# Practical Machine Learning Assignment

*blockhead17*

## Executive Summary

The goal of this project is to use a prediction model that can determine if an exercise is being done correctly. The data for this project are available from a team of researchers through their website and are described in greater detail in a publication[1]. This report describes the process of model building using cross validation as well as exploratory data analyses. The characteristics of several models are presented with one model selected to use for predicting the exercise technique for 20 records in a test data set. Appendices show detailed code for the majority of this report.

## Data Processing

To begin, the raw data sets were imported (Appendix 1). After visually inspecting the data using `str(rawData)` and `finalTest`, I noted that there seemed to be a number of columns with missing data and that the test data set provided by the original researchers did not include the outcome variable of interest. In order to build the prediction model effectively, the data would need to be split into training and test data sets. I chose to use the random subsampling approach for cross validating, assigning 75% of the data to the `train` data frame while reserving the remainder for `test`. By splitting the data at this point, cross validation allows us to build models using data that are completely separate from those used to check their accuracy. This provides an optimistic estimate of accuracy, but protects us from over fitting the models. The 20 record test set provided by the researchers is further held back until the very end, being used as a validation data set (Appendix 2).

## Exploratory Data Analysis and Cleaning

After reading the available documentation of the original study and thinking about its design, I was initially disposed to choosing a set of variables for potential predictors that essentially distilled down to the raw (not calculated) values for each user, device and location. However, due diligence requires a more substantive look at the data to supplement an informed opinion. I first checked to see what columns had a meaningful percentage of missing data (30%) in case some imputation might be necessary (in fact this percentage could have been much higher and yielded the same number of columns). I then checked for near zero variance (NZV) columns and excluded columns from the data that met either the high missing or NZV criterion. All that remained after this trimming were the variables I originally proposed above, as well as the primary outcome and other identifier variables (e.g. user, time stamps). The identifier variables are assumed to be unimportant for the purpose of model prediction leaving a final training data set of 53 variables.

```
checkNA <- function (v) {
      percentNA=sum(is.na(v))/length(v)
      ifelse(percentNA>0.3,TRUE,FALSE)
}

w <- sapply(train, checkNA)
z <- nearZeroVar(train,saveMetrics = TRUE)

train2 <- train[,!w & !z$nzv]
```

```r
train3 <- train2[,c("classe",
                   grep("^gyros|^accel|^magnet|^roll|^pitch|^yaw|^total",
                   names(train), value=TRUE))]
```

After reaching this point, having reduced the number of potential predictors to about 1/3 of its original size, I explored some corrlations in the remaining data columns (Appendix 3) and principal components analysis (Appendix 4) but decided to not reduce the data further unless needed in subsequent analyses. Appendix 5 shows the minimum and maximum values for the remaining predictors. We can see that the variables are a bit diverse, with many of them having both positive and negative values and the orders of magnitude in the measurements were not too far from each other. Because this was an exercise in classifcation instead of regression, interpretability of coefficients was less of a concern, so I opted not to center and scale the values as part of the pre-processing step.

## Model Building

Three different models were fitted using the `train3` data (Appendix 6). The first used a multinomial log-linear model to predict a nominal (factor) response variable. The second used a regression tree approach and the third used a random forest approach. For the multinomial log-linear model, the reference group for comparisons was set to "A" (doing the exercise correctly). Otherwise, all three approaches used the same syntactical and analytic approach. Fit the model in question and save to an object, then use the predict function to apply that model to our test set. The results of this were again saved to an object that was used to generate a confusion matrix (always compared to `test$classe`). The most important predictors for the random forest model are showing in Appendix 7.

## Predicting Results on the Final Test Data

The accuracy for the multinomial log-linear model was 74.1% and the regression tree approach only yielded a small improvement (74.9%). The random forest approach, however, had an overall accuracy rate of 99.6%, making it the easy choice for use against the final validation set. The expected out of sample (generalization) error - generated from the test data set - is 0.4%.

```r
finalPredict <- predict(fitForest, finalTest, type = "class")
finalPredict
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

# Appendices

### About the Data

- The study from which these data are used pertains to "quantified self movement." In an effort to quantify how well the participants performed an exercise (barbell lifts), 6 participants accelerometers to quantify their movements while performing the exercise correctly as well as using 4 techniques that highlight typical mistakes in performing hte exercise (the *classe* variable; *classe=A* represents correct technique).
- Each participant had four sensors on his body to capture data: glove (forearm), armband (arm), lumbar belt (belt) and dumbbell.
- Spatial data (x, y and z measurements) were collected at each time point for the accelerometer, gyroscope and magnetometer.
- Also collected at each time point were roll, pitch and yaw to measure rotational data.
- For each sensor, the data set included summary (calculated) statisitics: mean, variance, standard deviation, max, min, amplitude, kurtosis and skewness.
- The data set does include time stamp information though these data are not treated as a time series.

### References

1. 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.

### Appendix 1: Initialization

```r
#Empty the console and environment
cat("\014")
rm(list=ls())

#Import raw data
setwd("~/Documents/R Working Directory/PracticalMachineLearningProject")
finalTest<-read.table("pml-testing.csv", sep = ",",header=TRUE)
rawData<-read.table("pml-training.csv", sep = ",",header=TRUE)

#Load the libraries to be used
library(dplyr)
library(caret)
library(nnet)
library(reshape2)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)

#Set the seed for reproducibility
set.seed(1)
```

### Appendix 2: Cross Validation

### Appendix 3: Correlation Analysis

```
M <- abs(cor(train3[,-1]))
diag(M) <- 0
M2 <- as.data.frame(which(M>0.8,arr.ind = TRUE,useNames = TRUE))
M2$row2 <- rownames(M2)
M2$col2 <- factor(colnames(train3[M2$col+1]))
M2$col2 <- gsub("\\.*", "", M2$col2)
M2 <- M2[3:4]
```

### Appendix 4: Principal Components

```
prComp <- prcomp(train3[,-1],retx=TRUE)
summary(prComp)
```

### Appendix 5: Ranges for Numeric Predictors in Final Training Set

```
minmax <- sapply(train3[,-1],range)
minmax <- t(minmax)
colnames(minmax) <- c("min","max")
minmax
```

```
##                            min      max
## roll_belt             -28.8000 162.0000
## pitch_belt            -55.8000  60.2000
## yaw_belt             -180.0000 179.0000
## total_accel_belt        0.0000  28.0000
## gyros_belt_x           -1.0400   2.2200
## gyros_belt_y           -0.6400   0.6400
## gyros_belt_z           -1.4600   1.6200
## accel_belt_x         -120.0000  85.0000
## accel_belt_y          -65.0000 164.0000
## accel_belt_z         -269.0000 104.0000
## magnet_belt_x         -52.0000 485.0000
## magnet_belt_y         359.0000 673.0000
## magnet_belt_z        -623.0000 289.0000
## roll_arm             -178.0000 180.0000
## pitch_arm             -87.9000  88.5000
## yaw_arm              -180.0000 180.0000
## total_accel_arm         1.0000  65.0000
## gyros_arm_x            -6.3700   4.8700
## gyros_arm_y            -3.4000   2.8400
## gyros_arm_z            -2.2800   3.0200
## accel_arm_x          -383.0000 435.0000
## accel_arm_y          -315.0000 308.0000
## accel_arm_z          -630.0000 292.0000
## magnet_arm_x         -580.0000 782.0000
## magnet_arm_y          -392.0000  582.0000
## magnet_arm_z          -596.0000  694.0000
## roll_dumbbell         -153.5097  153.3776
## pitch_dumbbell        -149.5936  149.4024
## yaw_dumbbell          -148.7655  154.9523
## total_accel_dumbbell     0.0000   58.0000
## gyros_dumbbell_x      -204.0000    2.2200
## gyros_dumbbell_y        -2.1000   52.0000
## gyros_dumbbell_z        -2.3800  317.0000
## accel_dumbbell_x      -419.0000  219.0000
## accel_dumbbell_y      -179.0000  310.0000
## accel_dumbbell_z      -334.0000  318.0000
## magnet_dumbbell_x     -643.0000  584.0000
## magnet_dumbbell_y    -3600.0000  633.0000
## magnet_dumbbell_z     -262.0000  452.0000
## roll_forearm          -180.0000  180.0000
## pitch_forearm          -72.5000   88.7000
## yaw_forearm           -180.0000  180.0000
## total_accel_forearm      0.0000  108.0000
## gyros_forearm_x        -22.0000    3.9700
## gyros_forearm_y         -7.0200  311.0000
## gyros_forearm_z         -8.0900  231.0000
## accel_forearm_x       -498.0000  477.0000
## accel_forearm_y       -632.0000  923.0000
## accel_forearm_z       -446.0000  291.0000
## magnet_forearm_x     -1280.0000  672.0000
## magnet_forearm_y      -896.0000 1480.0000
## magnet_forearm_z      -973.0000 1090.0000
```

### Appendix 6: Three Types of Models

### Multinomial log-linear model

```r
train3$classe <- relevel(train3$classe, ref = "A")
fitMult <- multinom(classe ~ ., data = train3,maxit = 1000)
```

```
## # weights:  270 (212 variable)
## initial  value 23687.707195
## iter  10 value 19068.548900
## iter  20 value 16687.455505
## iter  30 value 15669.685388
## iter  40 value 14820.139986
## iter  50 value 14305.940642
## iter  60 value 13873.796851
## iter  70 value 13648.054527
## iter  80 value 13521.834712
## iter  90 value 13428.431043
## iter 100 value 13336.347488
## iter 110 value 13296.913270
## iter 120 value 13267.517464
## iter 130 value 13236.867723
## iter 140 value 13207.632611
## iter 150 value 13178.175231
## iter 160 value 13107.808192
## iter 170 value 12977.407456
## iter 180 value 11448.958077
## iter 190 value 10844.810798
## iter 200 value 10420.914627
## iter 210 value 10212.951502
## iter 220 value 10150.162700
## iter 230 value 10135.197974
## iter 240 value 10135.064339
## iter 250 value 10135.061509
## final  value 10135.060043
## converged
```

```
# c <- summary(fitMult)$coefficients
# z <- summary(fitMult)$coefficients/summary(fitMult)$standard.errors
# p <- (1 - pnorm(abs(z), 0, 1)) * 2
#summary(fitMult)
predictMult <- predict(fitMult, test, type = "class")
cmMult <- confusionMatrix(predictMult, test$classe)
cmMult
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1203  122   91   43   45
##          B   43  631   71   45  110
##          C   69   89  616   94   52
##          D   62   26   45  573   68
##          E   18   81   32   49  626
##
## Overall Statistics
##
##                Accuracy : 0.7441
##                  95% CI : (0.7316, 0.7563)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6754
##  Mcnemar's Test P-Value : 1.714e-15
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8624   0.6649   0.7205   0.7127   0.6948
## Specificity            0.9142   0.9320   0.9249   0.9510   0.9550
## Pos Pred Value         0.7999   0.7011   0.6696   0.7403   0.7767
## Neg Pred Value         0.9435   0.9206   0.9400   0.9441   0.9329
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2453   0.1287   0.1256   0.1168   0.1277
## Detection Prevalence   0.3067   0.1835   0.1876   0.1578   0.1644
## Balanced Accuracy      0.8883   0.7984   0.8227   0.8318   0.8249
```

### Regression tree

```r
fitTree <- rpart(classe ~ ., data=train3, method="class")
#summary(fitTree)
##fancyRpartPlot(fitTree)
predictTree <- predict(fitTree, test, type = "class")
cmTree <- confusionMatrix(predictTree, test$classe)
cmTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1241  147    7   41   16
##          B   41  556   52   56   65
##          C   36  101  687  119  104
##          D   54   75   53  511   56
##          E   23   70   56   77  660
##
## Overall Statistics
##
##                Accuracy : 0.7453
##                  95% CI : (0.7329, 0.7575)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6776
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8896   0.5859   0.8035   0.6356   0.7325
## Specificity            0.9399   0.9459   0.9111   0.9420   0.9435
## Pos Pred Value         0.8547   0.7221   0.6562   0.6822   0.7449
## Neg Pred Value         0.9554   0.9049   0.9564   0.9295   0.9400
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2531   0.1134   0.1401   0.1042   0.1346
## Detection Prevalence   0.2961   0.1570   0.2135   0.1527   0.1807
## Balanced Accuracy      0.9147   0.7659   0.8573   0.7888   0.8380
```

### Random forest

```r
fitForest <- randomForest(classe ~., data=train3)
#summary(fitRF)
predictForest <- predict(fitForest, test, type = "class")
cmForest <- confusionMatrix(predictForest,test$classe)
cmForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    1    0    0
##          C    0    0  854    2    0
##          D    0    0    0  802    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9994
##                  95% CI : (0.9982, 0.9999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9992
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9988   0.9975   1.0000
## Specificity            1.0000   0.9997   0.9995   1.0000   1.0000
## Pos Pred Value         1.0000   0.9989   0.9977   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9998   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1741   0.1635   0.1837
## Detection Prevalence   0.2845   0.1937   0.1746   0.1635   0.1837
## Balanced Accuracy      1.0000   0.9999   0.9992   0.9988   1.0000
```

**Appendix 7: Most Important Variables from Random Forest**

```
impForest <- varImp(fitForest)
impForest <- cbind(Variable = rownames(impForest), impForest)
impForest <- arrange(impForest,desc(Overall))
head(impForest,20)
```

```
##               Variable  Overall
## 1            roll_belt 926.9865
## 2             yaw_belt 669.4127
## 3        pitch_forearm 588.3592
## 4     magnet_dumbbell_z 582.2289
## 5     magnet_dumbbell_y 517.2811
## 6           pitch_belt 510.2164
## 7         roll_forearm 447.0109
## 8     magnet_dumbbell_x 365.3823
## 9        roll_dumbbell 318.6215
## 10     accel_dumbbell_y 315.4362
## 11        magnet_belt_z 290.9501
## 12        magnet_belt_y 290.7846
## 13         accel_belt_z 284.6641
## 14     accel_dumbbell_z 251.8776
## 15         gyros_belt_z 246.3269
## 16             roll_arm 245.0926
## 17      accel_forearm_x 230.3974
## 18     magnet_forearm_z 219.4380
## 19 total_accel_dumbbell 210.7394
## 20         magnet_arm_x 195.4374
```