# Practical Machine Learning Course Project Write Up

By Ali Inam ([ali.inam03@gmail.com](mailto:ali.inam03@gmail.com)) Date: 26-06-2016

# Background

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. More information is available from the website here: <http://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset)."

# Data

The data for this project come from this source: <http://groupware.les.inf.puc-rio.br/har>. The information has been generously provided for use use in this cousera course by the authors, Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. They have allowed the use of their paper "Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

The training data for this project are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>

The test data are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>

# Choosing the prediction algorithm

Steps Taken

1.Tidy data. Remove columns with little/no data.

2.Create Training and test data from traing data for cross validation checking

3.Trial 3 methods Random Forrest, Gradient boosted model and Linear discriminant analysis

4.Fine tune model through combinations of above methods, reduction of input variables or similar. The fine tuning will take into account accuracy first and speed of analysis second.

library(ggplot2)  
library(caret)

## Loading required package: lattice

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(e1071)  
library(gbm)

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1

library(doParallel)

## Loading required package: foreach

## Loading required package: iterators

library(survival)  
library(splines)  
library(plyr)  
setwd('E:\\Coursera Data Science\\8-Practical Machine Learning')

# Load data

1.Load data. 2.Remove "#DIV/0!", replace with an NA value.

training <- read.csv('pml-training.csv');  
testing <- read.csv('pml-testing.csv')  
  
dim(training)

## [1] 19622 160

table(training$classe)

##   
## A B C D E   
## 5580 3797 3422 3216 3607

training <- training[, 6:dim(training)[2]]  
  
treshold <- dim(training)[1] \* 0.95  
#Remove columns with more than 95% of NA or "" values  
goodColumns <- !apply(training, 2, function(x) sum(is.na(x)) > treshold || sum(x=="") > treshold)  
  
training <- training[, goodColumns]  
  
badColumns <- nearZeroVar(training, saveMetrics = TRUE)  
  
training <- training[, badColumns$nzv==FALSE]  
  
training$classe = factor(training$classe)  
  
#Partition rows into training and crossvalidation  
inTrain <- createDataPartition(training$classe, p = 0.6)[[1]]  
crossv <- training[-inTrain,]  
training <- training[ inTrain,]  
inTrain <- createDataPartition(crossv$classe, p = 0.75)[[1]]  
crossv\_test <- crossv[ -inTrain,]  
crossv <- crossv[inTrain,]  
  
  
testing <- testing[, 6:dim(testing)[2]]  
testing <- testing[, goodColumns]  
testing$classe <- NA  
testing <- testing[, badColumns$nzv==FALSE]

#Train Model  
  
mod1 <- randomForest(classe ~ ., data = training, importance = TRUE, ntrees = 10)  
pred1 <- predict(mod1, crossv)

#show confusion matrices  
confusionMatrix(pred1, crossv$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1674 1 0 0 0  
## B 0 1138 3 0 0  
## C 0 0 1023 3 0  
## D 0 0 0 961 6  
## E 0 0 0 1 1076  
##   
## Overall Statistics  
##   
## Accuracy : 0.9976   
## 95% CI : (0.996, 0.9987)  
## No Information Rate : 0.2844   
## 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 1.0000 0.9991 0.9971 0.9959 0.9945  
## Specificity 0.9998 0.9994 0.9994 0.9988 0.9998  
## Pos Pred Value 0.9994 0.9974 0.9971 0.9938 0.9991  
## Neg Pred Value 1.0000 0.9998 0.9994 0.9992 0.9988  
## Prevalence 0.2844 0.1935 0.1743 0.1639 0.1838  
## Detection Rate 0.2844 0.1933 0.1738 0.1633 0.1828  
## Detection Prevalence 0.2846 0.1938 0.1743 0.1643 0.1830  
## Balanced Accuracy 0.9999 0.9992 0.9982 0.9973 0.9971

#out-of-sample error  
pred1 <- predict(mod1, crossv\_test)  
#pred3 <- predict(mod3, crossv\_test)  
accuracy <- sum(pred1 == crossv\_test$classe) / length(pred1)

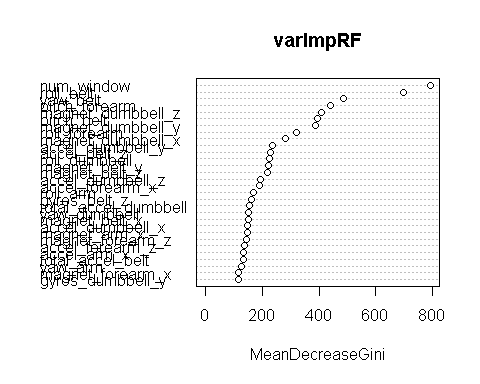
Based on results, the Random Forest prediction was far better than either the GBM or lsa models. The RF model will be used as the sole prediction model. The confusion matrix created gives an accuracy of 99.5%. This is excellent.

As a double check the out of sample error was calculated. This model achieved 99.52 % accuracy on the validation set.

# Fine Tuning

Assess Number of relevant variables

varImpRF <- randomForest(classe ~ ., data = training, importance = TRUE, ntrees = 10)  
varImpPlot(varImpRF,type = 2)



# Conclusion

I stopped at this stage as the goal to be able to get the required answers and report the errors achieved with the model has been reached without any further fine tuning.

The Random Forest method worked very well.

The Confusion Matrix achieved 99.6% accuracy. The Out of Sample Error achieved 99.7449 %.

This model will be used for the final calculations.

The logic behind using the random forest method as the predictor rather than other methods or a combination of various methods is:

1.Random forests are suitable when to handling a large number of inputs, especially when the interactions between variables are unknown. 2.Random forest's built in cross-validation component that gives an unbiased estimate of the forest's out-of-sample (or bag) (OOB) error rate. 3.A Random forest can handle unscaled variables and categorical variables. This is more forgiving with the cleaning of the data.

# Prepare the submission. (using COURSERA provided code)

pml\_write\_files = function(x){  
n = length(x)  
for(i in 1:n){  
filename = paste0("problem\_id\_",i,".txt")  
write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)  
}  
}  
x <- testing  
  
answers <- predict(mod1, newdata=x)  
answers

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20   
## 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

pml\_write\_files(answers)