# Activity Recognition of Weight Lifting Exercises

Bharathwaj Sundaresan

2/28/2021

## Executive Summary

The goal of this project is to predict the manner in which a particular exercise is done. 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. We will be to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants.They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. This project tends to investigate "how (well)" an activity was performed by the wearer.

## The Dataset

The dataset for this project is sourced from the research paper on Qualitative Activity Recognition of Weight Lifting Exercises. The data is extracted using an accelerometer to measure motion, force-sensing resistors to measure forces on the skier's feet and a gyroscope to measure rotation.

Six young health 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).

Data Source: Training , Test

## Exploratory Data Analysis

Lets load the data and do some exploratory plots.

```
set.seed(12345)
training<-read.csv("data/pml-training.csv")
testCases<-read.csv("data/pml-testing.csv")

dim(training)
```

```
## [1] 19622   160
```

The entire training data has 160 columns describing the accelerometer readings for 4 different locations (belt, forearm, arm, and dumbbell). Each location has 38 attributes which define that location.

Before plotting the attributes, Its also observed that the data set has many `na` and blank values which does not help in our assessment. These data points needs to be imputed, or ignored based on its relation to our outcome (class).

Lets look at the records which have more than one NA counts.

```
c<-training
c[ ,c("new_window","user_name","num_window","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestar
c<-data.frame(lapply(c, function(x) as.numeric(as.character(x))))
c<-cbind(training[c("classe")], c)

k<-data.frame(sapply(c, function(x) sum(is.na(x))))
k<-cbind(Column = rownames(k), k)
rownames(k)<-1:nrow(k)
colnames(k) = c("Column", "Null Counts")
nulls<-k[k$`Null Counts` !=0,]
rownames(nulls)<-1:nrow(nulls)
r<-nrow(nulls)
avg<-as.integer(sum(nulls$`Null Counts`)/nrow(nulls))
tail(nulls)
```

```
##                   Column Null Counts
## 95     avg_pitch_forearm       19216
## 96  stddev_pitch_forearm       19216
## 97     var_pitch_forearm       19216
## 98       avg_yaw_forearm       19216
## 99    stddev_yaw_forearm       19216
## 100      var_yaw_forearm       19216
```

From the above output, we can see that there are 100 attributes which have NA records, averaging around 19251 across all attributes. Upon checking all the attributes with na records. Most of the data points related to Euler's angle measurements (pitch, roll, yaw) have been reported as NULL. As more than 95% of data is missing. Imputing does not help here. Lets ignore these attributes from our analysis.

```
library(caret)
Final_data<-training[k[k$`Null Counts` ==0,]$Column]
part<-createDataPartition(Final_data$classe, p = 0.7, list = FALSE)
train<-Final_data[part,]
test<-Final_data[-part,]
dim(train)
```

```
## [1] 13737    53
```

```
dim(test)
```

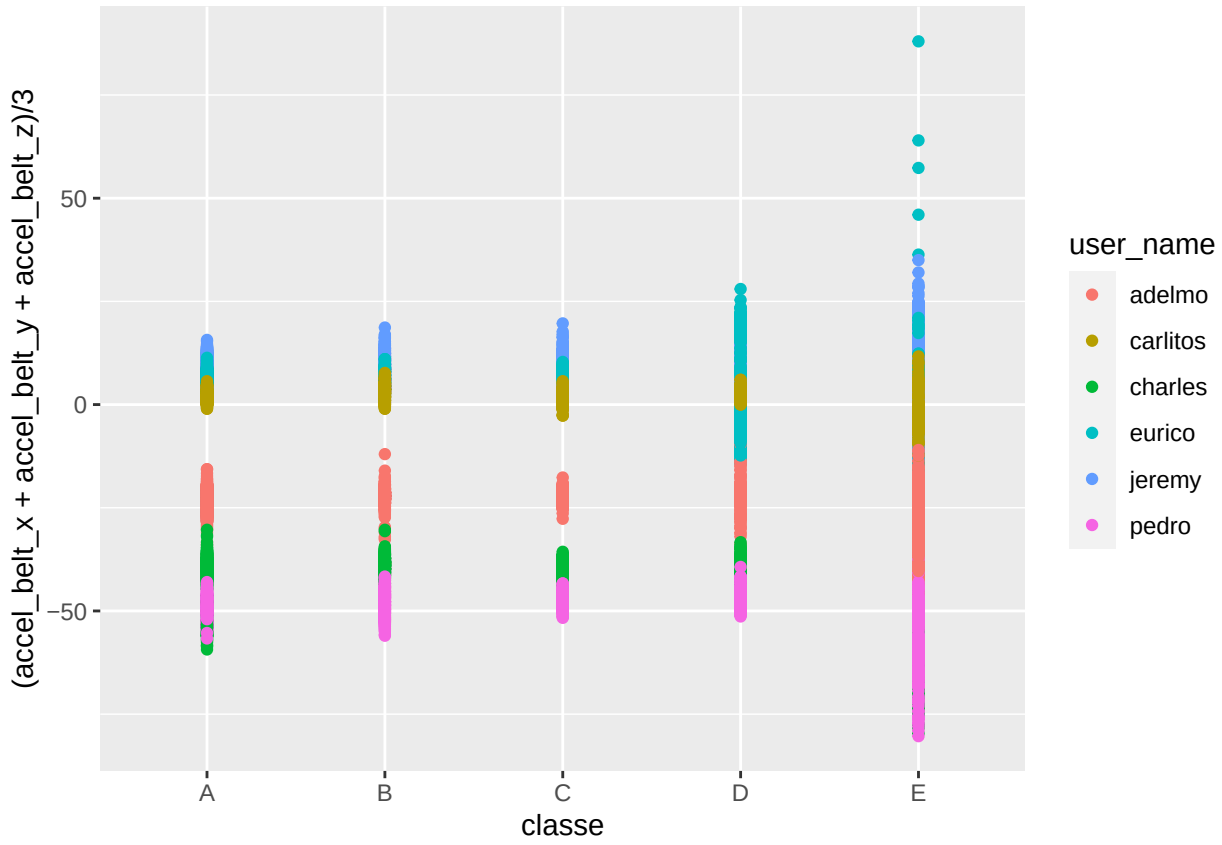
```
## [1] 5885   53
```

After removing 100 attributes related to Euler's measurements and the initial timestamp measurements. We have a dataset of 53 columns and 19622 rows. This is divided to get a training data of 13737 rows and testing data of 5885 rows.

Based on the original research paper.Lets do some exploratory plots comparing the accelerometer, gyroscope and magnetometer readings for each class by each user on the original data.

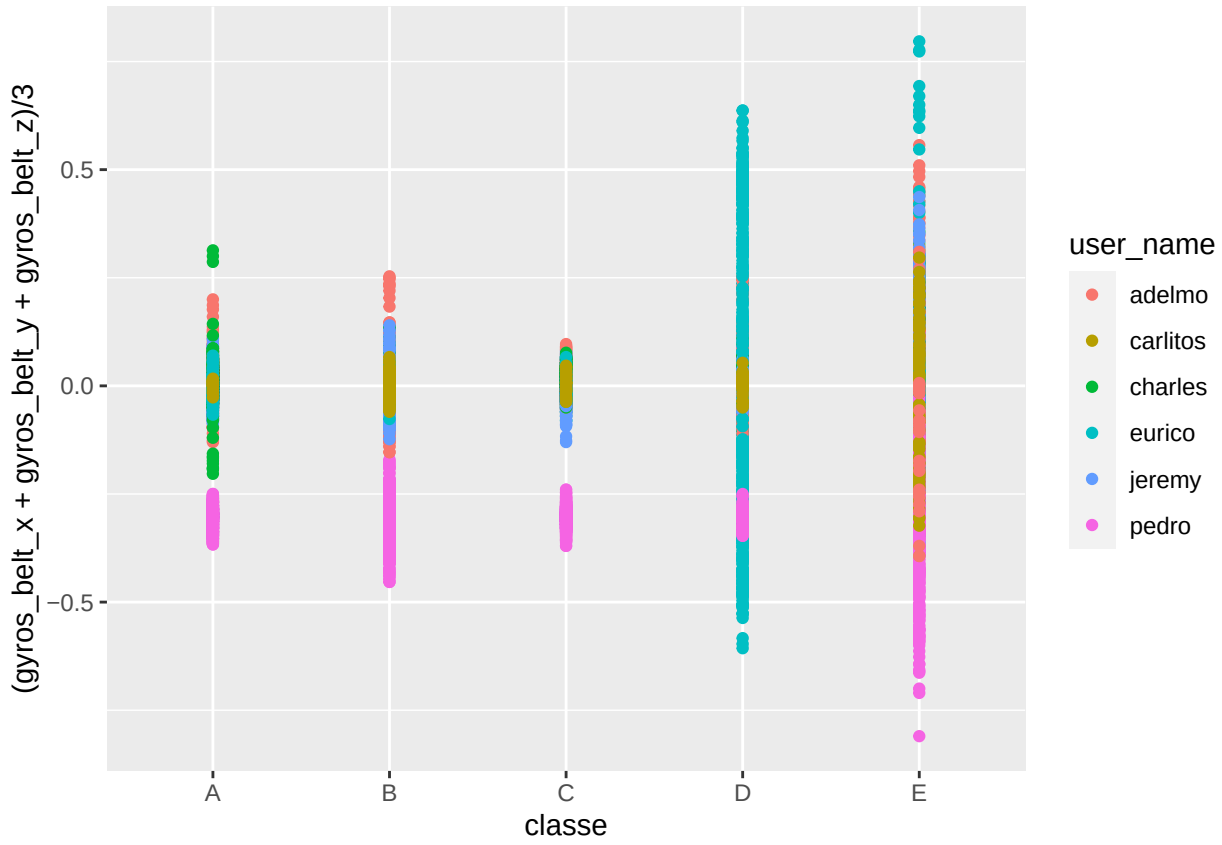
1.Belt Accelerometer average of vectors vs Class for each user_name

```
library(ggplot2)
qplot(classe, (accel_belt_x + accel_belt_y + accel_belt_z)/3, data = training, color = user_name)
```



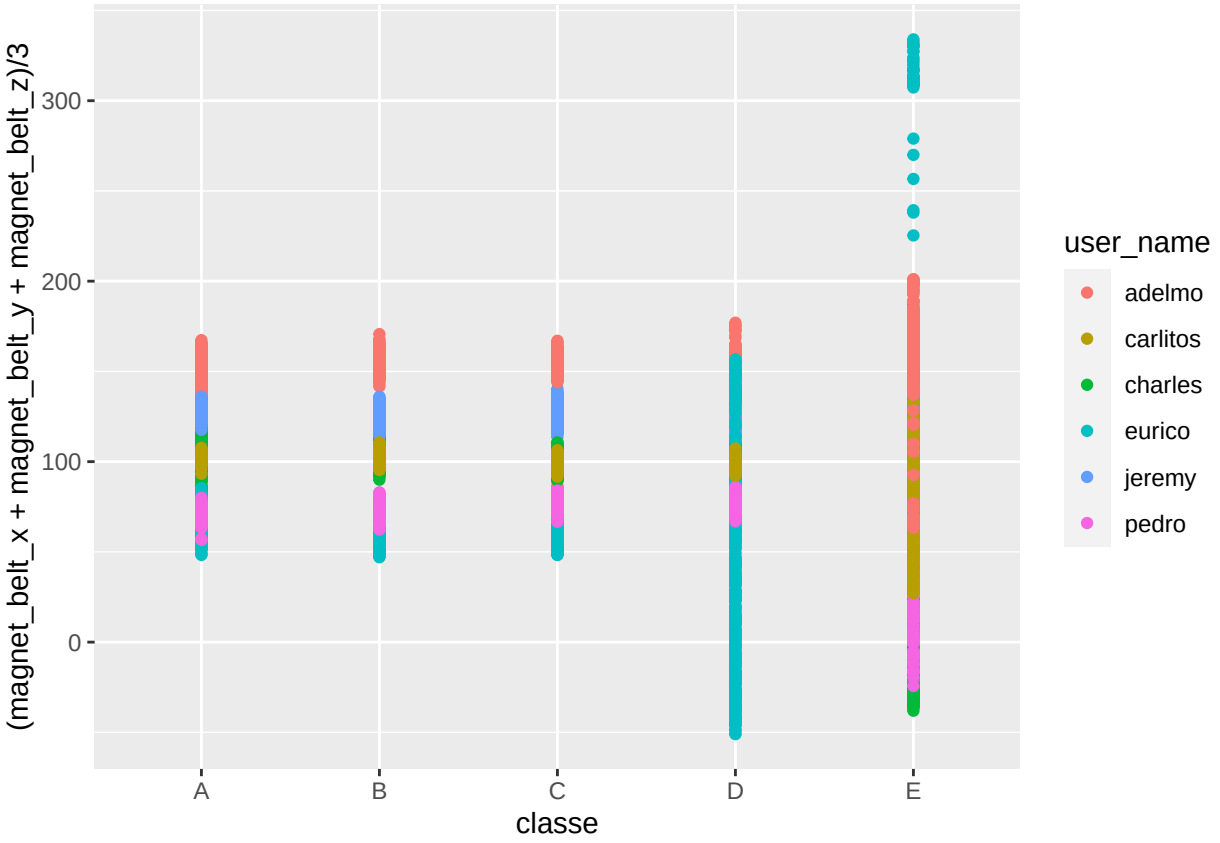

2.Gyroscope average of vectors vs Class for each user_name

```
qplot(classe, (gyros_belt_x + gyros_belt_y + gyros_belt_z)/3, data = training, color = user_name)
```




3.Magnetometer average of vectors vs Class for each user_name

```
qplot(classe, (magnet_belt_x + magnet_belt_y + magnet_belt_z)/3, data = training, color = user_name)
```




By averaging the vectors, we can see a clear distinction in readings for each user. Based on the data source, The class "A" is the right way to do the exercise. The remaining classes contribute to the intended noise. The similar trend is also observed for the readings from other locations.

## Feature Selection and Model Build

Based on the exploratory plots. We can see the data points have a characteristic noise in the sensor readings. Let us compare the efficiency of the Decision Tree model and the Random Forest model.

1. Decision Tree

```
library(rpart)
dtree<-rpart(classe~., data = train, method = "class")
pr_dtree<-predict(dtree, newdata = test, type = "class")
cnfnmtrx<-confusionMatrix(pr_dtree, factor(test$classe))
cnfnmtrx
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1532  176   28   48   41
##          B   54  585   57   64   76
##          C   35  154  819  134  126
##          D   25   76   58  631   56
```

```
##          E   28  148   64   87  783
##
## Overall Statistics
##
##                Accuracy : 0.7392
##                  95% CI : (0.7277, 0.7503)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6692
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9152  0.51361   0.7982   0.6546   0.7237
## Specificity            0.9304  0.94711   0.9076   0.9563   0.9319
## Pos Pred Value         0.8395  0.69976   0.6459   0.7459   0.7054
## Neg Pred Value         0.9650  0.89028   0.9552   0.9339   0.9374
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2603  0.09941   0.1392   0.1072   0.1331
## Detection Prevalence   0.3101  0.14206   0.2155   0.1438   0.1886
## Balanced Accuracy      0.9228  0.73036   0.8529   0.8054   0.8278
```

The overall accuracy of the model is 73%. However it has failed to identify few classes correctly.

Let us try to build a model using `Random Forest`

2. Random Forest

```
#Train control to define the K-folds. I have used 4 considering for bias and variance
tr<-trainControl(method = "cv", number = 3, verboseIter = FALSE)
rand<-train(classe~., data = train, method = "rf", trControl = tr)
rand$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: 2
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    4    0    0    0 0.001024066
## B   20 2634    4    0    0 0.009029345
## C    0   21 2372    3    0 0.010016694
## D    0    0   36 2214    2 0.016873890
## E    0    0    2    4 2519 0.002376238
```

With the characteristic noise in the data. Random Forest has been able to fix the overfit done in decision trees to the training data. The number of correctly predicted rows from the out-of-bag samples are also averaging around 0.7%

Lets calculate the confusion Matrix on Test data using this model.

```
pr_rand<-predict(rand, newdata = test)
cf_rand<-confusionMatrix(pr_rand, factor(test$classe))
cf_rand
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1134    5    0    0
##          C    0    1 1021   23    0
##          D    0    0    0  940    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9915, 0.9957)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9956   0.9951   0.9751   0.9991
## Specificity            0.9991   0.9987   0.9951   0.9998   0.9998
## Pos Pred Value         0.9976   0.9947   0.9770   0.9989   0.9991
## Neg Pred Value         0.9998   0.9989   0.9990   0.9951   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1927   0.1735   0.1597   0.1837
## Detection Prevalence   0.2850   0.1937   0.1776   0.1599   0.1839
## Balanced Accuracy      0.9992   0.9972   0.9951   0.9875   0.9994
```

Random Forest model has performed better with 99% accuracy, this gives a better prediction compared to the decision tree. Resulting in an accuracy of 99%. Lets select this model to predict the test case data points.

## Predict Test cases using select Model

```
pr_test_case<-predict(rand, newdata = testCases)
pr_test_case
```

```
##  [1] 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
```