Machine Learning Project

Richard Verbrugge

Sunday, July 26, 2015

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(RCurl)
```

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

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. In this project, we will 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. The goal is to predict the manner in which they did the exercise. We will be using the classe variable in the training set and a number of other variables as predictors to build a model with. Cross validation will be used to test the model on an independent training set and to predict the out-of-sample error rate.

The data for this exercise is available at :

- Training : https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- Test : https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The project is split in the following sections : * Reading the data and loading the required packages * some exploratory analysis and data cleaning * creating the model through random forests * testing the model * conclusions

Reading the data, loading packages, and splitting the data in training and test

The data is downloaded and stored in the working directory from which it is read. Some exploratory analysis was done in excell and unix (using grep) to see what needs to be defined as NA values.

For training and test purposes 60% of the data is used for training.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(4428)
downloadCSVAndLoadTable <- function(fileURL, filename, dir, header = FALSE, skip=0) {
  dir
  filename
  if (file.exists(filename) == FALSE) {
    download.file(fileURL, destfile = filename, method="auto")
  }

  read.csv(paste(c(dir, "/", filename), collapse=''), sep=",", header = header, skip=skip, na.st
rings=c("NA","#DIV/0!",""))
}
```

```
testing_file_coursera <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training_file_coursera <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training_file_local <- "pml-training.csv"
testing_file_local <- "pml-testing.csv"
training_data<-downloadCSVAndLoadTable(training_file_coursera, training_file_local,
                                       dir=getwd(), header=TRUE)
testing_data <- downloadCSVAndLoadTable(testing_file_coursera, testing_file_local,
                                       dir=getwd(),header=TRUE)
```

```
# Split data set to training and test set
inTrain = createDataPartition(y=training_data$classe, p=0.6, list=FALSE)
training = training_data[inTrain,]
testing = training_data[-inTrain,]
```

Exploratoration and cleaning of data

Exploration of the data learns that there are columns with numerous NA values and the first 7 columns do not contain data that is related to activity measurement.

Columns with the most missing values are removed and the columns that do ot contain activity data as well. To determine what a good value is whether to remove NA values a count is done after which it was decided to take 11000 as the amount above which the column will be removed.

The new dataset contains 53 columns

there are 5 levels for the variable to be predicted which are A, B, C, D, E

```
# Calculate columns with the most missing values in them and remove those columns from the data
set
count_na = sapply(training, function(x) {sum(is.na(x))})
table(count_na)
```

```
## count_na
##     0 11539 11540 11541 11544 11545 11547 11556 11582 11583 11585 11587
##    60    68     2     4     1     4     2     2     1     1     2     1
## 11588 11776
##     6     6
```

```
remove_cols <- training[,colSums(is.na(training_data)) >= 11000]
training <- training[, !names(training) %in% names(remove_cols)]
training <- training[, -c(1:7)]
dim(training)
```

```
## [1] 11776    53
```

```
str(training)
```

```
## 'data.frame':    11776 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.48 1.43 1.45 1.45 1.42 1.42 1.45 1.48 1.55 ...
##  $ pitch_belt          : num  8.07 8.05 8.16 8.17 8.18 8.2 8.21 8.2 8.15 8.08 ...
##  $ 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 ...
##  $ gyros_belt_x        : num  0.02 0.02 0.02 0.03 0.03 0.02 0.02 0 0 0 ...
##  $ gyros_belt_y        : num  0 0 0 0 0 0 0 0 0 0.02 ...
##  $ gyros_belt_z        : num  -0.02 -0.03 -0.02 0 -0.02 0 -0.02 0 0 0 ...
##  $ accel_belt_x        : int  -22 -22 -20 -21 -21 -22 -22 -21 -21 -21 ...
##  $ accel_belt_y        : int  4 3 2 4 2 4 4 2 4 5 ...
##  $ accel_belt_z        : int  22 21 24 22 23 21 21 22 23 21 ...
##  $ magnet_belt_x       : int  -7 -6 1 -3 -5 -3 -8 -1 0 1 ...
##  $ magnet_belt_y       : int  608 604 602 609 596 606 598 597 592 600 ...
##  $ magnet_belt_z       : int  -311 -310 -312 -308 -317 -309 -310 -310 -305 -316 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -129 -129 -129 ...
##  $ pitch_arm           : num  22.5 22.1 21.7 21.6 21.5 21.4 21.4 21.4 21.3 21.2 ...
##  $ 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 ...
##  $ gyros_arm_x         : num  0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 0 0 0 -0.02 ...
##  $ gyros_arm_z         : num  -0.02 0.02 -0.02 -0.02 0 -0.02 -0.03 -0.03 -0.03 -0.03 ...
##  $ accel_arm_x         : int  -290 -289 -288 -288 -290 -287 -288 -289 -289 -288 ...
##  $ accel_arm_y         : int  110 111 109 110 110 111 111 111 109 108 ...
##  $ accel_arm_z         : int  -125 -123 -122 -124 -123 -124 -124 -124 -121 -124 ...
##  $ magnet_arm_x        : int  -369 -372 -369 -376 -366 -372 -371 -374 -367 -373 ...
##  $ magnet_arm_y        : int  337 344 341 334 339 338 331 342 340 336 ...
##  $ magnet_arm_z        : int  513 512 518 516 509 509 523 510 509 510 ...
##  $ roll_dumbbell       : num  13.1 13.4 13.2 13.3 13.1 ...
##  $ pitch_dumbbell      : num  -70.6 -70.4 -70.4 -70.9 -70.6 ...
##  $ yaw_dumbbell        : num  -84.7 -84.9 -84.9 -84.4 -84.7 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 36 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0.02 0 0 0.02 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 -0.02 0 0 0 -0.02 -0.02 0 0 -0.02 ...
##  $ accel_dumbbell_x    : int  -233 -232 -232 -235 -233 -234 -234 -234 -233 -231 ...
##  $ accel_dumbbell_y    : int  47 48 47 48 47 48 48 47 48 47 ...
##  $ accel_dumbbell_z    : int  -269 -269 -269 -270 -269 -269 -268 -270 -271 -268 ...
##  $ magnet_dumbbell_x   : int  -555 -552 -549 -558 -564 -552 -554 -554 -554 -557 ...
##  $ magnet_dumbbell_y   : int  296 303 292 291 299 302 295 294 297 292 ...
##  $ magnet_dumbbell_z   : num  -64 -60 -65 -69 -64 -69 -68 -63 -73 -62 ...
##  $ roll_forearm        : num  28.3 28.1 27.7 27.7 27.6 27.2 27.2 27.2 27.1 27 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.8 -63.8 -63.8 -63.9 -63.9 -63.9 -64 -64 ...
##  $ yaw_forearm         : num  -153 -152 -152 -152 -152 -151 -151 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.02 0.02 0.03 0.02 0.02 0 0 0 0.02 0.02 ...
##  $ gyros_forearm_y     : num  0 -0.02 0 0 -0.02 0 -0.02 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 0 -0.02 -0.02 -0.02 -0.03 -0.03 -0.02 0 -0.02 ...
##  $ accel_forearm_x     : int  192 189 193 190 193 193 193 192 194 192 ...
##  $ accel_forearm_y     : int  203 206 204 205 205 205 202 201 204 206 ...
##  $ accel_forearm_z     : int  -216 -214 -214 -215 -214 -215 -214 -214 -215 -216 ...
##  $ magnet_forearm_x    : int  -18 -16 -16 -22 -17 -15 -14 -16 -13 -16 ...
```

```
##  $ magnet_forearm_y    : num  661 658 653 656 657 655 659 656 656 653 ...
##  $ magnet_forearm_z    : num  473 469 476 473 465 472 478 472 471 472 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 ...
```

```
levels(training$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

Building the model and testing it against the test data

A prediction model is build on the training data using the randomForest method and is test against the test dataset. A confusion matrix and overall statistics are produced.

```
model <- randomForest(classe~., data=training)
prediction <- predict(model,testing)
cm <- confusionMatrix(prediction, testing$classe)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    9    0    0    0
##          B    1 1507    8    0    0
##          C    0    2 1359   29    3
##          D    1    0    1 1257    2
##          E    1    0    0    0 1437
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9906, 0.9945)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9928   0.9934   0.9774   0.9965
## Specificity            0.9984   0.9986   0.9948   0.9994   0.9998
## Pos Pred Value         0.9960   0.9941   0.9756   0.9968   0.9993
## Neg Pred Value         0.9995   0.9983   0.9986   0.9956   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1921   0.1732   0.1602   0.1832
## Detection Prevalence   0.2852   0.1932   0.1775   0.1607   0.1833
## Balanced Accuracy      0.9985   0.9957   0.9941   0.9884   0.9982
```

The overall quality of the model seems to work quite well on the test data with and overall accuracy of 99.43%. The accuracy whithin each class is very close to the overall avarage. It means an avarage out of sample error rate of 0.57% which is low. The specifity has a simular value. The overall conclusion is that the model created if of high quality.

Conclusion

The machine algolrithm build has a high level of predictability with a high value for accuracy, and as a consequnce a very low value for the out of sample error rate. the values are :

- Accuracy : 99.27 %
- out of sample error rate 0,73%

A visual inspection of the other statistics do not raise any concerns around specifity or confidence intervals.