*Montero, Jose Apr 4th, 2018*

# Classe Prediction Model

## 0 Executive summary

The goal of your project is to predict the manner in which they people engaged in a personal activity device test did the exercise. This is the “classe” variable in the training set.

The results of the analysis show that using a Random Forest algorithm, the accuracy obtained is very high (0.993, while the other 2 models studied remain at least 4 points below.

## 1 Data load and processing

First of all it is needed to load the packages that will be used later during the analysis, and also read the data files. Also a seed is set to allow reproducibility.

```
setwd("G:/My Drive/R/Rprogramming scripts/8.week4 practices")
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.4.4
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

```
library(dplyr)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.3
```

```
opts_chunk$set(cache=TRUE, cache.path = 'PML_cache/')
set.seed(4343)
```

```
trainingdata<-read.csv("./pml-training.csv", na.strings=c("NA","#DIV/0!", ""))
testingdata<-read.csv("./pml-testing.csv", na.strings=c("NA","#DIV/0!", ""))
```

Looking at the variables, we can see that there 2 cases that we could probably analyze to clean up

- Running a str command we see that the first 7 variables are not relevant for the analysis (ie row_id or timestamps)
- Just running a simple summary we see that there variables with many NAs.

We will remove the first lines, but also we will start the study getting rid of the variables with NAs, then we will see if our model is enough accurate or we need to go include variables with NAs.

```
trainingdata <-trainingdata[,-c(1:7)]
testingdata <-testingdata[,-c(1:7)]
trainingdata<-trainingdata[,colSums(is.na(trainingdata)) == 0]
```

# 2 Crossvalidation sets creation

Since we don't now the actual results for the testing set, we will split the training set in order to obtain one set for the training of the model (70% of the training observations) and another one to validate (30% of the training observations) ourselves the models.

```
trainChunk <- createDataPartition(y=trainingdata$classe,p=.70,list=F)
training <- trainingdata[trainChunk,]
validation <- trainingdata[-trainChunk,]
```

We could also see the boxplot in appendix A showing this initial feeling.

# 3 Models creation

We are going to create 3 different types of models.

Random Forest

```
mod_rf<-train(classe~., method="rf", data=training)
```

Boosted Predictor

```
mod_gbm<-train(classe~., method="gbm", data=training, verbose=FALSE)
```

Multivariate

```
mod_lda<-train(classe~., method="lda", data=training)
```

Let's have a look on the results in the training set.

```
mod_rf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9886649  0.9856571
##   27    0.9871570  0.9837497
##   52    0.9756667  0.9692098
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
mod_gbm
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7502889  0.6833911
##   1                  100      0.8157015  0.7666984
##   1                  150      0.8472803  0.8066836
##   2                   50      0.8494789  0.8092725
##   2                  100      0.9009272  0.8745887
##   2                  150      0.9260805  0.9064423
##   3                   50      0.8911315  0.8621360
##   3                  100      0.9368194  0.9200361
##   3                  150      0.9556924  0.9439346
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
mod_lda
```

```
## Linear Discriminant Analysis
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6957006  0.615084
```

# 4 Models evaluation

Now we are going to test the models with the validation set we created with the initial set of training data. Once the prediction is created, we will check the confusion matrixes.

```
pred_rf<-predict(mod_rf, validation)
pred_gbm<-predict(mod_gbm, validation)
pred_lda<-predict(mod_lda, validation)
confusionMatrix(pred_rf, validation$classe)$overall[1]
```

```
##  Accuracy
## 0.9932031
```

```
confusionMatrix(pred_gbm, validation$classe)$overall[1]
```

```
##  Accuracy
## 0.9629567
```

```
confusionMatrix(pred_lda, validation$classe)$overall[1]
```

```
##  Accuracy
## 0.7073917
```

As we can see in calculating the confusion matrixes of each model, **the hightest accuracy is obtained using Random Forest algorithm.** It is quite high, 0.993 of accuracy with 95% of confidence, we will not create a new model combining several ones.

# 5 Predict values for test data

```
pred_rf_test<-predict(mod_rf, testingdata)
pred_rf_test
```

```
##  [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
```