Practical Machine Learning - Prediction Assignment Writeup

Cyril Levasseur

2017-09-20

```r
knitr::opts_chunk$set(message = FALSE, warning = FALSE)
```

1. 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).

2. Goal

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 explains the model that is built, and used on a test set to predict 20 different cases.

3. Data loading

```r
library(caret)
library(rpart)
set.seed(52)

TrainUrl <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
TestUrl <- 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

DataTraining <- read.csv(file = TrainUrl)
DataTest <- read.csv(file = TestUrl)

dim(DataTraining)
```

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

```r
dim(DataTest)
```

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

Both sets have 160 variables.

4. Data partitioning

```r
InTrain <- createDataPartition(DataTraining$classe, p = 0.7, list = FALSE)
TrainSet <- DataTraining[InTrain, ]
TestSet  <- DataTraining[-InTrain, ]
dim(TrainSet)
```

```
## [1] 13737   160
```

```r
dim(TestSet)
```

```
## [1] 5885  160
```

5. Data cleaning

```r
str(TrainSet)
```

```
## 'data.frame':    13737 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 8 9 10 11 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 440390 484323 484434 5003
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 1.43 1.45 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.13 8.16 8.17 8.18 ...
##  $ 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      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 0.03 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 -0.02 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -20 -21 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 4 2 4 2 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 24 22 23 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -2 1 -3 -5 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 603 602 609 596 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -313 -312 -308 -317 ...
##  $ 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 21.8 21.7 21.6 21.5 ...
##  $ 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 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 -0.02 -0.02 0 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -288 -288 -290 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 109 110 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -124 -122 -124 -123 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -372 -369 -376 -366 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 338 341 334 339 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 510 518 516 509 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
# Removing of variables linked to the user
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]

# Replace #DIV/0! by NA's
TrainSet[TrainSet == "#DIV/0!"] <- NA
TestSet[TestSet == "#DIV/0!"] <- NA

# Removing of variables with lots of NA's
NaToRemove <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, NaToRemove==FALSE]
TestSet  <- TestSet[, NaToRemove==FALSE]
dim(TrainSet)
```

```
## [1] 13737    88
```

```
dim(TestSet)
```

```
## [1] 5885   88
```

```
# Removing variables with nearly Zero Variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TrainSet)
```

```
## [1] 13737    54
```

```
dim(TestSet)
```

```
## [1] 5885   54
```

==> We finally keep 53 variables for the modeling.

6. Prediction Model

We are going to use 2 different algorithms, and choose the one with the best result on the test set for the final prediction of the 20 cases :

1/ Random Forest 2/ Generalized Boosted Model

Random Forest

```
MyControl <- trainControl(method="cv", number=3, verboseIter=FALSE)
ModFitRF <- train(classe ~ .,
                  data = TrainSet,
                  method = "rf",
                  trControl = MyControl,
                  verbose = FALSE)
PredictRF <- predict(ModFitRF, newdata = TestSet)
confusionMatrix(PredictRF, TestSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1138    4    0    2
##          C    0    1 1022    5    0
##          D    0    0    0  958    0
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9978
##                  95% CI : (0.9962, 0.9988)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9972
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9961   0.9938   0.9982
## Specificity            1.0000   0.9987   0.9988   1.0000   0.9998
## Pos Pred Value         1.0000   0.9948   0.9942   1.0000   0.9991
## Neg Pred Value         1.0000   0.9998   0.9992   0.9988   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1737   0.1628   0.1835
## Detection Prevalence   0.2845   0.1944   0.1747   0.1628   0.1837
## Balanced Accuracy      1.0000   0.9989   0.9974   0.9969   0.9990
```

Generalized Boosted Model

```
ModFitGBM <- train(classe ~ .,
                  data = TrainSet,
                  method = "gbm",
                  trControl = MyControl,
                  verbose = FALSE)
PredictGBM <- predict(ModFitGBM, newdata = TestSet)
confusionMatrix(PredictGBM, TestSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668   11    0    0    0
##          B    6 1111    9   10    6
##          C    0   15 1009    9    1
##          D    0    2    8  945    7
##          E    0    0    0    0 1068
##
## Overall Statistics
##
##                Accuracy : 0.9857
##                  95% CI : (0.9824, 0.9886)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9819
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9754   0.9834   0.9803   0.9871
## Specificity            0.9974   0.9935   0.9949   0.9965   1.0000
## Pos Pred Value         0.9934   0.9729   0.9758   0.9823   1.0000
## Neg Pred Value         0.9986   0.9941   0.9965   0.9961   0.9971
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1888   0.1715   0.1606   0.1815
## Detection Prevalence   0.2853   0.1941   0.1757   0.1635   0.1815
## Balanced Accuracy      0.9969   0.9844   0.9891   0.9884   0.9935
```

Accuracy seems to be better with the model fitted with random forest algorithm -> that's the model we are going to use for the prediction of the 20 test cases.

7. Final test prediction

```
PredictTest <- predict(ModFitRF, newdata = DataTest)
PredictTest
```

```
##  [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
```