# Project

MaryM

11/11/2019

```
## Run time: 2019-11-11 15:49:18
## R version: R version 3.6.1 (2019-07-05)
```

# Loading Required Packages

```
library(ggplot2)

library(lattice)

library(caret)

library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(knitr)
```

# Introduction

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, your 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.

For more information please refer to: http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX (http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX)

# Project Goal

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

# Reading and Cleaning the Data

# Reading the Data

```
# Load the training and testing data

TrainingData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
TestingData  <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))

# Partitioning the training dataset

inTrain  <- createDataPartition(TrainingData$classe, p=0.7, list=FALSE)

training <- TrainingData[inTrain, ] ; testing <- TrainingData[-inTrain, ]
```

```
dim(training)
```

```
## [1] 13737   160
```

```
dim(testing)
```

```
## [1] 5885  160
```

# Checking for NAs

Now, we should check and remove the NA values for cleaning the data.

```
NAs  <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, NAs==FALSE]
testing  <- testing[, NAs==FALSE]
```

```
dim(training)
```

```
## [1] 13737    93
```

```
dim(testing)
```

```
## [1] 5885   93
```

In addition, the near zero variance variables will be removed from the data.

```
NZV <- nearZeroVar(training)
training <- training[, -NZV]
testing  <- testing[, -NZV]
```

```
dim(training)
```

```
## [1] 13737    59
```

```
dim(testing)
```

```
## [1] 5885   59
```

Now, we can check the dat ahead to make sure that all the variables are needed to the final analysis.

```
head(training)
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 2 2  carlitos           1323084231               808298 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
## 5 5  carlitos           1323084232               196328 05/12/2011 11:23
## 6 6  carlitos           1323084232               304277 05/12/2011 11:23
## 7 7  carlitos           1323084232               368296 05/12/2011 11:23
## 8 8  carlitos           1323084232               440390 05/12/2011 11:23
##   num_window roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x
## 2         11      1.41       8.07    -94.4                3         0.02
## 4         12      1.48       8.05    -94.4                3         0.02
## 5         12      1.48       8.07    -94.4                3         0.02
## 6         12      1.45       8.06    -94.4                3         0.02
## 7         12      1.42       8.09    -94.4                3         0.02
## 8         12      1.42       8.13    -94.4                3         0.02
##   gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y accel_belt_z
## 2         0.00        -0.02          -22            4           22
## 4         0.00        -0.03          -22            3           21
## 5         0.02        -0.02          -21            2           24
## 6         0.00        -0.02          -21            4           21
## 7         0.00        -0.02          -22            3           21
## 8         0.00        -0.02          -22            4           21
##   magnet_belt_x magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm
## 2            -7           608          -311     -128      22.5    -161
## 4            -6           604          -310     -128      22.1    -161
## 5            -6           600          -302     -128      22.1    -161
## 6             0           603          -312     -128      22.0    -161
## 7            -4           599          -311     -128      21.9    -161
## 8            -2           603          -313     -128      21.8    -161
##   total_accel_arm gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x
## 2              34        0.02       -0.02       -0.02        -290
## 4              34        0.02       -0.03        0.02        -289
## 5              34        0.00       -0.03        0.00        -289
## 6              34        0.02       -0.03        0.00        -289
## 7              34        0.00       -0.03        0.00        -289
## 8              34        0.02       -0.02        0.00        -289
##   accel_arm_y accel_arm_z magnet_arm_x magnet_arm_y magnet_arm_z
## 2         110        -125         -369          337          513
## 4         111        -123         -372          344          512
## 5         111        -123         -374          337          506
## 6         111        -122         -369          342          513
## 7         111        -125         -373          336          509
## 8         111        -124         -372          338          510
##   roll_dumbbell pitch_dumbbell yaw_dumbbell total_accel_dumbbell
## 2      13.13074      -70.63751    -84.71065                   37
## 4      13.43120      -70.39379    -84.87363                   37
## 5      13.37872      -70.42856    -84.85306                   37
## 6      13.38246      -70.81759    -84.46500                   37
## 7      13.12695      -70.24757    -85.09961                   37
## 8      12.75083      -70.34768    -85.09708                   37
##   gyros_dumbbell_x gyros_dumbbell_y gyros_dumbbell_z accel_dumbbell_x
## 2                0            -0.02             0.00             -233
## 4                0            -0.02            -0.02             -232
## 5                0            -0.02             0.00             -233
```

```
## 6                0            -0.02             0.00             -234
## 7                0            -0.02             0.00             -232
## 8                0            -0.02             0.00             -234
##   accel_dumbbell_y accel_dumbbell_z magnet_dumbbell_x magnet_dumbbell_y
## 2               47             -269              -555                296
## 4               48             -269              -552                303
## 5               48             -270              -554                292
## 6               48             -269              -558                294
## 7               47             -270              -551                295
## 8               46             -272              -555                300
##   magnet_dumbbell_z roll_forearm pitch_forearm yaw_forearm
## 2               -64         28.3         -63.9        -153
## 4               -60         28.1         -63.9        -152
## 5               -68         28.0         -63.9        -152
## 6               -66         27.9         -63.9        -152
## 7               -70         27.9         -63.9        -152
## 8               -74         27.8         -63.8        -152
##   total_accel_forearm gyros_forearm_x gyros_forearm_y gyros_forearm_z
## 2                  36            0.02            0.00           -0.02
## 4                  36            0.02           -0.02            0.00
## 5                  36            0.02            0.00           -0.02
## 6                  36            0.02           -0.02           -0.03
## 7                  36            0.02            0.00           -0.02
## 8                  36            0.02           -0.02            0.00
##   accel_forearm_x accel_forearm_y accel_forearm_z magnet_forearm_x
## 2             192             203            -216              -18
## 4             189             206            -214              -16
## 5             189             206            -214              -17
## 6             193             203            -215               -9
## 7             195             205            -215              -18
## 8             193             205            -213               -9
##   magnet_forearm_y magnet_forearm_z classe
## 2              661              473      A
## 4              658              469      A
## 5              655              473      A
## 6              660              478      A
## 7              659              470      A
## 8              660              474      A
```

As seen, the first five colums are introductory and are not required for the analysis; therefore, those columns can be removed from the data:

```
training <- training[, -(1:5)]
testing  <- testing[, -(1:5)]
```

```
dim(training)
```

```
## [1] 13737    54
```

```
dim(testing)
```

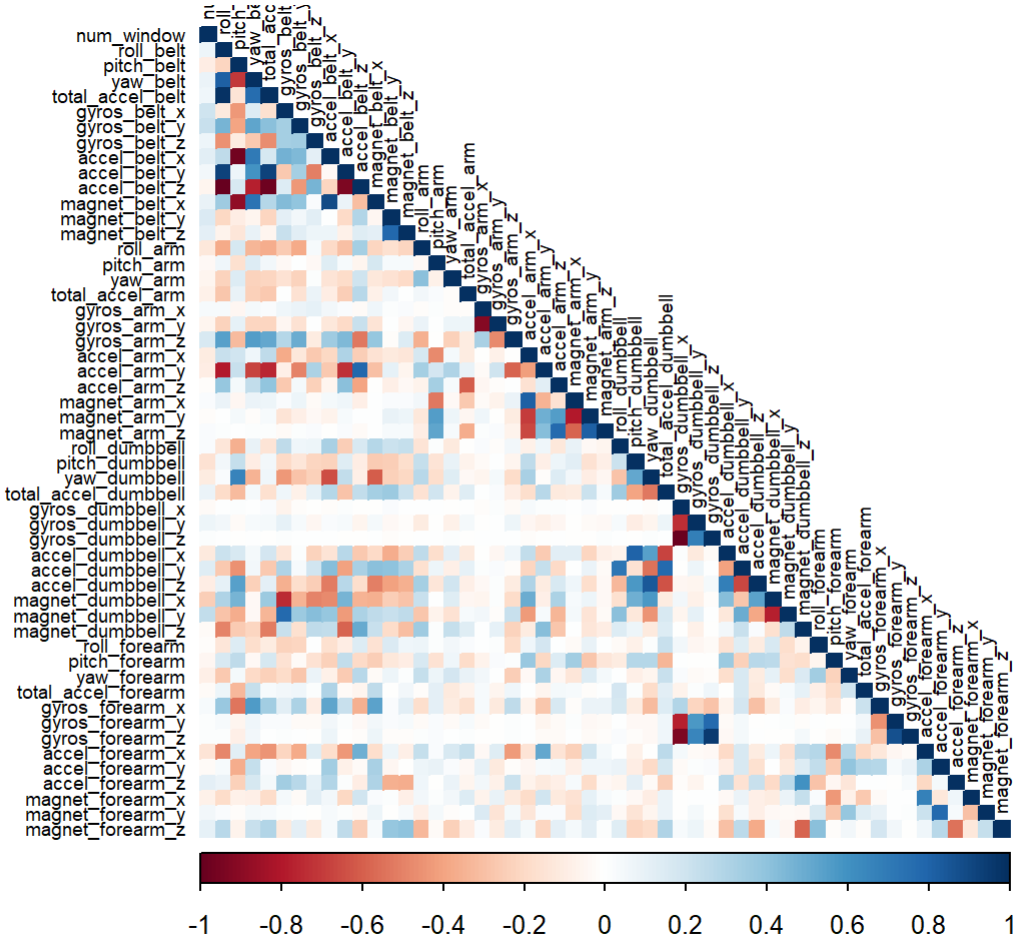
```
## [1] 5885   54
```

Eventually, after cleaning the data, 54 variables are left.

# Performing Correlation Analysis

Before conducting the machine learning model, we have to test for the correlation between the variables:

```
CorVar <- cor(training[, -54])  # the dependent variable should be removed from the correlation
 test
corrplot(CorVar,   type = "lower", method = 'color', #type only keeps a part of the data
         tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```




```
#method = "number" : gives numbers
```

As depicted in the figure, there are a few variables that are highly correlated. But, if there were more correlated variables, a PCA could be conducted to be able to develop the model.

# Developing the Prediction Model

## Random Forest

Random forest will be used to develop the prediction model.

```
set.seed(1111)

cntrl <- trainControl(method="cv", 5)

RFModel <- train(classe ~ ., data=training, method="rf", trControl=cntrl)

RFModel$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: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    6 2646    6    0    0 0.0045146727
## C    0    4 2392    0    0 0.0016694491
## D    0    0    8 2243    1 0.0039964476
## E    0    1    0    7 2517 0.0031683168
```

```
predictRF <- predict(RFModel, newdata = testing)
confMatRF<- confusionMatrix(predictRF, testing$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    2    0    0    0
##          B    1 1137    2    0    0
##          C    0    0 1024    5    0
##          D    0    0    0  959    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9982   0.9981   0.9948   0.9963
## Specificity            0.9995   0.9994   0.9990   0.9992   1.0000
## Pos Pred Value         0.9988   0.9974   0.9951   0.9958   1.0000
## Neg Pred Value         0.9998   0.9996   0.9996   0.9990   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1932   0.1740   0.1630   0.1832
## Detection Prevalence   0.2846   0.1937   0.1749   0.1636   0.1832
## Balanced Accuracy      0.9995   0.9988   0.9985   0.9970   0.9982
```

To calculate the model accuracy and overall out of sample error, the following code will be used.

```
accuracy <- postResample(predictRF, testing$classe)
acc.out <- accuracy[1]

overall.ose <-
        1 - as.numeric(confusionMatrix(testing$classe, predictRF)
                       $overall[1])

acc.out
```

```
##  Accuracy
## 0.9976211
```

```
acc.out
```

```
##  Accuracy
## 0.9976211
```

```
overall.ose
```

```
## [1] 0.002378929
```

Therefore, the moel accuracy is 0.9972812, and the overall out of sample error is 0.002718777.

# Data Validation

For validation process, the test dataset will be used.

```
predictTEST <- predict(RFModel, newdata=TestingData)
predictTEST
```

```
##  [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
```