PRACTICAL MACHINE LEARNING COURSE PROJECT

PROJECT SUMMARY:

In this project, the goal is to use data from the accelerometers (Fitbit, Nike Fuelband, etc.) of six participants and predict the manner in which they did a dumbbell lifting exercise.

In the study, six people participated in a dumbell lifting exercise five different ways. The five ways, as described in the study, were “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). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.”

The variable we will be predicting on is “classe” and the data is split up between the five classes. By processing data gathered from accelerometers in a machine learning algorithm, can the appropriate activity quality (class A-E) be predicted?

Training and test data were provided from the following study: 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.

More information is available from this website: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Processing

Let’s begin by loading the data and R packages needed.

```
## Set working directory then load train and test data
train = read.csv("pml-training.csv", na.strings=c("NA",""), header=T)
test = read.csv("pml-testing.csv", na.strings=c("NA",""),header=T)
library(AppliedPredictiveModeling)
library(caret)
```

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

```
library(e1071)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Let’s look at the missing values in each column.

```
na_test = sapply(train, function(x) {sum(is.na(x))})
table(na_test)
```

```
## na_test
##     0 19216
##    60   100
```

100 columns of the 160 in the dataset have almost all missing values. We will remove these columns from our training data and then take a look at the remaining columns.

```
na_columns = names(na_test[na_test==19216])
train = train[, !names(train) %in% na_columns]
str(train)
```

```
## 'data.frame':    19622 obs. of  60 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 13
23084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440390 484323 48443
4 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

In order to look only at the variables related to movement, we will also remove the first seven columns of the dataset that have to do with the sequence and subject.

```
train = train[,-c(1:7)]
```

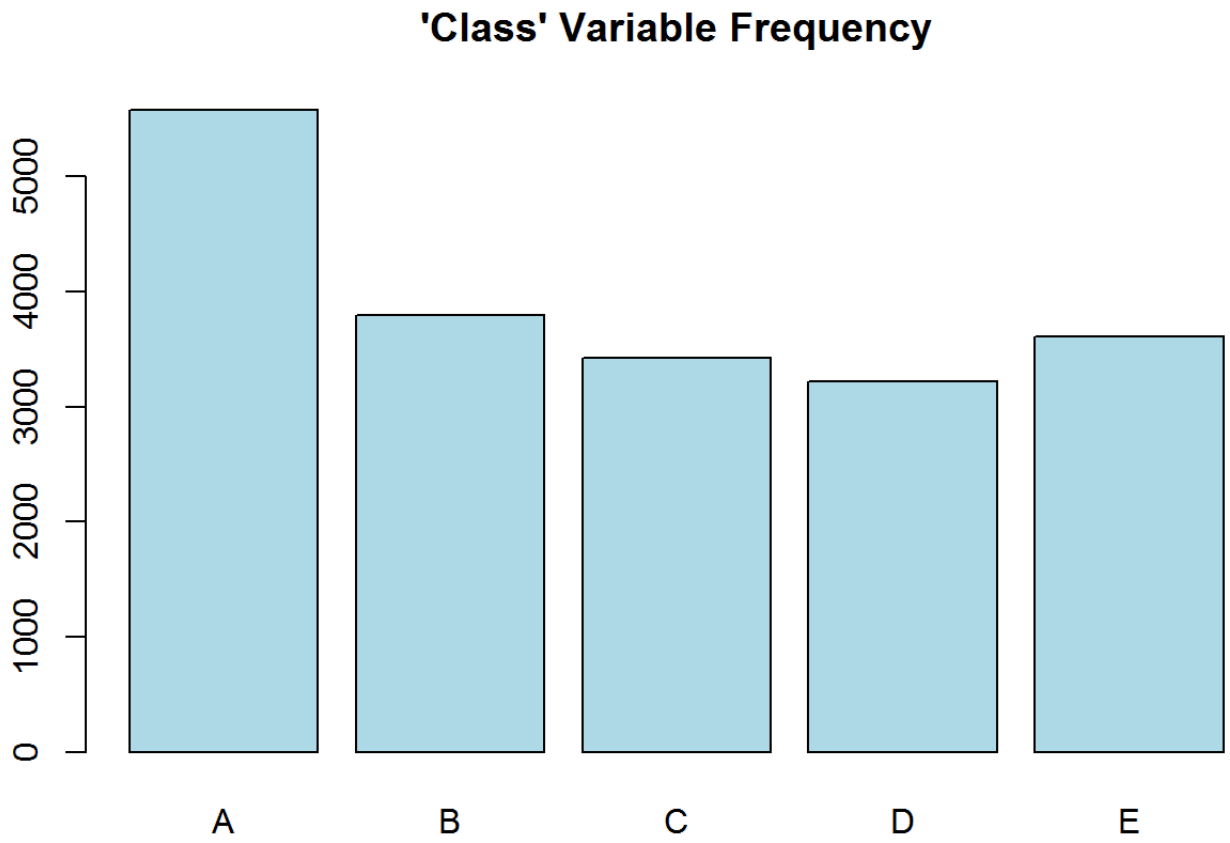
Exploratory Data Analysis

Let's take a look at the "classe" variable since that's what we will build our prediction model on.

```
summary(train$classe)
```

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

```
plot(train$classe, col="light blue", main = "'Class' Variable Frequency")
```



Build model

We will now create a model to predict the movement class using the random forest machine learning technique on the remaining variables in the Train dataset. We will build the model using 4-fold cross validation. Note: This model takes a while to run so you may want to save it once it's finished.

```
model = train(classe ~ ., data=train, method="rf", prox=T, trControl = trainControl(method="cv", numb
er=4, allowParallel=T))
saveRDS(model, "rfmodel.RDS")
model = readRDS("rfmodel.RDS")
model
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 14715, 14717, 14717, 14717
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    1         1      0.002        0.002
##   27    1         1      0.002        0.003
##   52    1         1      0.002        0.002
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

In Sample Accuracy

Next, we calculate the in-sample accuracy which is the prediction accuracy of our model when applied to the Train dataset.

```
train_pred = predict(model, train)
confusionMatrix(train_pred, train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.194    0.174    0.164    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

So from the Confusion Matrix statistics, we see that the in-sample accuracy value is 1 which is 100%!

Prediction applied to Test dataset

Next, we apply the model machine learning algorithm we built above, to each of the 20 test cases in the Test dataset.

```
assignmtans = predict(model, test)
assignmtans = as.character(assignmtans)
assignmtans
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

Conclusion

We have successfully built a model to predict exercise form based on movement data from accelerometers and we estimate the out of sample error to be extremely low. This is a promising result regarding the use of machine learning to detect bad exercise form in a small sample of six participants but we would expect the error of predicting bad form in real life situations on large groups of subjects to be higher.