# Machine Learning Course Project

*D. Spence*

# Overview

We are given a data set with human activity measurements taken from several weight lifting exercises, performed in one of five ways (labeled A-E). Only one of the ways is correct (method A). The other ways of performing the exercise reflect common mistakes made during weight lifting. Our goal is to generate a model from the measurements that correctly predicts which way the exercise was performed. The “Weight Lifting Exercises” dataset for this analysis was made available by the authors named below and was part of their study in the work cited:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. *Qualitative Activity Recognition of Weight Lifting Exercises.* Proceedings of the 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13). Stuttgart, Germany: ACM SIGCHI, 2013.

# Loading and Preprocessing Data

First we load essential libraries for the task. Then we read in the data file that contains the training data.

```
library(caret)
library(randomForest)
wle <- read.csv("pml-training.csv", na.strings = c("","NA"))
```

The data set contains 160 variables, but many of these are mostly blank or NA. This is the reason we stipulated both empty string and NA as representing missing data when reading in the file. We remove variables for which values are missing over half the time. Since the data set has 19622 observations, I have set the threshold for missing values at 10,000.

```
#Remove identifier and timestamp variables; these shouldn't be predictors
wle <- wle[,-c(1:5)]

#Also remove columns that are over half empty/NA
countNA <- function(column) sum(is.na(column))
badcols <- (sapply(wle, countNA) > 10000)
badcolnames <- colnames(wle)[badcols]

wle <- wle[,!badcols]
```

*We are removing these columns from the data set:*

```
##   [1] "kurtosis_roll_belt"       "kurtosis_picth_belt"
##   [3] "kurtosis_yaw_belt"        "skewness_roll_belt"
##   [5] "skewness_roll_belt.1"     "skewness_yaw_belt"
##   [7] "max_roll_belt"            "max_picth_belt"
##   [9] "max_yaw_belt"             "min_roll_belt"
##  [11] "min_pitch_belt"           "min_yaw_belt"
##  [13] "amplitude_roll_belt"      "amplitude_pitch_belt"
##  [15] "amplitude_yaw_belt"       "var_total_accel_belt"
##  [17] "avg_roll_belt"            "stddev_roll_belt"
##  [19] "var_roll_belt"            "avg_pitch_belt"
##  [21] "stddev_pitch_belt"        "var_pitch_belt"
##  [23] "avg_yaw_belt"             "stddev_yaw_belt"
##  [25] "var_yaw_belt"             "var_accel_arm"
##  [27] "avg_roll_arm"             "stddev_roll_arm"
##  [29] "var_roll_arm"             "avg_pitch_arm"
##  [31] "stddev_pitch_arm"         "var_pitch_arm"
##  [33] "avg_yaw_arm"              "stddev_yaw_arm"
##  [35] "var_yaw_arm"              "kurtosis_roll_arm"
##  [37] "kurtosis_picth_arm"       "kurtosis_yaw_arm"
##  [39] "skewness_roll_arm"        "skewness_pitch_arm"
##  [41] "skewness_yaw_arm"         "max_roll_arm"
##  [43] "max_picth_arm"            "max_yaw_arm"
##  [45] "min_roll_arm"             "min_pitch_arm"
##  [47] "min_yaw_arm"              "amplitude_roll_arm"
##  [49] "amplitude_pitch_arm"      "amplitude_yaw_arm"
##  [51] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [53] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [55] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [57] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [59] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [61] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [63] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
##  [65] "amplitude_yaw_dumbbell"   "var_accel_dumbbell"
##  [67] "avg_roll_dumbbell"        "stddev_roll_dumbbell"
##  [69] "var_roll_dumbbell"        "avg_pitch_dumbbell"
##  [71] "stddev_pitch_dumbbell"    "var_pitch_dumbbell"
##  [73] "avg_yaw_dumbbell"         "stddev_yaw_dumbbell"
##  [75] "var_yaw_dumbbell"         "kurtosis_roll_forearm"
##  [77] "kurtosis_picth_forearm"   "kurtosis_yaw_forearm"
##  [79] "skewness_roll_forearm"    "skewness_pitch_forearm"
##  [81] "skewness_yaw_forearm"     "max_roll_forearm"
##  [83] "max_picth_forearm"        "max_yaw_forearm"
##  [85] "min_roll_forearm"         "min_pitch_forearm"
##  [87] "min_yaw_forearm"          "amplitude_roll_forearm"
##  [89] "amplitude_pitch_forearm"  "amplitude_yaw_forearm"
##  [91] "var_accel_forearm"        "avg_roll_forearm"
##  [93] "stddev_roll_forearm"      "var_roll_forearm"
##  [95] "avg_pitch_forearm"        "stddev_pitch_forearm"
##  [97] "var_pitch_forearm"        "avg_yaw_forearm"
##  [99] "stddev_yaw_forearm"       "var_yaw_forearm"
```

This process leaves 55 columns in the data set, one of which is the outcome variable. The other 54 are potential predictors.

# Method for Creating Prediction Model

The outcome variable, classe, is a categorical variable. We will create the model using a random forest with functions available in the caret package. These functions also allow us to incorporate cross-validation into the training process. The machine on which these analyses are being run is limited in its capacity, so even though the caret package functions would theoretically allow us to accomplish the training with cross-validation on a large section of the training data at once, the machine could not handle the load. Therefore, the following phased approach was employed instead.

## 1. Set training options for 3-fold cross-validation

*All training to follow will use the random forest method with these training options.*

```
trainOpts <- trainControl()
trainOpts$method <- "cv"
trainOpts$number <- 3
```

## 2. Train initially with 10% of the training data using all 54 predictors

```
set.seed(44)

inTrain <- createDataPartition(y=wle$classe, p=0.1, list=FALSE)
trainingInit <- wle[inTrain,]
trainingRest <- wle[-inTrain,]
```

```
modFitInit <- train(classe ~ ., data = trainingInit, method = "rf", trControl=trainOpts, prox=TR
UE)
```

```
print(modFitInit)
```

```
## Random Forest
##
## 1964 samples
##   54 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 1309, 1308, 1311
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9399225  0.9239676
##   28    0.9526451  0.9400525
##   54    0.9501092  0.9368676
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 28.
```

# 3. Narrow down number of predictors in data set

The initial model achieved the most accuracy with 28 variables per step. We determine which 28 variables were found to be most important (based on Gini index, the average decrease in impurity after splitting on that variable). We then pare down the remaining data to train with only those 28 predictors. This allows us to run the random forest with a larger set of data without locking up the PC.

```
priority <- importance(modFitInit$finalModel, sort=TRUE)
key <- order(priority, decreasing=TRUE)
priority <- priority[key,, drop=FALSE]

#Keep top 28 predictors and outcome variable only
trainingRest <- trainingRest[,c(key[1:28],55)]
```

*These are the 28 predictor variables we are keeping:*

```
##                      MeanDecreaseGini
## num_window                  221.82114
## roll_belt                   159.71699
## pitch_forearm               114.81937
## magnet_dumbbell_z            85.91821
## magnet_dumbbell_y            85.69131
## yaw_belt                     65.63817
## pitch_belt                   59.32341
## roll_forearm                 49.16322
## roll_dumbbell                42.02146
## magnet_dumbbell_x            41.52198
## accel_dumbbell_y             34.59549
## accel_dumbbell_z             31.19551
## magnet_belt_z                30.77790
## magnet_belt_y                30.20246
## accel_belt_z                 28.93299
## accel_forearm_x              27.76092
## magnet_belt_x                24.27342
## total_accel_dumbbell         22.92248
## yaw_dumbbell                 22.68515
## gyros_dumbbell_y             20.59365
## magnet_forearm_z             20.11563
## magnet_arm_y                 18.36494
## accel_forearm_z              18.26859
## magnet_arm_x                 16.54996
## gyros_belt_z                 16.43942
## accel_dumbbell_x             14.66975
## pitch_dumbbell               13.41846
## magnet_forearm_x             12.87166
```

# 4. Generate main model

Using only the 28 predictors above, we train with 3-fold cross-validation a second time, using a new set of observations from the training data. The initial model was based on 10% of the observations in the data set; we are now generating a model with 30% of the **remaining** observations. I would like to have used more, but it was not feasible on the available computer. Therefore, the rest of the data set was saved to validate the model.

```
inTrain <- createDataPartition(y=trainingRest$classe, p=0.3, list=FALSE)
training <- trainingRest[inTrain,]
validation <- trainingRest[-inTrain,]
```

```
modFitMain <- train(classe ~ ., data = training, method = "rf", trControl=trainOpts)
```

**Here is the main model:**

```
print(modFitMain)
```

```
## Random Forest
##
## 5300 samples
##   28 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 3533, 3533, 3534
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9735847  0.9665661
##   15    0.9824528  0.9777977
##   28    0.9811320  0.9761271
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 15.
```

```
modFitMain$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: 15
##
##         OOB estimate of  error rate: 1.17%
## Confusion matrix:
##      A    B   C   D   E class.error
## A 1502    2   2   1   0  0.00331785
## B    8 1010   7   1   0  0.01559454
## C    0    5 913   5   1  0.01190476
## D    0    2  16 849   2  0.02301496
## E    0    3   0   7 964  0.01026694
```

This model has in-sample accuracy of 98%, even higher than the initial model.

## 5. Estimate out-of-sample accuracy with validation data

```
validPred <- predict(modFitMain,validation)
confusionMatrix(validPred,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3510   14    0    0    0
##          B    4 2361   17    0    4
##          C    0    9 2135    8    2
##          D    0    4    3 2014   12
##          E    1    3    0    3 2254
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9916, 0.9946)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9875   0.9907   0.9946   0.9921
## Specificity            0.9984   0.9975   0.9981   0.9982   0.9993
## Pos Pred Value         0.9960   0.9895   0.9912   0.9907   0.9969
## Neg Pred Value         0.9994   0.9970   0.9980   0.9989   0.9982
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1911   0.1728   0.1630   0.1824
## Detection Prevalence   0.2852   0.1931   0.1743   0.1645   0.1830
## Balanced Accuracy      0.9985   0.9925   0.9944   0.9964   0.9957
```

# Conclusion

In this particular case, the accuracy on the validation set (99%) was even higher than the in-sample accuracy (98%). This is unusual, since the model was fitted to the training data and not the validation data. Possibly, the two-phased training approach helped to avoid over-fitting. Nevertheless, this result suggests an estimated out-of-sample accuracy of 99% for this model, which is exceptionally high. This model should predict the user's weight-lifting style extremely well.