How Well They Do It

Luis Fernando Agottani

12/09/2019

1. Summary

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.

2. 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.

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: 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).

The data for this project come from this source:

http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

3. The reference:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. “Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13)”. Stuttgart, Germany: ACM SIGCHI, 2013.

4. Data project

```
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(rpart.plot)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
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:rattle':
##
##     importance
```

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

```
library(RColorBrewer)

set.seed(2019) ##to make data reproducible
```

```
trainingfilename <- "pml-training.csv"

#File exists? -No - (Download)
if (!file.exists(trainingfilename)){
  trainingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(trainingURL, trainingfilename)
}
```

```
testingfilename <- "pml-testing.csv"

#File exists? -No - (Download)
if (!file.exists(testingfilename)){
  testingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(testingURL, testingfilename)
}
```

```
data_train<- read.csv(trainingfilename, strip.white=TRUE, na.strings=c("NA",""))
```

```
data_testing<- read.csv(testingfilename, strip.white=TRUE, na.strings=c("NA",""))
```

```
dim(data_train)
```

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

```
dim(data_testing)
```

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

Partition the training data in two parts (80% and 20%).

```
in_train  <- createDataPartition(data_train$classe, p=0.80, list=FALSE)
train_set <- data_train[ in_train, ]
test_set  <- data_train[-in_train, ]

dim(train_set)
```

```
## [1] 15699   160
```

```
dim(test_set)
```

```
## [1] 3923  160
```

Cleaning data to make it light to load removing columns with a high number of NAs and near zeros variance values.

```
nzv_var <- nearZeroVar(train_set)
train_set <- train_set[ , -nzv_var]
test_set  <- test_set [ , -nzv_var]

na_var <- sapply(train_set, function(x) mean(is.na(x))) > 0.95
train_set <- train_set[ , na_var == FALSE]
test_set  <- test_set [ , na_var == FALSE]
```

Remove the firsts 5 ID columns because it is useless for this analysis.

```
train_set <- train_set[ , -(1:5)]
test_set  <- test_set [ , -(1:5)]

dim(train_set)
```

```
## [1] 15699    54
```

```
dim(test_set)
```

```
## [1] 3923   54
```

5. Data Prediction and Modelling

Using Random Forest Model

```
set.seed(2019)
setting <- trainControl(method="cv", 5)
RMF <- train(classe ~ ., data=train_set, method="rf", trControl=setting, ntree=250)
RMF
```

```
## Random Forest
##
## 15699 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 12559, 12559, 12560, 12559, 12559
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9932479  0.9914581
##   27    0.9964965  0.9955681
##   53    0.9950952  0.9937952
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
predict_RMF <- predict(RMF, test_set)
confusionMatrix(test_set$classe, predict_RMF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    1    0    0    1
##          B    0  756    3    0    0
##          C    0    2  682    0    0
##          D    0    0    3  640    0
##          E    0    0    0    0  721
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9953, 0.9988)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9960   0.9913   1.0000   0.9986
## Specificity            0.9993   0.9991   0.9994   0.9991   1.0000
## Pos Pred Value         0.9982   0.9960   0.9971   0.9953   1.0000
## Neg Pred Value         1.0000   0.9991   0.9981   1.0000   0.9997
## Prevalence             0.2840   0.1935   0.1754   0.1631   0.1840
## Detection Rate         0.2840   0.1927   0.1738   0.1631   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9996   0.9975   0.9953   0.9995   0.9993
```

Using Generalized Boosted Model (GBM)

```
set.seed(2019)
setting_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
GBM <- train(classe ~ ., data = train_set, method = "gbm",
                  trControl = setting_GBM, verbose = FALSE)
GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predict_GBM <- predict(GBM, test_set)
confusionMatrix(test_set$classe, predict_GBM)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    2    0    0    0
##          B    5  747    6    1    0
##          C    0    4  676    4    0
##          D    1    3    5  633    1
##          E    0    4    2    8  707
##
## Overall Statistics
##
##                Accuracy : 0.9883
##                  95% CI : (0.9844, 0.9914)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9852
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9829   0.9811   0.9799   0.9986
## Specificity            0.9993   0.9962   0.9975   0.9969   0.9956
## Pos Pred Value         0.9982   0.9842   0.9883   0.9844   0.9806
## Neg Pred Value         0.9979   0.9959   0.9960   0.9960   0.9997
## Prevalence             0.2855   0.1937   0.1756   0.1647   0.1805
## Detection Rate         0.2840   0.1904   0.1723   0.1614   0.1802
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9970   0.9896   0.9893   0.9884   0.9971
```

The accuracy of the models:

Random Forest Model: 99.68 % (The best accuracy). Generalized Boosted Model: 98.83 %.

6. Applying Random Forest Model (The best accuracy) to the 20 data points of the test file.

```
predict_testing <- predict(RMF, newdata = data_testing)
predict_testing
```

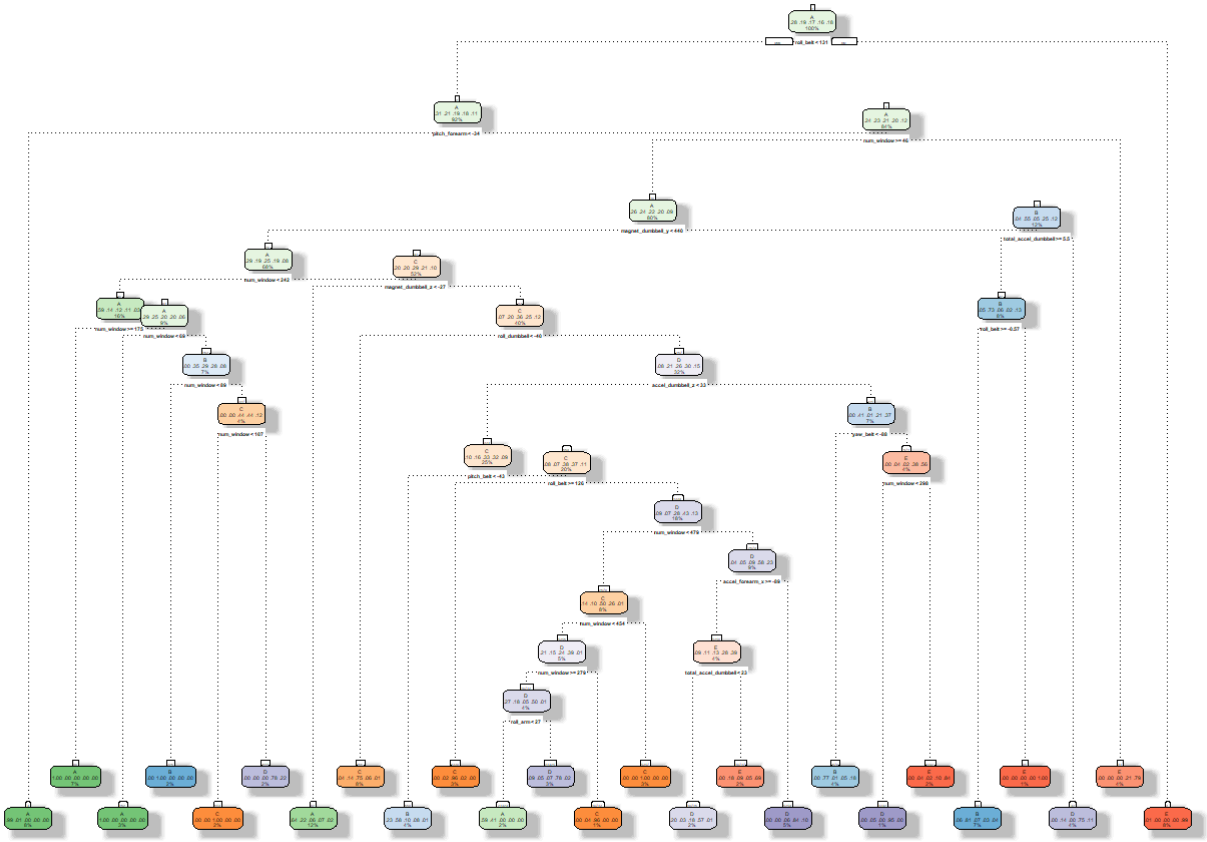
```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

7. Appendix

Decision tree model

```
set.seed(2019)
fit_decision_tree <- rpart(classe ~ ., data = train_set, method="class")
fancyRpartPlot(fit_decision_tree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2019-set-12 23:57:13 Luis