Final Proyect Practical Machine Learning

Juan Andrade

10/4/2022

Review criteria

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.

Peer Review Portion Your submission for the Peer Review portion should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

Course Project Prediction Quiz Portion Apply your machine learning algorithm to the 20 test cases available in the test data above and submit your predictions in appropriate format to the Course Project Prediction Quiz for automated grading.

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

Prediction Assignment Writeup

Background

This is an R 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).

Data

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 data for this project come from this source: http://web.archive.org/web/20161224072740/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.

Data loading and analysis

As a first step, we upload the necessary libraries for the analysis

```
library(rpart)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.1.3

## Loading required package: tibble

## Warning: package 'tibble' was built under R version 4.1.1

## Loading required package: bitops

## Warning: package 'bitops' was built under R version 4.1.1

## Rattle: A free graphical interface for data science with R.
## Versión 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.1

## Loading required package: lattice
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.3

## randomForest 4.7-1
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

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

Then we upload our data to objects of the R tool to be able to carry out the modeling. In addition, we observe the dimensions of the same

```
Url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Url_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

Archivo_train <- "pml-trainging.csv"
Archivo_test <- "pml-testing.csv"

if (!file.exists(Archivo_train)){download.file(Url_train, Archivo_train, method = "curl")}
if (!file.exists(Archivo_test)){download.file(Url_test, Archivo_test, method = "curl")}

training <- read.csv(Archivo_train)
testing <- read.csv(Archivo_test)

dim(training)
```

```
## [1] 19622   160
```

```
dim(testing)
```

```
## [1]  20 160
```

We partition the data:

```
inTrain <- createDataPartition(training$classe, p = .7, list = FALSE)
data_train <- training[inTrain,]
data_test <- training[-inTrain,]
```

We carry out the preparation and cleaning of the data where the N/A fields are eliminated, the variables with variance 0 such as those of identity.

```
data_train <- data_train[, colSums(is.na(data_train)) == 0]
data_test <- data_test[, colSums(is.na(data_test)) == 0]

VarianceZero <- nearZeroVar(data_train)
```

```
data_train <- data_train[,-VarianceZero]
data_test <- data_test[,-VarianceZero]

data_train <- data_train[,-(1:5)]
data_test <- data_test[,-(1:5)]
```

The construction of the models will be carried out, in addition to the cross-validation.

###Random Forest:

```
Model_RandomForest <- train(classe ~., data = data_train, method = "rf",
                trControl = trainControl("cv", number = 5))
Model_RandomForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.27%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    4 2649    5    0    0 0.0033860045
## C    0    4 2392    0    0 0.0016694491
## D    0    0   14 2237    1 0.0066607460
## E    0    1    0    7 2517 0.0031683168
```

Prediction with Random Forest:

```
Predict_Model_RF <- predict(Model_RandomForest, data_test)
Matrix_RF <-confusionMatrix(Predict_Model_RF,as.factor(data_test$classe))
Accuracy_RF <- Matrix_RF$overall[1]
Error_RF <- 1 - Matrix_RF$overall[1]
```

The accuracy in this case is:

```
Accuracy_RF
```

```
##  Accuracy
## 0.9976211
```

And the error (1-accuracy) is:

```
Error_RF
```

```
##    Accuracy
## 0.002378929
```

Generalized Boosted Model

Then we are going to do the Generalized Boosted Model.

```
Model_GBM <- train(classe ~., data = data_train, method = "gbm", verbose = FALSE,
                 trControl = trainControl(method = "cv", number = 5))
Model_GBM
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10989, 10991
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7623205  0.6985452
##   1                  100      0.8340243  0.7898871
##   1                  150      0.8730436  0.8393063
##   2                   50      0.8894235  0.8599703
##   2                  100      0.9426371  0.9273993
##   2                  150      0.9636751  0.9540303
##   3                   50      0.9331740  0.9154108
##   3                  100      0.9717554  0.9642609
##   3                  150      0.9857317  0.9819511
##
## 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.
```

We get the prediction of the model

```
Predict_Model_GBM <- predict(Model_GBM, data_test)
Matrix_GBM <- confusionMatrix(Predict_Model_GBM,as.factor(data_test$classe))
Accuracy_GBM <- Matrix_GBM$overall[1]
Error_GBM <- 1 - Matrix_GBM$overall[1]
```

In the modelo GBM, the accuracy is:

```
Accuracy_GBM
```

```
##  Accuracy
## 0.9864061
```

And the error (1-accuracy) is :

```
Error_GBM
```

```
##   Accuracy
## 0.01359388
```

Decision Tree

Finally, we are going to make a Decision Tree model

```
Moldel_Des_Tree <- train(classe ~., data = data_train, method = "rpart")
Moldel_Des_Tree
```

```
## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03722917  0.5786964  0.46488658
##   0.04282372  0.5366525  0.40055353
##   0.11789238  0.3217855  0.05568058
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03722917.
```

We get the prediction of the model

```
Predict_Model_DT <- predict(Moldel_Des_Tree, data_test)
Matrix_DT <- confusionMatrix(Predict_Model_DT, as.factor(data_test$classe))
Accuracy_DT <- Matrix_DT$overall[1]
Error_DT <- 1 - Matrix_DT$overall[1]
```

In this case the accuracy is:

```
Accuracy_DT
```

```
##  Accuracy
## 0.5587086
```

And the error is:

```
Error_DT
```

```
##  Accuracy
## 0.4412914
```

Conclusions

Of the three models made, the one with the best accuracy and the least error is the Random Forest model, with which it is concluded that it is the best model (selected model).