Prediction_Assignment_Writeup

Franklin Santos

8/31/2020

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).

Overview

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. This report describes how data was cleaned, how I split "pml-training.csv" into train set and test set, and some of models are investigated.

Data Processing and Results

1. Loading add-on package and set seed

```
set.seed(12345)
library(caret)
```

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

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

```
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:ggplot2':
##
##     margin
```

2. Download rawdata and submit_data

```
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
rawdata <- read.csv(url_train, na.strings = c("", "NA"))
url_submit <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
submit_data <- read.csv(url_submit, na.strings = c("", "NA"))
```

3. Cleaning data

We should delete the column that contains NA to avoid the error. In addition, in order to make accurate predictions, columns that is not related exercise must also be deleted. In particular “X”, “user_name”, “raw_timestamp_part_1”, “raw_timestamp_part_2”, “cvtd_timestamp”, “new_window”, “num_window” are deleted.

```
#Remove NA cols
colname <- colnames(rawdata)[!colSums(is.na(rawdata)) > 0]
colname
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [13] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [16] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [19] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [22] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [25] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [28] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [31] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [34] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [37] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [40] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [43] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [46] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [49] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [52] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [55] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [58] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

```
#Remove NA cols from submit data
colnamesub <- colnames(submit_data)[!colSums(is.na(rawdata)) > 0]
colnamesub
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [13] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [16] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [19] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [22] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [25] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [28] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [31] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [34] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [37] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [40] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [43] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [46] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [49] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [52] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [55] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [58] "magnet_forearm_y"     "magnet_forearm_z"     "problem_id"
```

```
#Slice data related with exercise
colname <- colname[8: length(colname)]
df_wo_NA <- rawdata[colname]

#Submit data related with exercise
colnamesub <- colnamesub[8: length(colnamesub)]
submit_NA <- submit_data[colnamesub]


#Check the colnames of df_wo_NA is in submit_data.
#The last colname is "classe"
is.element(colname, colnames(submit_data))
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [49]  TRUE  TRUE  TRUE  TRUE FALSE
```

```
df_wo_NA$classe <- factor(df_wo_NA$classe)
```

4. Split data into random train and test

```
inTrain = createDataPartition(df_wo_NA$classe, p = 3/4)[[1]]
training = df_wo_NA[ inTrain,]
testing = df_wo_NA[-inTrain,]

#Other option for model_rf
training.ids <- createDataPartition(df_wo_NA$classe, p = 0.7, list = FALSE)
```

5. Random Forest

It takes a very long time for training, but it has a high accuracy.

```
model_rf <- randomForest(x = df_wo_NA[training.ids, 1:52],
                         y = df_wo_NA[training.ids, 53],
                         ntree = 500,
                         keep.forest = TRUE)

pred_rf <- predict(model_rf, testing)
confusionMatrix(testing$classe, pred_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    0    0
##          D    0    0    3  800    1
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.9979, 0.9998)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.999
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9965   1.0000   0.9989
## Specificity            1.0000   1.0000   1.0000   0.9990   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   0.9950   1.0000
## Neg Pred Value         1.0000   1.0000   0.9993   1.0000   0.9998
## Prevalence             0.2845   0.1935   0.1750   0.1631   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1631   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      1.0000   1.0000   0.9983   0.9995   0.9994
```

6. Liner Discriminant Analysis

It takes a short time but poor accuracy.

```
model_lda <- train(classe ~ ., data = training, method = "lda")
pred_lda <- predict(model_lda, testing)
confusionMatrix(testing$classe, pred_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1161   34   96  100    4
##          B  144  600  115   46   44
##          C   84   78  558  120   15
##          D   38   38   94  605   29
##          E   35  143   73   96  554
##
## Overall Statistics
##
##                Accuracy : 0.7092
##                  95% CI : (0.6963, 0.7219)
##     No Information Rate : 0.2981
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.632
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7941   0.6719   0.5962   0.6256   0.8576
## Specificity            0.9320   0.9130   0.9252   0.9495   0.9185
## Pos Pred Value         0.8323   0.6322   0.6526   0.7525   0.6149
## Neg Pred Value         0.9142   0.9259   0.9066   0.9117   0.9770
## Prevalence             0.2981   0.1821   0.1909   0.1972   0.1317
```

```
## Detection Rate         0.2367   0.1223   0.1138   0.1234   0.1130
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.8631   0.7924   0.7607   0.7876   0.8880
```

7. Recursive Partitioning and Regression Trees

The results can be confirmed visually, but poor accuracy.

```
model_rpart <- train(classe ~ ., data = training, method = "rpart")
pred_rpart<- predict(model_rpart, testing)
confusionMatrix(testing$classe, pred_rpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1252   30   90    0   23
##          B  396  317  236    0    0
##          C  434   24  397    0    0
##          D  343  151  310    0    0
##          E  114  132  229    0  426
##
## Overall Statistics
##
##                Accuracy : 0.4878
##                  95% CI : (0.4737, 0.5019)
##     No Information Rate : 0.5177
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3306
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4931  0.48471  0.31458       NA  0.94878
## Specificity            0.9395  0.85129  0.87424   0.8361  0.89338
## Pos Pred Value         0.8975  0.33404  0.46433       NA  0.47281
## Neg Pred Value         0.6332  0.91479  0.78637       NA  0.99425
## Prevalence             0.5177  0.13336  0.25734   0.0000  0.09156
## Detection Rate         0.2553  0.06464  0.08095   0.0000  0.08687
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7163  0.66800  0.59441       NA  0.92108
```

```
library(rattle)
```

```
## Loading required package: tibble
```

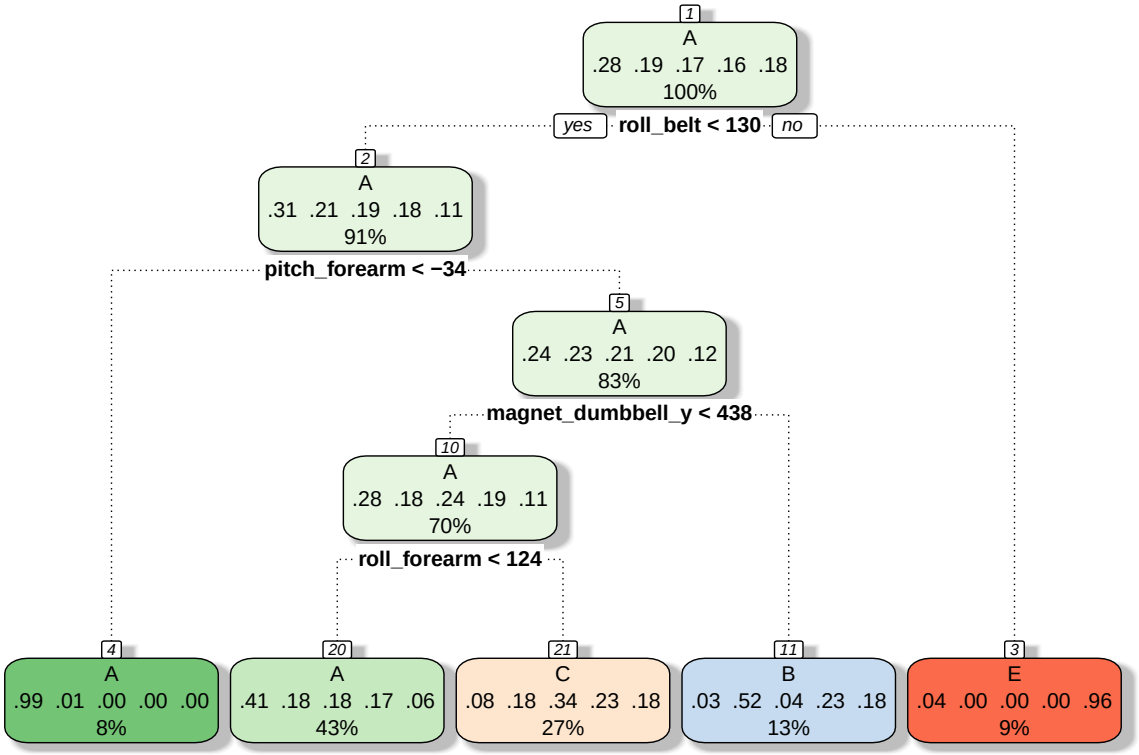
```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
fancyRpartPlot(model_rpart$finalModel)
```



8. Submit data with Random Forest

We can use the high accuracy model to submit data. In this report the Random Forest accuracy has the highest value 99.92. We can show the prediction.

```
submit_rf <- predict(model_rf, submit_NA)
submit_rf
```

```
##  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
```