=="")
  quality[i] <- (1-(check[i])/length(training[,i]))
  }
# Only keep the covariates that have no more than 2% inadequate values
preproc_training <- training[,quality > 0.98]
# Remove first 7 columns
preproc_training <- preproc_training[,-(1:7)]
```

```
nearZeroVariates <- nearZeroVar(preproc_training, saveMetrics = TRUE)
print(nearZeroVariates)
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.101904     6.7781062   FALSE FALSE
## pitch_belt            1.036082     9.3772296   FALSE FALSE
## yaw_belt              1.058480     9.9734991   FALSE FALSE
## total_accel_belt      1.063160     0.1477933   FALSE FALSE
## gyros_belt_x          1.058651     0.7134849   FALSE FALSE
## gyros_belt_y          1.144000     0.3516461   FALSE FALSE
## gyros_belt_z          1.066214     0.8612782   FALSE FALSE
## accel_belt_x          1.055412     0.8357966   FALSE FALSE
## accel_belt_y          1.113725     0.7287738   FALSE FALSE
## accel_belt_z          1.078767     1.5237998   FALSE FALSE
## magnet_belt_x         1.090141     1.6664968   FALSE FALSE
## magnet_belt_y         1.099688     1.5187035   FALSE FALSE
## magnet_belt_z         1.006369     2.3290184   FALSE FALSE
## roll_arm             52.338462    13.5256345   FALSE FALSE
## pitch_arm            87.256410    15.7323412   FALSE FALSE
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## gyros_arm_x           1.015504     3.2769341   FALSE FALSE
## gyros_arm_y           1.454369     1.9162165   FALSE FALSE
## gyros_arm_z           1.110687     1.2638875   FALSE FALSE
## accel_arm_x           1.017341     3.9598410   FALSE FALSE
## accel_arm_y           1.140187     2.7367241   FALSE FALSE
## accel_arm_z           1.128000     4.0362858   FALSE FALSE
## magnet_arm_x          1.000000     6.8239731   FALSE FALSE
## magnet_arm_y          1.056818     4.4439914   FALSE FALSE
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
```

```
## gyros_dumbbell_x      1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z      1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x     1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y     1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y       1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

It turns out that no covariates are (highly) correlated with another covariate. The preprocessed training set therefore stays as it is. We can do the model fit tp predict the variable Classe with the 52 identified covariates.

We can check for an even distribution of the classes over the training set, a skewed distribution over the classes A-E may be a problem while fitting. We therefore do this final check:

```
table(preproc_training$classe)
```

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

Conclusion of the check is that there seems to be a quite healthy spread in measurements. A fair number of correct exercises (Classe A) and an about equal number of measurements over Classes B-E. We can proceed.

### Splitting into a training, test and validation set

To get an estimate of the Out of Sample error we need to split the training set into a training set and a validation set. My choice is a 70-30 split of the training set in a set for training and validation, while using random sampling without replacement. In the training set 20% of the total number of data points is used for testing.

```
set.seed(5555)
inTrain <- createDataPartition(y = preproc_training$classe, p = .7, list = FALSE)
train_and_test_set <- preproc_training[inTrain,]
val_set <- preproc_training[-inTrain,]

index_split_train_test <- createDataPartition(train_and_test_set$classe, p = 0.7143, list = FALSE)
train_set <- train_and_test_set[index_split_train_test,]
test_set <- train_and_test_set[-index_split_train_test,]
```

As described by the authors in http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf a prediction accuracy of 98% is 'up to par', so this is the aim for the first model fit.

# Model Fit with Linear Discriminant Analysis and Generalized Boosted Regression

At first, I tried a random forest tree fit on the data, but this was computationlly apparently too intense for my personal computer. The calculation did not finish in 18 hours (with several runs that were tried). Therefore, I needed to see in how far less computationally intensive methods could do the job. I chose to see what the results with a Linear Discriminant Analysis (LDA) method, a very simple approach with high speed, and a Generalized Boosted Regression Methods (GBM) method would be. The lattera more computation but in the literature reported to approach the random forest method.

```
model_fit_lda <- train(classe ~ . , data= train_set, method="lda")
model_fit_lda
```

```
## Linear Discriminant Analysis
##
## 9815 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 9815, 9815, 9815, 9815, 9815, 9815, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6964921  0.6158006
```

```
knitr::opts_chunk$set(verbose = FALSE, message = FALSE, warning = FALSE)
model_fit_gbm <- train(classe ~ . , data= train_set, method="gbm", verbose=FALSE)
model_fit_gbm
```

```
## Stochastic Gradient Boosting
##
## 9815 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 9815, 9815, 9815, 9815, 9815, 9815, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7491059  0.6817824
##   1                  100      0.8117897  0.7616844
##   1                  150      0.8450540  0.8038690
##   2                   50      0.8454709  0.8042224
##   2                  100      0.9000695  0.8735030
##   2                  150      0.9245817  0.9045509
##   3                   50      0.8897957  0.8604451
```

```
##   3                  100      0.9345572  0.9171859
##   3                  150      0.9527136  0.9401739
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Based on the training set results, the LDA accuracy is limited, around 0.7. The GMB scores considerably better, but took 30 minutes to process on the personal computer that I used. I then tried whether combining the prediction models would provide for an improvement in accuracy. To compute a combined model a Random Forest was chosen because of the maximum accuracy in the combination. The RF computation took around 45 minutes to compute, which was acceptable for this purpose.

# Combining the models

```
prediction_lda <- predict(model_fit_lda, test_set)
prediction_gbm <- predict(model_fit_gbm, test_set)
prediction_DF <- data.frame(prediction_lda, prediction_gbm, classe = test_set$classe)
combined_model <- train(classe ~. , method="rf", data=prediction_DF)
combined_predictions <- predict(combined_model, prediction_DF)
confusionMatrix(combined_predictions, prediction_DF$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1088   22    0    0    1
##          B   12  719   17    1    5
##          C    4   16  654   17    6
##          D    3    0   13  620    4
##          E    8    2    0    5  705
##
## Overall Statistics
##
##                Accuracy : 0.9653
##                  95% CI : (0.9591, 0.9708)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9561
##  Mcnemar's Test P-Value : 0.006733
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9758   0.9473   0.9561   0.9642   0.9778
## Specificity            0.9918   0.9889   0.9867   0.9939   0.9953
## Pos Pred Value         0.9793   0.9536   0.9383   0.9687   0.9792
## Neg Pred Value         0.9904   0.9874   0.9907   0.9930   0.9950
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
```

```
## Detection Rate         0.2774   0.1833   0.1668   0.1581   0.1798
## Detection Prevalence   0.2833   0.1922   0.1777   0.1632   0.1836
## Balanced Accuracy      0.9838   0.9681   0.9714   0.9791   0.9866
```

The conclusion on the Confusion Matrix for the combined model is that the gain in accuracy is only marginal, certainly for the heavy processing that the Random Forest on the combined model caused. I therefore continue with validating on basis of the the GBM model. The approach with combining models with RF would not be scalable anyway.

# Use the model to generate a GBM prediction for the Validation set

```
predValgbm <- predict(model_fit_gbm, newdata = val_set)
confusionMatrix(predValgbm, val_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1646   32    0    2    0
##          B   19 1079   31    0   17
##          C    4   24  977   35    9
##          D    5    3   17  920   16
##          E    0    1    1    7 1040
##
## Overall Statistics
##
##                Accuracy : 0.9621
##                  95% CI : (0.9569, 0.9668)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9521
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9833   0.9473   0.9522   0.9544   0.9612
## Specificity            0.9919   0.9859   0.9852   0.9917   0.9981
## Pos Pred Value         0.9798   0.9415   0.9314   0.9573   0.9914
## Neg Pred Value         0.9933   0.9873   0.9899   0.9911   0.9913
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2797   0.1833   0.1660   0.1563   0.1767
## Detection Prevalence   0.2855   0.1947   0.1782   0.1633   0.1782
## Balanced Accuracy      0.9876   0.9666   0.9687   0.9730   0.9797
```

# Conclusion

The accuracy of the Graded Boosting Regression Model as determined on the independent validation set is 0.9621, with a 95% confidence interval of (0.9569, 0.9668). The Out of Sample error is therefore 3.8%.

The GBM model scores relatively well in accuracy compared to the Random Forest accuracy (98%) as reported by the researchers Velloso, E. et al. in the paper 'Qualitative Activity Recognition of Weight Lifting Exercises', Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

The judgement 'relatively well' is based on the huge performance difference where an RF computation did not finish in 18 hours, whereas the GBM ran around 30 minutes.

The combination of models, e.g with a simple LDA model, could be leading to improvements in accuracy. The LDA modelling (which is fast) is too inaccurate to really boost accuracy. Other maybe computationally more intense fitting tree models could be considered that have a better trade-off in computation speed and accuracy after combining predictors.