# Predictions: Weight Lifting Exercises Dataset

*Roberto Martinez de Morentin*

*26/09/2019*

## Overview

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. The aim of this project is to predict the manner in which participants perform a barbell lift. The data comes from http://groupware.les.inf.puc-rio.br/har wherein 6 participants were asked to perform the same set of exercises correctly and incorrectly with accelerometers placed on the belt, forearm, arm, and dumbell.

The following steps are followed:

1. Data Preprocessing
2. Exploratory Analysis
3. Prediction Model Selection
4. Predicting Test Set Output

## Data Preprocessing

Load the training and testing set from the online sources and then split the training set further into training and test sets.

```
library(caret)
setwd("~/GitHub/PracticalMachineLearning/")
train.url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test.url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(train.url))
testing <- read.csv(url(test.url))

label <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train <- training[label, ]
test <- training[-label, ]
```

Since from 160 variables present in the dataset, some variables have nearly zero variance while others contain a lot of NA terms which need to be excluded from the dataset. Let's remove those.

```
nzv <- nearZeroVar(train)
train <- train[ ,-nzv]
test <- test[ ,-nzv]

label <- apply(train, 2, function(x) mean(is.na(x))) > 0.95
train <- train[, -which(label, label == FALSE)]
test <- test[, -which(label, label == FALSE)]

train <- train[ , -(1:5)]
test <- test[ , -(1:5)]
```

We have reduced 160 variables to 54.

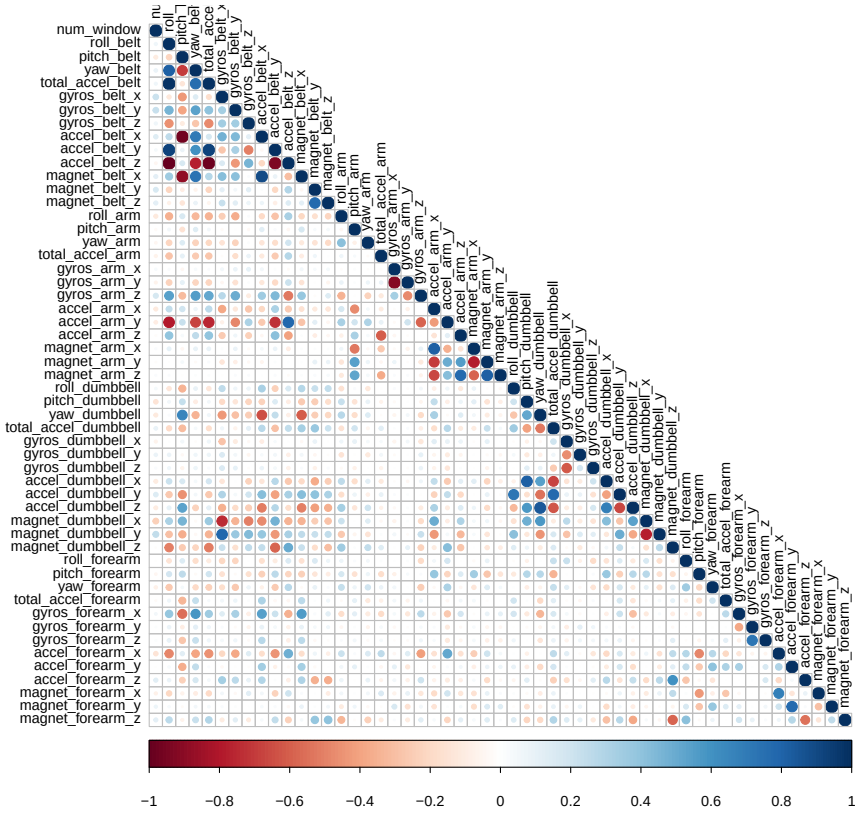
## Exploratory Analysis

After cleaneing the dataset off absolutely useless variables, we look at the correlation of these variables.

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrMat <- cor(train[,-54])
corrplot(corrMat, method = "circle", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```




Darker gradient correspond to having high correlation. A PCA could be run to further reduce the correlated variables but it not neccessary due to the few number of correlations.

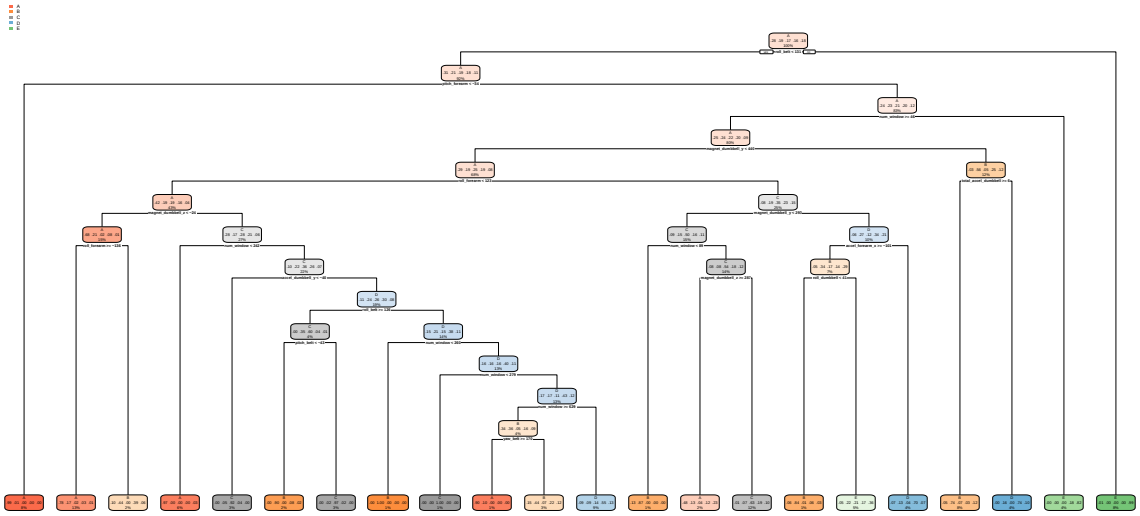
## Prediction Model Selection

We are going to use 3 methods to model the training set and thereby choose the one having the best accuracy to predict the outcome variable in the testing set. The methods will be Decision Tree, Random Forest and Generalized Boosted Model.

A confusion matrix plotted at the end of each model will help us to visualize each analysis.

### Decision Tree

```
library(rpart)
library(rpart.plot)
library(rattle)
set.seed(9306)
modelDT <- rpart(classe ~ ., data = train, method = "class")
rpart.plot(modelDT)
```



```
predictDT <- predict(modelDT, test, type = "class")
confMatDT <- confusionMatrix(predictDT, test$classe)
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1468  146   28   31   44
##          B   96  765   45  121   92
##          C    9   65  832  131   70
##          D   73  107   60  567  113
##          E   28   56   61  114  763
##
## Overall Statistics
##
##                Accuracy : 0.7468
##                  95% CI : (0.7355, 0.7579)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6795
##
##  Mcnemar's Test P-Value : 3.385e-13
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8769   0.6716   0.8109  0.58817   0.7052
## Specificity            0.9409   0.9254   0.9434  0.92827   0.9461
```

```
## Pos Pred Value         0.8550   0.6836   0.7516  0.61630   0.7466
## Neg Pred Value         0.9506   0.9215   0.9594  0.92004   0.9344
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2494   0.1300   0.1414  0.09635   0.1297
## Detection Prevalence   0.2918   0.1901   0.1881  0.15633   0.1737
## Balanced Accuracy      0.9089   0.7985   0.8772  0.75822   0.8256
```

### Random Forest

```
library(caret)
set.seed(9306)
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
modelRF <- train(classe ~ ., data = train, method = "rf", trControl = control)
modelRF$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: 27
##
##         OOB estimate of  error rate: 0.22%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    6 2650    2    0    0 0.003009782
## C    0    5 2391    0    0 0.002086811
## D    0    0   10 2241    1 0.004884547
## E    0    1    0    5 2519 0.002376238
```

```
predictRF <- predict(modelRF, test)
confMatRF <- confusionMatrix(predictRF, test$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    2    0    0    0
##          B    0 1134    2    0    2
##          C    0    3 1024    2    0
##          D    0    0    0  962    0
##          E    1    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.9964, 0.9989)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9956   0.9981   0.9979   0.9982
## Specificity            0.9995   0.9992   0.9990   1.0000   0.9998
## Pos Pred Value         0.9988   0.9965   0.9951   1.0000   0.9991
## Neg Pred Value         0.9998   0.9989   0.9996   0.9996   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1927   0.1740   0.1635   0.1835
## Detection Prevalence   0.2846   0.1934   0.1749   0.1635   0.1837
## Balanced Accuracy      0.9995   0.9974   0.9985   0.9990   0.9990
```

### Generalized Boosted Model

```
library(caret)
set.seed(9306)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelGBM <- train(classe ~ ., data = train, trControl = control, method = "gbm", verbose = FALSE)
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predictGBM <- predict(modelGBM, test)
confMatGBM <- confusionMatrix(predictGBM, test$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669   13    0    0    0
##          B    5 1104   13    3    2
##          C    0   17 1013   13    2
##          D    0    5    0  947   10
##          E    0    0    0    1 1068
##
## Overall Statistics
##
##                Accuracy : 0.9857
##                  95% CI : (0.9824, 0.9886)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9819
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9693   0.9873   0.9824   0.9871
## Specificity            0.9969   0.9952   0.9934   0.9970   0.9998
```

```
## Pos Pred Value         0.9923   0.9796   0.9694   0.9844   0.9991
## Neg Pred Value         0.9988   0.9926   0.9973   0.9965   0.9971
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1876   0.1721   0.1609   0.1815
## Detection Prevalence   0.2858   0.1915   0.1776   0.1635   0.1816
## Balanced Accuracy      0.9970   0.9822   0.9904   0.9897   0.9934
```

As Random Forest offers the maximum accuracy of 99.74%, we will go with Random Forest Model to predict our test data class variable.

## Predicting Test Set Output

```
predictRF <- predict(modelRF, testing)
predictRF
```

```
##  [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
```