Machine Learnign Final Project

Javier E. Sanmiguel

1/28/2021

## 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 dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways, exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. More details can be found in the paper “Qualitative Activity Recognition of Weight Lifting Exercises” written by Eduardo Velloso et. al., which can be found at this site: <http://web.archive.org/web/20170519033209/http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf>

Data to conduct this analysis is found in these sites. 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>

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.

## Methodology

The following steps were followed in order to determine what is the best way to predict “classe” on function of the data acquired through devices located on different parts of the participant bodies.

1. Download, and read the data
2. Inspect the data
3. Clean the data by removing variables (columns) that don’t have values/NA, or great majority are zeros)
4. Analyze the train set by creating models using different techniques such as “Predicting with Trees”, “Random Forest”, “Boosting”, “Linear Discriminant Analysis”, and finally “Combining Predictors” 5 A summary of the results will be provided and a recommendation on what is the most applicable technique to predict “classe”

## Downloading, reading and saving data sets

Use the above links to download, read and save the data sets.

traindataURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
download.file(traindataURL, destfile = "./pml-training.csv")  
pmlTrainData <- read.csv("./pml-training.csv")  
  
testdataURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  
download.file(testdataURL, destfile = "./pml-testing.csv")  
pmlTestData <- read.csv("./pml-testing.csv")

## Data inspection and cleaning

The data set will be inspected and all the variables without values, zeros, and/or NA will be remove. Them the training set will be split in training set and validation set. The test set will be used to confirm the accuracy of the model.

dim(pmlTrainData)

## [1] 19622 160

str(pmlTrainData)

## 'data.frame': 19622 obs. of 160 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ user\_name : chr "carlitos" "carlitos" "carlitos" "carlitos" ...  
## $ raw\_timestamp\_part\_1 : int 1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...  
## $ raw\_timestamp\_part\_2 : int 788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...  
## $ cvtd\_timestamp : chr "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" ...  
## $ new\_window : chr "no" "no" "no" "no" ...  
## $ num\_window : int 11 11 11 12 12 12 12 12 12 12 ...  
## $ roll\_belt : num 1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...  
## $ pitch\_belt : num 8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...  
## $ yaw\_belt : num -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...  
## $ total\_accel\_belt : int 3 3 3 3 3 3 3 3 3 3 ...  
## $ kurtosis\_roll\_belt : chr "" "" "" "" ...  
## $ kurtosis\_picth\_belt : chr "" "" "" "" ...  
## $ kurtosis\_yaw\_belt : chr "" "" "" "" ...  
## $ skewness\_roll\_belt : chr "" "" "" "" ...  
## $ skewness\_roll\_belt.1 : chr "" "" "" "" ...  
## $ skewness\_yaw\_belt : chr "" "" "" "" ...  
## $ max\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_picth\_belt : int NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_yaw\_belt : chr "" "" "" "" ...  
## $ min\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_pitch\_belt : int NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_yaw\_belt : chr "" "" "" "" ...  
## $ amplitude\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_pitch\_belt : int NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_yaw\_belt : chr "" "" "" "" ...  
## $ var\_total\_accel\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_roll\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_pitch\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_yaw\_belt : num NA NA NA NA NA NA NA NA NA NA ...  
## $ gyros\_belt\_x : num 0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...  
## $ gyros\_belt\_y : num 0 0 0 0 0.02 0 0 0 0 0 ...  
## $ gyros\_belt\_z : num -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...  
## $ accel\_belt\_x : int -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...  
## $ accel\_belt\_y : int 4 4 5 3 2 4 3 4 2 4 ...  
## $ accel\_belt\_z : int 22 22 23 21 24 21 21 21 24 22 ...  
## $ magnet\_belt\_x : int -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...  
## $ magnet\_belt\_y : int 599 608 600 604 600 603 599 603 602 609 ...  
## $ magnet\_belt\_z : int -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...  
## $ roll\_arm : num -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...  
## $ pitch\_arm : num 22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...  
## $ yaw\_arm : num -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...  
## $ total\_accel\_arm : int 34 34 34 34 34 34 34 34 34 34 ...  
## $ var\_accel\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ avg\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ stddev\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ var\_yaw\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ gyros\_arm\_x : num 0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...  
## $ gyros\_arm\_y : num 0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...  
## $ gyros\_arm\_z : num -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...  
## $ accel\_arm\_x : int -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...  
## $ accel\_arm\_y : int 109 110 110 111 111 111 111 111 109 110 ...  
## $ accel\_arm\_z : int -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...  
## $ magnet\_arm\_x : int -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...  
## $ magnet\_arm\_y : int 337 337 344 344 337 342 336 338 341 334 ...  
## $ magnet\_arm\_z : int 516 513 513 512 506 513 509 510 518 516 ...  
## $ kurtosis\_roll\_arm : chr "" "" "" "" ...  
## $ kurtosis\_picth\_arm : chr "" "" "" "" ...  
## $ kurtosis\_yaw\_arm : chr "" "" "" "" ...  
## $ skewness\_roll\_arm : chr "" "" "" "" ...  
## $ skewness\_pitch\_arm : chr "" "" "" "" ...  
## $ skewness\_yaw\_arm : chr "" "" "" "" ...  
## $ max\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_picth\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_roll\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_pitch\_arm : num NA NA NA NA NA NA NA NA NA NA ...  
## $ amplitude\_yaw\_arm : int NA NA NA NA NA NA NA NA NA NA ...  
## $ roll\_dumbbell : num 13.1 13.1 12.9 13.4 13.4 ...  
## $ pitch\_dumbbell : num -70.5 -70.6 -70.3 -70.4 -70.4 ...  
## $ yaw\_dumbbell : num -84.9 -84.7 -85.1 -84.9 -84.9 ...  
## $ kurtosis\_roll\_dumbbell : chr "" "" "" "" ...  
## $ kurtosis\_picth\_dumbbell : chr "" "" "" "" ...  
## $ kurtosis\_yaw\_dumbbell : chr "" "" "" "" ...  
## $ skewness\_roll\_dumbbell : chr "" "" "" "" ...  
## $ skewness\_pitch\_dumbbell : chr "" "" "" "" ...  
## $ skewness\_yaw\_dumbbell : chr "" "" "" "" ...  
## $ max\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_picth\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ max\_yaw\_dumbbell : chr "" "" "" "" ...  
## $ min\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_pitch\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## $ min\_yaw\_dumbbell : chr "" "" "" "" ...  
## $ amplitude\_roll\_dumbbell : num NA NA NA NA NA NA NA NA NA NA ...  
## [list output truncated]

# Remove all the variable with zeros, NA out of the train set  
# Here we get the indexes of the columns having at least 90% of NA or blank values   
indColToRemoveTrain <- which(colSums(is.na(pmlTrainData) |pmlTrainData=="")>0.9\*dim(pmlTrainData)[1])   
pmlTrainDataClean <- pmlTrainData[,-indColToRemoveTrain]  
pmlTrainDataClean <- pmlTrainDataClean[,-c(1:7)]  
  
dim(pmlTrainDataClean)

## [1] 19622 53

# The same protocol will be done in test data set  
indColToRemoveTest <- which(colSums(is.na(pmlTestData) |pmlTestData=="")>0.9\*dim(pmlTestData)[1])   
pmlTestDataClean <- pmlTestData[,-indColToRemoveTest]  
pmlTestDataClean <- pmlTestDataClean[,-c(1:7)]  
  
dim(pmlTestDataClean)

## [1] 20 53

## Setting up Training, Validation and Test sets

The train data set will be split in training set (75%) and Validation set (25%). The test set will not be changed.

set.seed(20210129)  
inTrain <- createDataPartition(pmlTrainDataClean$classe, p=3/4)[[1]]  
pmlTraining <- pmlTrainDataClean[inTrain, ]  
pmlValidation <- pmlTrainDataClean[-inTrain, ]  
pmlTesting <- pmlTestDataClean  
dim(pmlTraining)

## [1] 14718 53

dim(pmlValidation)

## [1] 4904 53

dim(pmlTesting)

## [1] 20 53

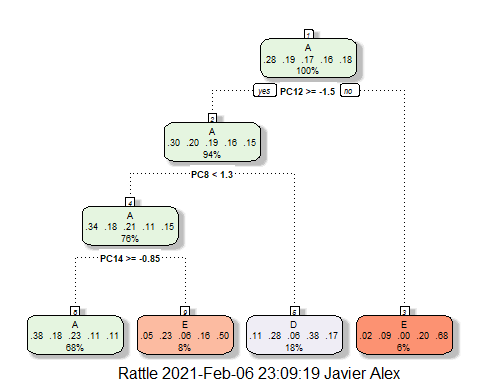
## Modeling

### Set up processing

library(doParallel)  
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS  
registerDoParallel(cluster)  
  
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)

### Predicting with Recursive Partitioning (Trees)

pmlModelrp <- train(classe~., data = pmlTraining, method="rpart", preProcess = "pca", na.action = na.omit, trControl = fitControl)  
pmlValrp <- predict(pmlModelrp, newdata=pmlValidation)  
pmllevels <- levels(factor(pmlValidation$classe))  
rpAccuracy <- confusionMatrix(factor(pmlValrp, levels = pmllevels),factor(pmlValidation$classe, levels = pmllevels))$overall['Accuracy']  
fancyRpartPlot(pmlModelrp$finalModel)



### Predicitng with Random Forests

pmlModelrf <- train(classe~., data = pmlTraining, method="rf", preProcess = "pca", na.action = na.omit, trControl = fitControl)  
pmlValrf <- predict(pmlModelrf, newdata=pmlValidation)  
rfAccuracy <- confusionMatrix(factor(pmlValrf, levels = pmllevels),factor(pmlValidation$classe, levels = pmllevels))$overall['Accuracy']

### Prediction with GBM (Gradient Boosting Machine) (Boosting with Trees)

pmlModelgbm <- train(classe~., data = pmlTraining, method="gbm", preProcess = "pca", na.action = na.omit, trControl = fitControl)

## Iter TrainDeviance ValidDeviance StepSize Improve  
## 1 1.6094 nan 0.1000 0.1267  
## 2 1.5315 nan 0.1000 0.0939  
## 3 1.4752 nan 0.1000 0.0709  
## 4 1.4313 nan 0.1000 0.0595  
## 5 1.3942 nan 0.1000 0.0493  
## 6 1.3617 nan 0.1000 0.0473  
## 7 1.3320 nan 0.1000 0.0372  
## 8 1.3062 nan 0.1000 0.0343  
## 9 1.2844 nan 0.1000 0.0329  
## 10 1.2629 nan 0.1000 0.0327  
## 20 1.1032 nan 0.1000 0.0170  
## 40 0.9288 nan 0.1000 0.0086  
## 60 0.8218 nan 0.1000 0.0055  
## 80 0.7424 nan 0.1000 0.0049  
## 100 0.6803 nan 0.1000 0.0021  
## 120 0.6257 nan 0.1000 0.0019  
## 140 0.5809 nan 0.1000 0.0014  
## 150 0.5615 nan 0.1000 0.0020

pmlValgbm <- predict(pmlModelgbm, newdata=pmlValidation)  
gbmAccuracy <- confusionMatrix(factor(pmlValgbm, levels = pmllevels),factor(pmlValidation$classe, levels = pmllevels))$overall['Accuracy']

### Prediction with Linear Discriminate Analysis

pmlModellda <- train(classe~., data = pmlTraining, method="lda", preProcess = "pca", na.action = na.omit, trControl = fitControl)  
pmlVallda <- predict(pmlModellda, newdata=pmlValidation)  
ldaAccuracy <- confusionMatrix(factor(pmlVallda, levels = pmllevels),factor(pmlValidation$classe, levels = pmllevels))$overall['Accuracy']

### Prediction with Naive Bayes

pmlModelnb <- train(classe~., data = pmlTraining, method="nb", preProcess = "pca", na.action = na.omit, trControl = fitControl)  
pmlValnb <- predict(pmlModelnb, newdata=pmlValidation)  
nbAccuracy <- confusionMatrix(factor(pmlValnb, levels = pmllevels),factor(pmlValidation$classe, levels = pmllevels))$overall['Accuracy']

# Results

The below table summarizes the results of running 5 different techniques to predict “classe”. Random Forest and Boosting with Trees offered the highest accuracy; where **0.9771615**, **0.8162724** are their accuracies respectably. Trees has highest accuracy, but it could be over fitting, therefore, it was selected Boosting with Trees to be used with test data set.

results <- matrix(c(rpAccuracy, rfAccuracy, gbmAccuracy, ldaAccuracy, nbAccuracy), nrow=5, ncol=1)  
row.names(results) <- c("Trees", "Random Forests", "Boosting with Trees", "Linear Discriminate Analysis", "Naive Bayes")  
resultstable <- as.table(results)  
colnames(resultstable)<-c("Accuracy")  
resultstable

## Accuracy  
## Trees 0.4023246  
## Random Forests 0.9771615  
## Boosting with Trees 0.8162724  
## Linear Discriminate Analysis 0.5320147  
## Naive Bayes 0.6358075

# Using Model with Test Dataset

The below table summarizes the predictions for each “problem ID” in the test data set

testresults <- predict(pmlModelgbm, newdata=pmlTesting)  
problemID <- as.data.frame(t(pmlTesting["problem\_id"]))  
testresultstable <- rbind(problemID, as.character(testresults) )  
row.names(testresultstable)<-c("problem\_ID", "Class")  
unname(testresultstable)

##   
## problem\_ID 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  
## Class C A C A A E D D A A A C B A E E A B B B