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
```

```r
dim(pmlTesting)
```

```
## [1] 20 53
```

Modeling

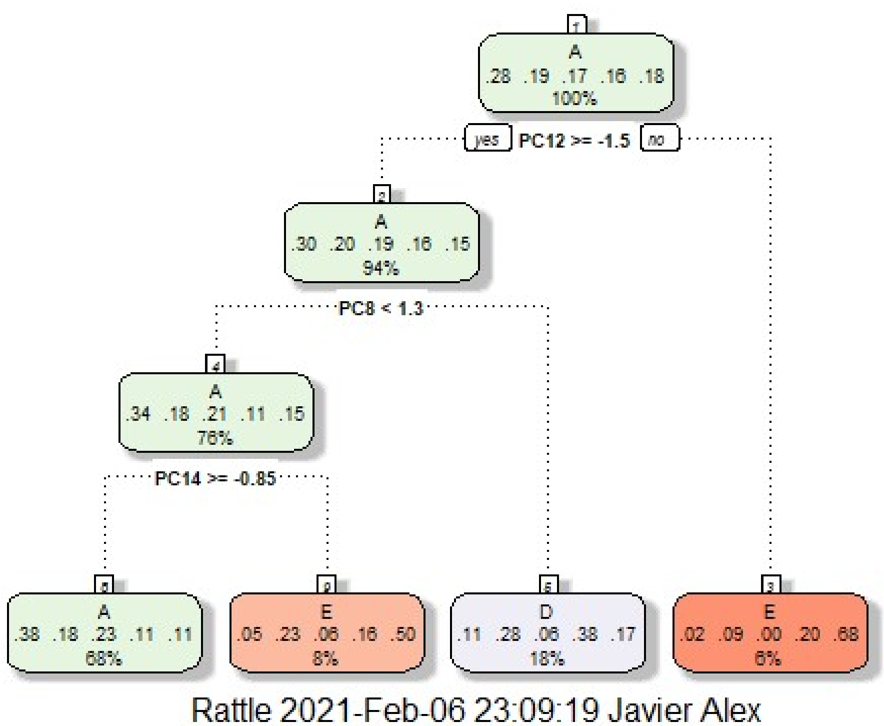
Set up processing

```r
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)

```r
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
```