# Practical Machine Learning Project

*Yash Prakash*

*25 August 2018*

# Data Analysis of Weight Lifting Exercises Dataset

## About the data

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, our 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (precisely, the section on the 'Weight Lifting Exercise Dataset').

## Objective

The goal of this project is to predict the manner in which the participants did the exercise. The 'classe' variable ranges from 'A' to 'E', which will be predicted for the test dataset.

## Loading the libraries needed

```
library(ggplot2)
library(lattice)
library(caret)
```

```
## Warning: replacing previous import by 'plyr::ddply' when loading 'caret'
```

```
## Warning: replacing previous import by 'tibble::as_tibble' when loading
## 'broom'
```

```
## Warning: replacing previous import by 'tibble::tibble' when loading 'broom'
```

```
## Warning: replacing previous import by 'rlang::!!' when loading 'recipes'
```

```
## Warning: replacing previous import by 'rlang::expr' when loading 'recipes'
```

```
## Warning: replacing previous import by 'rlang::f_lhs' when loading 'recipes'
```

```
## Warning: replacing previous import by 'rlang::f_rhs' when loading 'recipes'
```

```
## Warning: replacing previous import by 'rlang::invoke' when loading
## 'recipes'
```

```
## Warning: replacing previous import by 'rlang::is_empty' when loading
## 'recipes'
```

```
## Warning: replacing previous import by 'rlang::lang' when loading 'recipes'
```

```
## Warning: replacing previous import by 'rlang::na_dbl' when loading
## 'recipes'
```

```
## Warning: replacing previous import by 'rlang::names2' when loading
## 'recipes'
```

```
## Warning: replacing previous import by 'rlang::quos' when loading 'recipes'
```

```
## Warning: replacing previous import by 'rlang::sym' when loading 'recipes'
```

```
## Warning: replacing previous import by 'rlang::syms' when loading 'recipes'
```

# Loading the data

```
training<-read.csv('pml-training.csv',header=T)
testing<-read.csv('pml-testing.csv',header=T)
set.seed(12345)
```

# Data pre-processing

We see that the dataset contains a large number of non-useful arrtibutes that need not be used to train the model or predict from the model, like the name attribute, the time attribute, etc. We also need to remove the ‘na’ columns. We also convert the columns needed for the training of the model into factor variables.

```
training<-training[,colSums(is.na(training)) == 0]
i1 <- sapply(training,function(x)is.factor(x)&&length(levels(x))!=5)
training<-training[,!i1]
training<-training[,4:57]
```

We do the same for the test dataset.

```
testing<-testing[,colSums(is.na(testing)) == 0]
i1 <- sapply(testing,function(x)is.factor(x)&&length(levels(x))!=5)
testing<-testing[,!i1]
testing<-testing[4:57]
```

# Partitioning the data

We will be using ‘Random Forest’ to get maximum accuracy for the model, thus we resample the data.

Hence, we take the total training set and out of the total 19622 observations, we randomly sample 5000 observations into the train set and 1000 observations into the test set, achieving a 80:20 ratio for the data.

```
trainRows <- sample(nrow(training), 5000)
train <- training[trainRows,]
nrow(train)
```

```
## [1] 5000
```

```
ncol(train)
```

```
## [1] 54
```

```
testPre<-training[-trainRows,]
testRows <- sample(nrow(testPre), 1000)
test<- testPre[testRows,]
nrow(test)
```

```
## [1] 1000
```

```
ncol(test)
```

```
## [1] 54
```

# Fitting the model

Using the train function, method ‘rf’, we now fit the model.

```
trainctrl <- trainControl(verboseIter = TRUE)
modFit<-train(classe~ .,data=train,method="rf",proxy=T, trControl = trainctrl)
```

```
## + Resample01: mtry= 2
## - Resample01: mtry= 2
## + Resample01: mtry=27
## - Resample01: mtry=27
## + Resample01: mtry=53
## - Resample01: mtry=53
## + Resample02: mtry= 2
## - Resample02: mtry= 2
## + Resample02: mtry=27
## - Resample02: mtry=27
## + Resample02: mtry=53
## - Resample02: mtry=53
## + Resample03: mtry= 2
## - Resample03: mtry= 2
## + Resample03: mtry=27
## - Resample03: mtry=27
## + Resample03: mtry=53
## - Resample03: mtry=53
## + Resample04: mtry= 2
## - Resample04: mtry= 2
## + Resample04: mtry=27
## - Resample04: mtry=27
## + Resample04: mtry=53
## - Resample04: mtry=53
## + Resample05: mtry= 2
## - Resample05: mtry= 2
## + Resample05: mtry=27
## - Resample05: mtry=27
## + Resample05: mtry=53
## - Resample05: mtry=53
## + Resample06: mtry= 2
## - Resample06: mtry= 2
## + Resample06: mtry=27
## - Resample06: mtry=27
## + Resample06: mtry=53
## - Resample06: mtry=53
## + Resample07: mtry= 2
## - Resample07: mtry= 2
## + Resample07: mtry=27
## - Resample07: mtry=27
## + Resample07: mtry=53
## - Resample07: mtry=53
## + Resample08: mtry= 2
## - Resample08: mtry= 2
## + Resample08: mtry=27
## - Resample08: mtry=27
## + Resample08: mtry=53
## - Resample08: mtry=53
## + Resample09: mtry= 2
## - Resample09: mtry= 2
## + Resample09: mtry=27
## - Resample09: mtry=27
## + Resample09: mtry=53
## - Resample09: mtry=53
## + Resample10: mtry= 2
## - Resample10: mtry= 2
## + Resample10: mtry=27
```

```
## - Resample10: mtry=27
## + Resample10: mtry=53
## - Resample10: mtry=53
## + Resample11: mtry= 2
## - Resample11: mtry= 2
## + Resample11: mtry=27
## - Resample11: mtry=27
## + Resample11: mtry=53
## - Resample11: mtry=53
## + Resample12: mtry= 2
## - Resample12: mtry= 2
## + Resample12: mtry=27
## - Resample12: mtry=27
## + Resample12: mtry=53
## - Resample12: mtry=53
## + Resample13: mtry= 2
## - Resample13: mtry= 2
## + Resample13: mtry=27
## - Resample13: mtry=27
## + Resample13: mtry=53
## - Resample13: mtry=53
## + Resample14: mtry= 2
## - Resample14: mtry= 2
## + Resample14: mtry=27
## - Resample14: mtry=27
## + Resample14: mtry=53
## - Resample14: mtry=53
## + Resample15: mtry= 2
## - Resample15: mtry= 2
## + Resample15: mtry=27
## - Resample15: mtry=27
## + Resample15: mtry=53
## - Resample15: mtry=53
## + Resample16: mtry= 2
## - Resample16: mtry= 2
## + Resample16: mtry=27
## - Resample16: mtry=27
## + Resample16: mtry=53
## - Resample16: mtry=53
## + Resample17: mtry= 2
## - Resample17: mtry= 2
## + Resample17: mtry=27
## - Resample17: mtry=27
## + Resample17: mtry=53
## - Resample17: mtry=53
## + Resample18: mtry= 2
## - Resample18: mtry= 2
## + Resample18: mtry=27
## - Resample18: mtry=27
## + Resample18: mtry=53
## - Resample18: mtry=53
## + Resample19: mtry= 2
## - Resample19: mtry= 2
## + Resample19: mtry=27
## - Resample19: mtry=27
## + Resample19: mtry=53
## - Resample19: mtry=53
## + Resample20: mtry= 2
```

```
## - Resample20: mtry= 2
## + Resample20: mtry=27
## - Resample20: mtry=27
## + Resample20: mtry=53
## - Resample20: mtry=53
## + Resample21: mtry= 2
## - Resample21: mtry= 2
## + Resample21: mtry=27
## - Resample21: mtry=27
## + Resample21: mtry=53
## - Resample21: mtry=53
## + Resample22: mtry= 2
## - Resample22: mtry= 2
## + Resample22: mtry=27
## - Resample22: mtry=27
## + Resample22: mtry=53
## - Resample22: mtry=53
## + Resample23: mtry= 2
## - Resample23: mtry= 2
## + Resample23: mtry=27
## - Resample23: mtry=27
## + Resample23: mtry=53
## - Resample23: mtry=53
## + Resample24: mtry= 2
## - Resample24: mtry= 2
## + Resample24: mtry=27
## - Resample24: mtry=27
## + Resample24: mtry=53
## - Resample24: mtry=53
## + Resample25: mtry= 2
## - Resample25: mtry= 2
## + Resample25: mtry=27
## - Resample25: mtry=27
## + Resample25: mtry=53
## - Resample25: mtry=53
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 27 on full training set
```

# Comparing the results

We now predict the test set we sampled from the fitted model.

```
predictTest<-predict(modFit,test)
table(test$classe, predictTest)
```

```
##    predictTest
##       A   B   C   D   E
##   A 281   0   0   0   0
##   B   2 193   1   0   0
##   C   0   5 178   0   0
##   D   0   0   5 161   0
##   E   0   0   0   1 173
```

# We now draw the confusion matrix for the data to get the accuracy of the model.

```
confMat <- confusionMatrix(modFit)
confMat
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.2  0.4  0.0  0.0  0.0
##          B  0.0 18.4  0.4  0.0  0.1
##          C  0.0  0.2 16.6  0.3  0.0
##          D  0.0  0.0  0.1 17.0  0.2
##          E  0.0  0.0  0.0  0.0 18.1
##
##  Accuracy (average) : 0.9819
```

We get about 98% accuracy with the model.

# We now use the same model to get the prediction on the test dataset we obtained from the website. The dataset consists of 20 observations and our model will predict the class for each one.

```
prediction<-predict(modFit,testing)
table(prediction, 1:20)
```

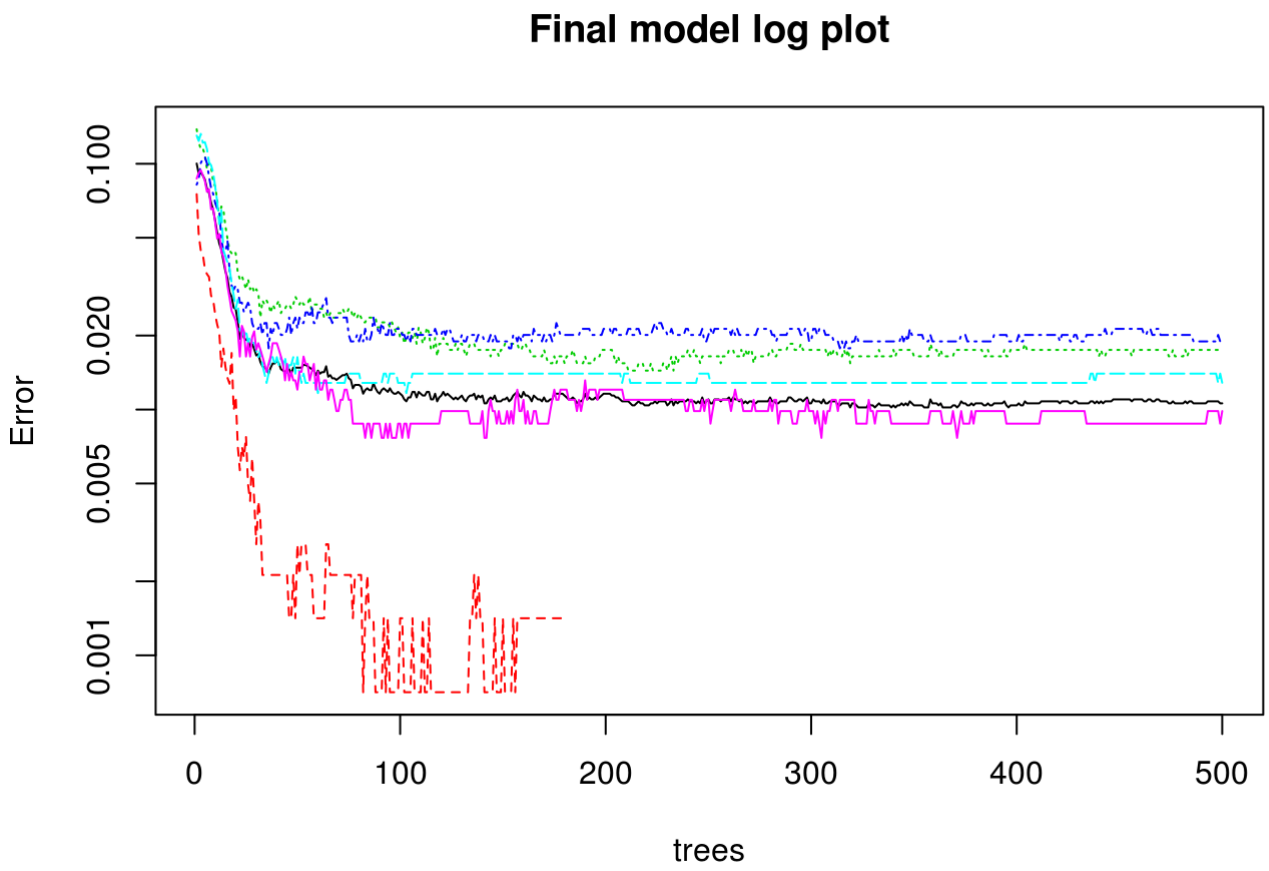
```
##
## prediction 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##          A 0 1 0 1 1 0 0 0 1  1  0  0  0  1  0  0  1  0  0  0
##          B 1 0 1 0 0 0 0 1 0  0  1  0  1  0  0  0  0  1  1  1
##          C 0 0 0 0 0 0 0 0 0  0  0  1  0  0  0  0  0  0  0  0
##          D 0 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0
##          E 0 0 0 0 0 1 0 0 0  0  0  0  0  0  1  1  0  0  0  0
```

This is the final prediction for all the 20 observations.

# Finally, we can see the plot for the model.

```
plot(modFit$finalModel,log="y",main="Final model log plot")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log = log): 320 y values <= 0
## omitted from logarithmic plot
```


Final model log plot
Error
0.001
0.005
0.020
0.100
0
100
200
300
400
500
trees