Course Project - Practical Machine Learning

Cristian Pelayo

11/22/2020

This page is the submission for Coursera Pratical Machine Learning Course Project.

INTRODUCTION

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

he 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 data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What you should submit

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.

GETTING THE DATA

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

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

```r
library(rpart)
library(rpart.plot)
```

```r
url1 <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(url1, destfile = "pml-training.csv")
download.file(url2, destfile = "pml-testing.csv")

trainingDATA <- read_csv("pml-training.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   user_name = col_character(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   kurtosis_roll_belt = col_character(),
##   kurtosis_picth_belt = col_character(),
##   kurtosis_yaw_belt = col_character(),
##   skewness_roll_belt = col_character(),
##   skewness_roll_belt.1 = col_character(),
##   skewness_yaw_belt = col_character(),
##   max_yaw_belt = col_character(),
##   min_yaw_belt = col_character(),
##   amplitude_yaw_belt = col_character(),
```

```
##   kurtosis_picth_arm = col_character(),
##   kurtosis_yaw_arm = col_character(),
##   skewness_pitch_arm = col_character(),
##   skewness_yaw_arm = col_character(),
##   kurtosis_yaw_dumbbell = col_character(),
##   skewness_yaw_dumbbell = col_character(),
##   kurtosis_roll_forearm = col_character(),
##   kurtosis_picth_forearm = col_character()
##   # ... with 8 more columns
## )

## See spec(...) for full column specifications.

## Warning: 182 parsing failures.
##  row               col expected  actual              file
## 2231 kurtosis_roll_arm a double #DIV/0! 'pml-training.csv'
## 2231 skewness_roll_arm a double #DIV/0! 'pml-training.csv'
## 2255 kurtosis_roll_arm a double #DIV/0! 'pml-training.csv'
## 2255 skewness_roll_arm a double #DIV/0! 'pml-training.csv'
## 2282 kurtosis_roll_arm a double #DIV/0! 'pml-training.csv'
## .... ................. ........ ....... ..................
## See problems(...) for more details.
```

```
testingDATA <- read_csv("pml-testing.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   .default = col_logical(),
##   X1 = col_double(),
##   user_name = col_character(),
##   raw_timestamp_part_1 = col_double(),
##   raw_timestamp_part_2 = col_double(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   num_window = col_double(),
##   roll_belt = col_double(),
##   pitch_belt = col_double(),
##   yaw_belt = col_double(),
##   total_accel_belt = col_double(),
##   gyros_belt_x = col_double(),
##   gyros_belt_y = col_double(),
##   gyros_belt_z = col_double(),
##   accel_belt_x = col_double(),
##   accel_belt_y = col_double(),
##   accel_belt_z = col_double(),
##   magnet_belt_x = col_double(),
##   magnet_belt_y = col_double(),
##   magnet_belt_z = col_double()
##   # ... with 40 more columns
## )
## See spec(...) for full column specifications.
```

DATA CLEANING

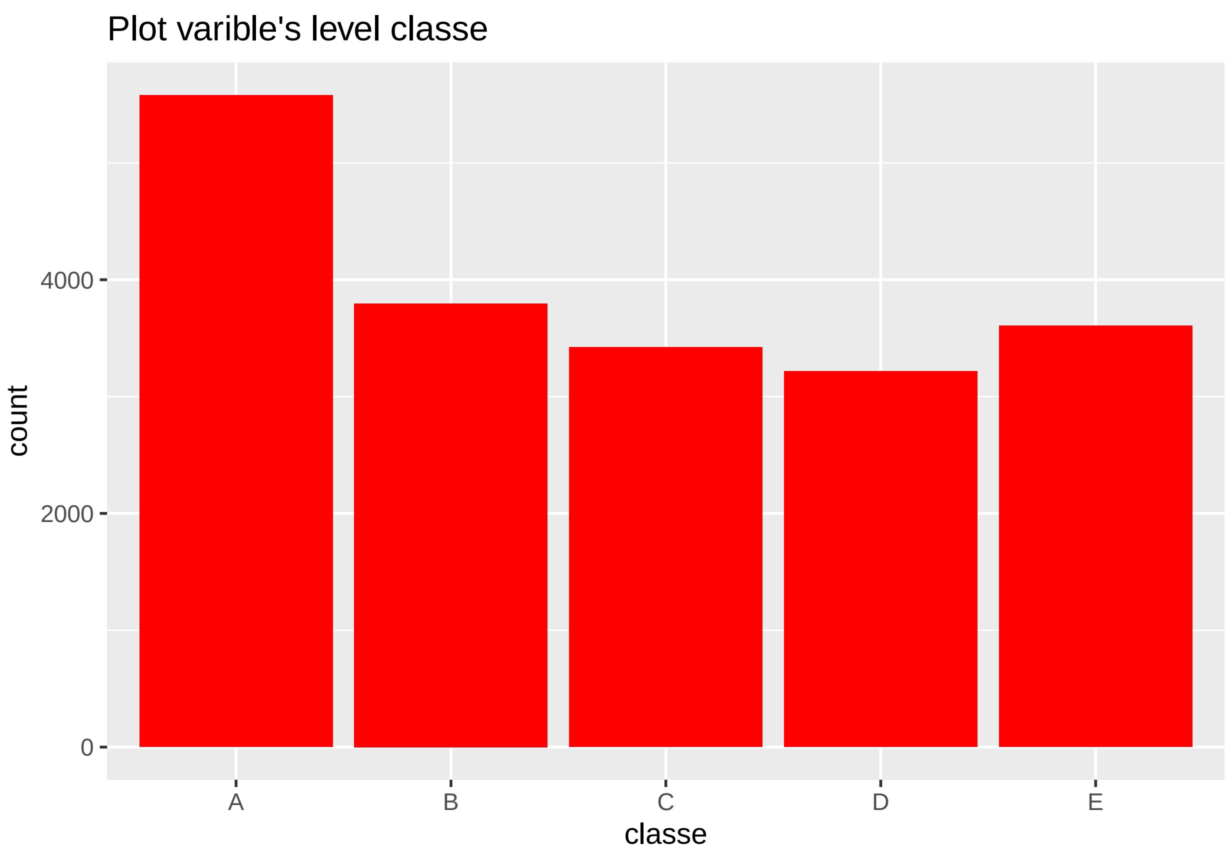
some varaibles do not have any value, then find which variables (if any) that are mostly na values

```
naprops <- colSums(is.na(trainingDATA))/nrow(trainingDATA)
mostlyNAs <- names(naprops[naprops > 0.90])
mostlyNACols <- which(naprops > 0.90)

training <- trainingDATA[,-mostlyNACols]
testing <- testingDATA[,-mostlyNACols]

training <- training[,-1]
testing <- testing[,-1]

training$classe <- as.factor(training$classe)
```

EXPLORATORY ANALYSIS

```
ggplot(data = training,mapping = aes(classe)) +
  geom_histogram(stat="count", fill = "red") +
  ggtitle("Plot varible's level classe")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



Based on the graph above, we can see that each level frequency is within the same order of magnitude of each other. Level A is the most frequent while level D is the least frequent.

PARTITION DATA

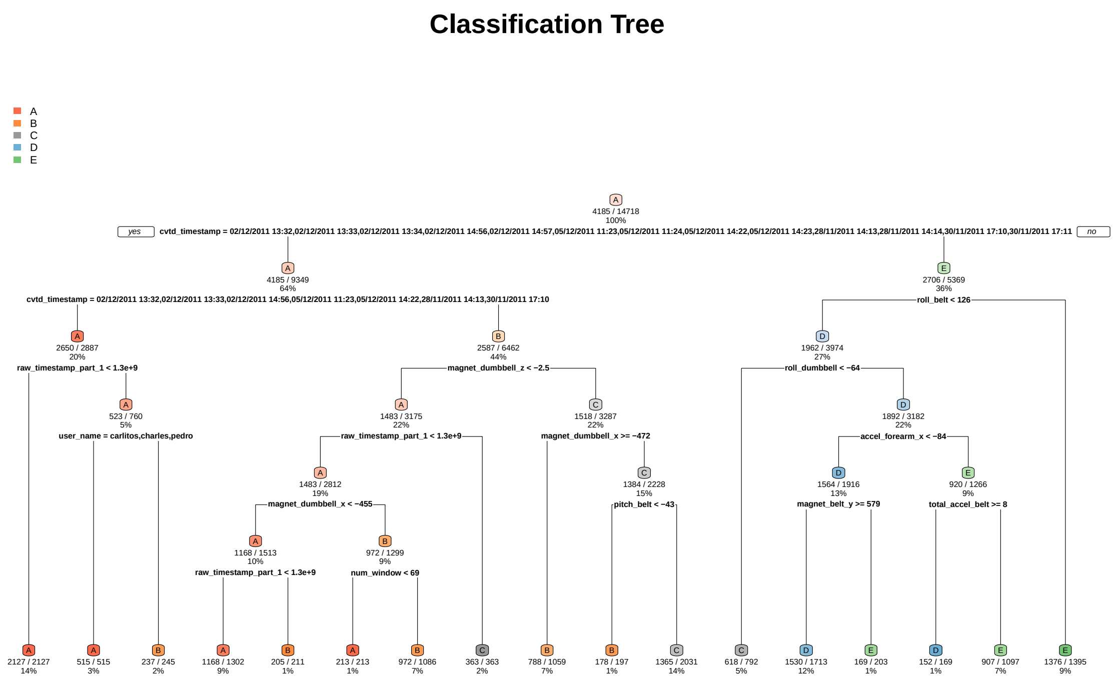
```
train <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
Trainingdf <- training[train, ]
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
Testingdf <- training[-train, ]
```

#PREDICTION MODEL 1. DECISION TREE

```
model1 <- rpart(classe ~ ., data=Trainingdf, method="class")
prediction1 <- predict(model1, Testingdf, type = "class")
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



Using confusion Matrix to test results:

```
confusionMatrix(prediction1, Testingdf$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1357   39    7    2    0
##          B   27  814   61   33    0
##          C   11   91  771  124   45
##          D    0    5   12  567   63
##          E    0    0    4   78  793
##
## Overall Statistics
##
##                Accuracy : 0.8772
```

```
##                  95% CI : (0.8677, 0.8863)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8446
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9728   0.8577   0.9018   0.7052   0.8801
## Specificity            0.9863   0.9694   0.9331   0.9805   0.9795
## Pos Pred Value         0.9658   0.8706   0.7399   0.8764   0.9063
## Neg Pred Value         0.9891   0.9660   0.9782   0.9443   0.9732
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2767   0.1660   0.1572   0.1156   0.1617
## Detection Prevalence   0.2865   0.1907   0.2125   0.1319   0.1784
## Balanced Accuracy      0.9795   0.9136   0.9174   0.8429   0.9298
```

#PREDICTION MODEL 12. RANDOM FOREST

```r
model2 <- randomForest(classe ~. , data=Trainingdf, method="class")
prediction2 <- predict(model2, Testingdf, type = "class")
confusionMatrix(prediction2, Testingdf$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  948    1    0    0
##          C    0    0  854    4    0
##          D    0    0    0  800    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9973, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9988   0.9950   1.0000
## Specificity            0.9997   0.9997   0.9990   1.0000   1.0000
## Pos Pred Value         0.9993   0.9989   0.9953   1.0000   1.0000
## Neg Pred Value         1.0000   0.9997   0.9998   0.9990   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
```

```
## Detection Rate         0.2845   0.1933   0.1741   0.1631   0.1837
## Detection Prevalence   0.2847   0.1935   0.1750   0.1631   0.1837
## Balanced Accuracy      0.9999   0.9993   0.9989   0.9975   1.0000
```

MODEL SELECTION

Random Forest algorithm performed better than Decision Trees. Accuracy for Random Forest model was 0.9988 (95% CI: (0.9973, 0.9996)) compared to Decision Tree model with 0.8787 (95% CI: (0.8692, 0.8877)). The Random Forests model is choosen. The expected out-of-sample error is estimated at 0.005, or 0.5%.

SUBMISSION

Here is the final outcome based on the Prediction Model 2 (Random Forest) applied against the Testing dataset

```
predictM2 <- predict(model2, testing, type="class")
predictM2
```

```
##  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
```