PML Course Project Word

Steven Sheridan

April 27, 2019

# 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. 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).

# Prepare data

Load libraries

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'rattle' was built under R version 3.5.3

## Rattle: A free graphical interface for data science with R.  
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

Load training and test data

Train<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),header=TRUE)  
dim(Train)

## [1] 19622 160

Test<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header=TRUE)  
dim(Test)

## [1] 20 160

str(Test)

## 'data.frame': 20 obs. of 160 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ user\_name : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5 1 4 5 5 5 2 3 ...  
## $ raw\_timestamp\_part\_1 : int 1323095002 1322673067 1322673075 1322832789 1322489635 1322673149 1322673128 1322673076 1323084240 1322837822 ...  
## $ raw\_timestamp\_part\_2 : int 868349 778725 342967 560311 814776 510661 766645 54671 916313 384285 ...  
## $ cvtd\_timestamp : Factor w/ 11 levels "02/12/2011 13:33",..: 5 10 10 1 6 11 11 10 3 2 ...  
## $ new\_window : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1 ...  
## $ num\_window : int 74 431 439 194 235 504 485 440 323 664 ...  
## $ roll\_belt : num 123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...  
## $ pitch\_belt : num 27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...  
## $ yaw\_belt : num -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...  
## $ total\_accel\_belt : int 20 4 5 17 3 4 4 4 4 18 ...  
## $ kurtosis\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ kurtosis\_picth\_belt : logi NA NA NA NA NA NA ...  
## $ kurtosis\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ skewness\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ skewness\_roll\_belt.1 : logi NA NA NA NA NA NA ...  
## $ skewness\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ max\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ max\_picth\_belt : logi NA NA NA NA NA NA ...  
## $ max\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ min\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ min\_pitch\_belt : logi NA NA NA NA NA NA ...  
## $ min\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ amplitude\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ amplitude\_pitch\_belt : logi NA NA NA NA NA NA ...  
## $ amplitude\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ var\_total\_accel\_belt : logi NA NA NA NA NA NA ...  
## $ avg\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ stddev\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ var\_roll\_belt : logi NA NA NA NA NA NA ...  
## $ avg\_pitch\_belt : logi NA NA NA NA NA NA ...  
## $ stddev\_pitch\_belt : logi NA NA NA NA NA NA ...  
## $ var\_pitch\_belt : logi NA NA NA NA NA NA ...  
## $ avg\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ stddev\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ var\_yaw\_belt : logi NA NA NA NA NA NA ...  
## $ gyros\_belt\_x : num -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...  
## $ gyros\_belt\_y : num -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...  
## $ gyros\_belt\_z : num -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...  
## $ accel\_belt\_x : int -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...  
## $ accel\_belt\_y : int 69 11 -1 45 4 -16 2 -2 1 63 ...  
## $ accel\_belt\_z : int -179 39 49 -156 27 38 35 42 32 -158 ...  
## $ magnet\_belt\_x : int -13 43 29 169 33 31 50 39 -6 10 ...  
## $ magnet\_belt\_y : int 581 636 631 608 566 638 622 635 600 601 ...  
## $ magnet\_belt\_z : int -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...  
## $ roll\_arm : num 40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...  
## $ pitch\_arm : num -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...  
## $ yaw\_arm : num 178 0 0 -142 102 0 0 0 -167 -75.3 ...  
## $ total\_accel\_arm : int 10 38 44 25 29 14 15 22 34 32 ...  
## $ var\_accel\_arm : logi NA NA NA NA NA NA ...  
## $ avg\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ stddev\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ var\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ avg\_pitch\_arm : logi NA NA NA NA NA NA ...  
## $ stddev\_pitch\_arm : logi NA NA NA NA NA NA ...  
## $ var\_pitch\_arm : logi NA NA NA NA NA NA ...  
## $ avg\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ stddev\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ var\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ gyros\_arm\_x : num -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...  
## $ gyros\_arm\_y : num 0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...  
## $ gyros\_arm\_z : num -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...  
## $ accel\_arm\_x : int 16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...  
## $ accel\_arm\_y : int 38 215 245 -57 200 130 79 175 111 -42 ...  
## $ accel\_arm\_z : int 93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...  
## $ magnet\_arm\_x : int -326 -325 -264 -173 -170 396 702 535 -367 -420 ...  
## $ magnet\_arm\_y : int 385 447 474 257 275 176 15 215 335 294 ...  
## $ magnet\_arm\_z : int 481 434 413 633 617 516 217 385 520 493 ...  
## $ kurtosis\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ kurtosis\_picth\_arm : logi NA NA NA NA NA NA ...  
## $ kurtosis\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ skewness\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ skewness\_pitch\_arm : logi NA NA NA NA NA NA ...  
## $ skewness\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ max\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ max\_picth\_arm : logi NA NA NA NA NA NA ...  
## $ max\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ min\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ min\_pitch\_arm : logi NA NA NA NA NA NA ...  
## $ min\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ amplitude\_roll\_arm : logi NA NA NA NA NA NA ...  
## $ amplitude\_pitch\_arm : logi NA NA NA NA NA NA ...  
## $ amplitude\_yaw\_arm : logi NA NA NA NA NA NA ...  
## $ roll\_dumbbell : num -17.7 54.5 57.1 43.1 -101.4 ...  
## $ pitch\_dumbbell : num 25 -53.7 -51.4 -30 -53.4 ...  
## $ yaw\_dumbbell : num 126.2 -75.5 -75.2 -103.3 -14.2 ...  
## $ kurtosis\_roll\_dumbbell : logi NA NA NA NA NA NA ...  
## $ kurtosis\_picth\_dumbbell : logi NA NA NA NA NA NA ...  
## $ kurtosis\_yaw\_dumbbell : logi NA NA NA NA NA NA ...  
## $ skewness\_roll\_dumbbell : logi NA NA NA NA NA NA ...  
## $ skewness\_pitch\_dumbbell : logi NA NA NA NA NA NA ...  
## $ skewness\_yaw\_dumbbell : logi NA NA NA NA NA NA ...  
## $ max\_roll\_dumbbell : logi NA NA NA NA NA NA ...  
## $ max\_picth\_dumbbell : logi NA NA NA NA NA NA ...  
## $ max\_yaw\_dumbbell : logi NA NA NA NA NA NA ...  
## $ min\_roll\_dumbbell : logi NA NA NA NA NA NA ...  
## $ min\_pitch\_dumbbell : logi NA NA NA NA NA NA ...  
## $ min\_yaw\_dumbbell : logi NA NA NA NA NA NA ...  
## $ amplitude\_roll\_dumbbell : logi NA NA NA NA NA NA ...  
## [list output truncated]

Remove columns with many missing values

Remove <- which(colSums(is.na(Train) |Train=="")>0.9\*dim(Train)[1])   
TrainClean<-Train[,-Remove]  
TrainClean<-TrainClean[,-c(1:7)]  
TestClean<-Test[,-Remove]  
TestClean<-TestClean[,-1]  
dim(TrainClean)

## [1] 19622 53

dim(TestClean)

## [1] 20 59

After cleaning, the training set has 53 variables.

Partition the data

set.seed(2319)  
DP <- createDataPartition(TrainClean$classe, p=0.75, list=FALSE)  
TrainP <- TrainClean[DP,]  
TestP <- TrainClean[-DP,]  
dim(TrainP)

## [1] 14718 53

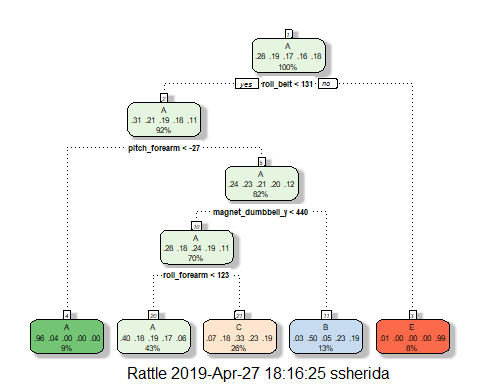
dim(TestP)

## [1] 4904 53

Three different methods for prediction are tested below. The cross-validation technique is being applied and five folds are being used.

# Classification Tree

trControl <- trainControl(method="cv", number=5)  
model\_CT <- train(classe~., data=TrainP, method="rpart", trControl=trControl)  
fancyRpartPlot(model\_CT$finalModel)



predCT <- predict(model\_CT,newdata=TestP)  
CMCT <- confusionMatrix(TestP$classe,predCT)  
CMCT$table

## Reference  
## Prediction A B C D E  
## A 1266 28 97 0 4  
## B 414 322 213 0 0  
## C 393 23 439 0 0  
## D 364 133 307 0 0  
## E 138 125 243 0 395

CMCT$overall[1]

## Accuracy   
## 0.4938825

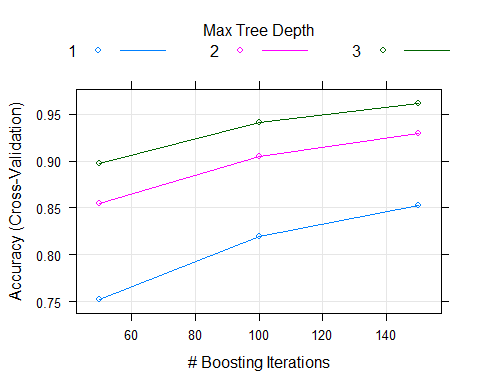
This model has very low accuracy of just about 50%.

# Gradient Boosting Method

model\_GBM <- train(classe~., data=TrainP, method="gbm", trControl=trControl, verbose=FALSE)  
print(model\_GBM)

## Stochastic Gradient Boosting   
##   
## 14718 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 11776, 11775, 11774, 11774, 11773   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.7524117 0.6861269  
## 1 100 0.8200157 0.7721644  
## 1 150 0.8527654 0.8136323  
## 2 50 0.8552119 0.8165488  
## 2 100 0.9052184 0.8800679  
## 2 150 0.9292031 0.9104162  
## 3 50 0.8974047 0.8701083  
## 3 100 0.9415008 0.9259851  
## 3 150 0.9615437 0.9513537  
##   
## 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.

plot(model\_GBM)



predGBM<-predict(model\_GBM,newdata=TestP)  
CMGBM<-confusionMatrix(TestP$classe,predGBM)  
CMGBM$table

## Reference  
## Prediction A B C D E  
## A 1371 14 7 3 0  
## B 26 894 28 1 0  
## C 0 27 818 10 0  
## D 0 2 20 776 6  
## E 2 14 8 11 866

CMGBM$overall[1]

## Accuracy   
## 0.9634992

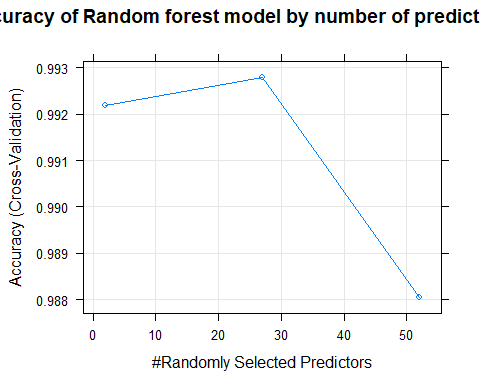
This model peformed much better than the classification tree, but there is still some room for improvement.

# Random Forests Method

model\_RF<-train(classe~., data=TrainP, method="rf", trControl=trControl, verbose=FALSE)  
print(model\_RF)

## Random Forest   
##   
## 14718 samples  
## 52 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 11773, 11775, 11775, 11776, 11773   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9921863 0.9901150  
## 27 0.9927976 0.9908884  
## 52 0.9880419 0.9848723  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 27.

plot(model\_RF,main="Accuracy of Random forest model by number of predictors")



predRF<-predict(model\_RF,newdata=TestP)  
CMRF<-confusionMatrix(TestP$classe,predRF)  
CMRF$table

## Reference  
## Prediction A B C D E  
## A 1393 2 0 0 0  
## B 4 939 6 0 0  
## C 0 1 853 1 0  
## D 0 0 2 801 1  
## E 0 1 1 2 897

CMRF$overall[1]

## Accuracy   
## 0.9957178

# Summary

The random forests model performs better (over 99% accuracy) than the other two methods and will be used for prediction.