ML Project

## 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

## Data Information

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>

## Import Data

# Data downloaded to local machine  
dataTest <- read.csv("pml-testing.csv", na.strings = c("NA", ""))  
dataTrain <- read.csv("pml-training.csv", na.strings = c("NA", ""))

## Data Cleaning and Processing

# Remove columns 1-7 as not good predictors  
testing <- dataTest[, -c(1:7)]  
training <- dataTrain[,-c(1:7)]  
  
# Count the number of NA's in each column  
NAtrain <- colSums(is.na(training))  
NAtest <- colSums(is.na(testing))  
  
# View NAtrain, multiple columns missing 19,216 entries (NAtest is the same)  
NAtrain

## roll\_belt pitch\_belt yaw\_belt   
## 0 0 0   
## total\_accel\_belt kurtosis\_roll\_belt kurtosis\_picth\_belt   
## 0 19216 19216   
## kurtosis\_yaw\_belt skewness\_roll\_belt skewness\_roll\_belt.1   
## 19216 19216 19216   
## skewness\_yaw\_belt max\_roll\_belt max\_picth\_belt   
## 19216 19216 19216   
## max\_yaw\_belt min\_roll\_belt min\_pitch\_belt   
## 19216 19216 19216   
## min\_yaw\_belt amplitude\_roll\_belt amplitude\_pitch\_belt   
## 19216 19216 19216   
## amplitude\_yaw\_belt var\_total\_accel\_belt avg\_roll\_belt   
## 19216 19216 19216   
## stddev\_roll\_belt var\_roll\_belt avg\_pitch\_belt   
## 19216 19216 19216   
## stddev\_pitch\_belt var\_pitch\_belt avg\_yaw\_belt   
## 19216 19216 19216   
## stddev\_yaw\_belt var\_yaw\_belt gyros\_belt\_x   
## 19216 19216 0   
## gyros\_belt\_y gyros\_belt\_z accel\_belt\_x   
## 0 0 0   
## accel\_belt\_y accel\_belt\_z magnet\_belt\_x   
## 0 0 0   
## magnet\_belt\_y magnet\_belt\_z roll\_arm   
## 0 0 0   
## pitch\_arm yaw\_arm total\_accel\_arm   
## 0 0 0   
## var\_accel\_arm avg\_roll\_arm stddev\_roll\_arm   
## 19216 19216 19216   
## var\_roll\_arm avg\_pitch\_arm stddev\_pitch\_arm   
## 19216 19216 19216   
## var\_pitch\_arm avg\_yaw\_arm stddev\_yaw\_arm   
## 19216 19216 19216   
## var\_yaw\_arm gyros\_arm\_x gyros\_arm\_y   
## 19216 0 0   
## gyros\_arm\_z accel\_arm\_x accel\_arm\_y   
## 0 0 0   
## accel\_arm\_z magnet\_arm\_x magnet\_arm\_y   
## 0 0 0   
## magnet\_arm\_z kurtosis\_roll\_arm kurtosis\_picth\_arm   
## 0 19216 19216   
## kurtosis\_yaw\_arm skewness\_roll\_arm skewness\_pitch\_arm   
## 19216 19216 19216   
## skewness\_yaw\_arm max\_roll\_arm max\_picth\_arm   
## 19216 19216 19216   
## max\_yaw\_arm min\_roll\_arm min\_pitch\_arm   
## 19216 19216 19216   
## min\_yaw\_arm amplitude\_roll\_arm amplitude\_pitch\_arm   
## 19216 19216 19216   
## amplitude\_yaw\_arm roll\_dumbbell pitch\_dumbbell   
## 19216 0 0   
## yaw\_dumbbell kurtosis\_roll\_dumbbell kurtosis\_picth\_dumbbell   
## 0 19216 19216   
## kurtosis\_yaw\_dumbbell skewness\_roll\_dumbbell skewness\_pitch\_dumbbell   
## 19216 19216 19216   
## skewness\_yaw\_dumbbell max\_roll\_dumbbell max\_picth\_dumbbell   
## 19216 19216 19216   
## max\_yaw\_dumbbell min\_roll\_dumbbell min\_pitch\_dumbbell   
## 19216 19216 19216   
## min\_yaw\_dumbbell amplitude\_roll\_dumbbell amplitude\_pitch\_dumbbell   
## 19216 19216 19216   
## amplitude\_yaw\_dumbbell total\_accel\_dumbbell var\_accel\_dumbbell   
## 19216 0 19216   
## avg\_roll\_dumbbell stddev\_roll\_dumbbell var\_roll\_dumbbell   
## 19216 19216 19216   
## avg\_pitch\_dumbbell stddev\_pitch\_dumbbell var\_pitch\_dumbbell   
## 19216 19216 19216   
## avg\_yaw\_dumbbell stddev\_yaw\_dumbbell var\_yaw\_dumbbell   
## 19216 19216 19216   
## gyros\_dumbbell\_x gyros\_dumbbell\_y gyros\_dumbbell\_z   
## 0 0 0   
## accel\_dumbbell\_x accel\_dumbbell\_y accel\_dumbbell\_z   
## 0 0 0   
## magnet\_dumbbell\_x magnet\_dumbbell\_y magnet\_dumbbell\_z   
## 0 0 0   
## roll\_forearm pitch\_forearm yaw\_forearm   
## 0 0 0   
## kurtosis\_roll\_forearm kurtosis\_picth\_forearm kurtosis\_yaw\_forearm   
## 19216 19216 19216   
## skewness\_roll\_forearm skewness\_pitch\_forearm skewness\_yaw\_forearm   
## 19216 19216 19216   
## max\_roll\_forearm max\_picth\_forearm max\_yaw\_forearm   
## 19216 19216 19216   
## min\_roll\_forearm min\_pitch\_forearm min\_yaw\_forearm   
## 19216 19216 19216   
## amplitude\_roll\_forearm amplitude\_pitch\_forearm amplitude\_yaw\_forearm   
## 19216 19216 19216   
## total\_accel\_forearm var\_accel\_forearm avg\_roll\_forearm   
## 0 19216 19216   
## stddev\_roll\_forearm var\_roll\_forearm avg\_pitch\_forearm   
## 19216 19216 19216   
## stddev\_pitch\_forearm var\_pitch\_forearm avg\_yaw\_forearm   
## 19216 19216 19216   
## stddev\_yaw\_forearm var\_yaw\_forearm gyros\_forearm\_x   
## 19216 19216 0   
## gyros\_forearm\_y gyros\_forearm\_z accel\_forearm\_x   
## 0 0 0   
## accel\_forearm\_y accel\_forearm\_z magnet\_forearm\_x   
## 0 0 0   
## magnet\_forearm\_y magnet\_forearm\_z classe   
## 0 0 0

# Count 100 columns of primarily empty entries  
length(NAtrain[NAtrain > 0])

## [1] 100

length(NAtest[NAtest >0])

## [1] 100

# Remove the mostly empty columns  
training <- training[,colSums(is.na(training)) == 0]  
testing <- testing[, colSums(is.na(testing))==0]

The training and testing data are now reduced from 160 columns to 53 columns. Also note that the training data contains the variable of interest ‘classe’ while the test data contains a column labeled ‘problem\_id’ (which is not a variable of interest for this analysis).

## Data Splitting

At this point, the testing data will be partitioned into 60% training data and 40% testing data. The original testing data will be redesignated as validation data.

partition <- createDataPartition(training$classe, p = .60, list = FALSE)  
train <- training[partition,]  
test <- training[-partition,]  
valid <- testing

## Setup Parallel Processing and train control

The method for establishing paralled processing is courteousy of Len Greski whose article can be found at <https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md)>.

library(parallel)  
library(doParallel)

## Loading required package: foreach

## Loading required package: iterators

cluster <- makeCluster(detectCores() - 1)   
registerDoParallel(cluster)  
  
#Setup the control method and parameters  
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)

## Model Development

The first model is a boost model using “gbm”.

modGBM <- train(classe ~ ., data = train, method = "gbm", trControl = fitControl)  
predictGBM <- predict(modGBM, newdata = test)   
cmGBM <- confusionMatrix(predictGBM, test$classe)

cmGBM

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2199 43 0 0 6  
## B 22 1429 46 5 7  
## C 6 45 1301 44 15  
## D 2 1 19 1229 17  
## E 3 0 2 8 1397  
##   
## Overall Statistics  
##   
## Accuracy : 0.9629   
## 95% CI : (0.9585, 0.967)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9531   
##   
## Mcnemar's Test P-Value : 4.895e-07   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9852 0.9414 0.9510 0.9557 0.9688  
## Specificity 0.9913 0.9874 0.9830 0.9941 0.9980  
## Pos Pred Value 0.9782 0.9470 0.9220 0.9692 0.9908  
## Neg Pred Value 0.9941 0.9860 0.9896 0.9913 0.9930  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2803 0.1821 0.1658 0.1566 0.1781  
## Detection Prevalence 0.2865 0.1923 0.1798 0.1616 0.1797  
## Balanced Accuracy 0.9882 0.9644 0.9670 0.9749 0.9834

The accuracy of this model is 0.962911. The out-of-sample error is 0.037089. We will see if this can be improved with a Random Forest model.

modRF <- train(classe ~ ., data = train, method = "rf", trControl = fitControl)  
predictRF <- predict(modRF, newdata = test)  
cmRF <- confusionMatrix(predictRF, test$classe)

cmRF

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2229 10 0 0 0  
## B 2 1507 12 0 1  
## C 1 1 1346 20 5  
## D 0 0 10 1265 6  
## E 0 0 0 1 1430  
##   
## Overall Statistics  
##   
## Accuracy : 0.9912   
## 95% CI : (0.9889, 0.9932)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9889   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9987 0.9928 0.9839 0.9837 0.9917  
## Specificity 0.9982 0.9976 0.9958 0.9976 0.9998  
## Pos Pred Value 0.9955 0.9901 0.9803 0.9875 0.9993  
## Neg Pred Value 0.9995 0.9983 0.9966 0.9968 0.9981  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2841 0.1921 0.1716 0.1612 0.1823  
## Detection Prevalence 0.2854 0.1940 0.1750 0.1633 0.1824  
## Balanced Accuracy 0.9984 0.9952 0.9899 0.9906 0.9958

The accuracy of this model is 0.9912057. The out-of-sample error is 0.0087943. This is the model selected for making final project predictions.

# Halt cluster for parallel processing  
stopCluster(cluster)  
registerDoSEQ()

## Final Predictions

The selected Random Forest model is used now to make final predictions on the validation set.

predictVALID <- predict(modRF, valid)  
predictVALID

## [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