Machine Learning Course Project: Predicting the manner of doing excercise based on parametric accelerometers data

Andrey

23 апреля 2016 г.

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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Source 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://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.

Purpose of the project / outcome

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.

Loading the data

```
library(caret)

## Warning: package 'caret' was built under R version 3.2.4

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.4

library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.2.4

library(RColorBrewer)
library(rattle)

## Warning: package 'rattle' was built under R version 3.2.4

## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.2.4

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## Следующий объект скрыт от 'package:ggplot2':
##
##     margin

set.seed(12345)
```

Whebn we first take a look at the data (e.g. downloading from the link above and oping in XL spreadsheet) we observe that

```
The header contains the column names
There are many missing values, which can be split into 3 types: NA, "","#DIV/0!"
The first column is not a variable, it counts the row number.

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Checking dim of 2 datasets, we conclude that both have 160 variables and different number of observatrions

```
dim(training); dim(testing)
```

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

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

Looking at the variable CLASSE which we need to predict, we conclude that this is a "factor"-class variable which can take one of 5 meanings:

```
## [1] "factor"
```

```
##   Var1 Freq
## 1    A 5580
## 2    B 3797
## 3    C 3422
## 4    D 3216
## 5    E 3607
```

CROSS-VALIDATION strategy and model selection strategy

Cross-validation STRATEGY: We'll split available Training dataset into 2 subsets: training1 (65%) and testing1 (35%).

We'll use 2 methods (Desision Trees and Random Forests) to train them on training1 sample and predict on testing 1 sample

We'll compare the models for accuracy/ out of sample error (received by predicting on testing1) and we'll choose the best model based out of sample error

We'll use te best model to predict on the original (testing) dataset which has 20 observations in it to predict the CLASSE variable

```
train_partition <- createDataPartition(y=training$classe, p=0.65, list=FALSE)
training1 <- training[train_partition, ]
testing1 <- training[-train_partition, ]
dim(training1); dim(testing1)
```

```
## [1] 12757   160
```

```
## [1] 6865  160
```

We want to identify and remove all variables which have NearZeroVariance

```
nzv <- nearZeroVar(training1, saveMetrics=TRUE)
training1 <- training1[,nzv$nzv==FALSE]
nzv<- nearZeroVar(testing1,saveMetrics=TRUE)
testing1 <- testing1[,nzv$nzv==FALSE]
```

We want to remove the first column, which is simply the order of record but not a variable

```
training1 <- training1[c(-1)]
```

Now we identify vatiables which have mostly NA. Those columns where NA are more than 60% will be removed

```
training1_temp <- training1
for(i in 1:length(training1)) {
        if( sum( is.na( training1[, i] ) ) /nrow(training1) >= .6 ) {
        for(j in 1:length(training1_temp)) {
            if( length( grep(names(training1[i]), names(training1_temp)[j]) )
==1)  {
                training1_temp <- training1_temp[ , -j]
            }
        }
    }
}
```

After checking how many columns / variables left after data cleaning, we conclude that their number reduced from 160 to 58

```
dim(training1_temp)
```

```
## [1] 12757    58
```

resuming to our training subset

```
training1 <- training1_temp
rm(training1_temp)
```

We need to repaet the same data cleaining for testing1 subset and for original testing dataset

```
clean1 <- colnames(training1)
clean2 <- colnames(training1[, -58])
testing1 <- testing1[clean1]
testing <- testing[clean2]
```

Making sure the number of columns has reduced acordingly

```
dim(testing1); dim(testing)
```

```
## [1] 6865   58
```

```
## [1] 20 57
```

We want to coerce the data into the same type

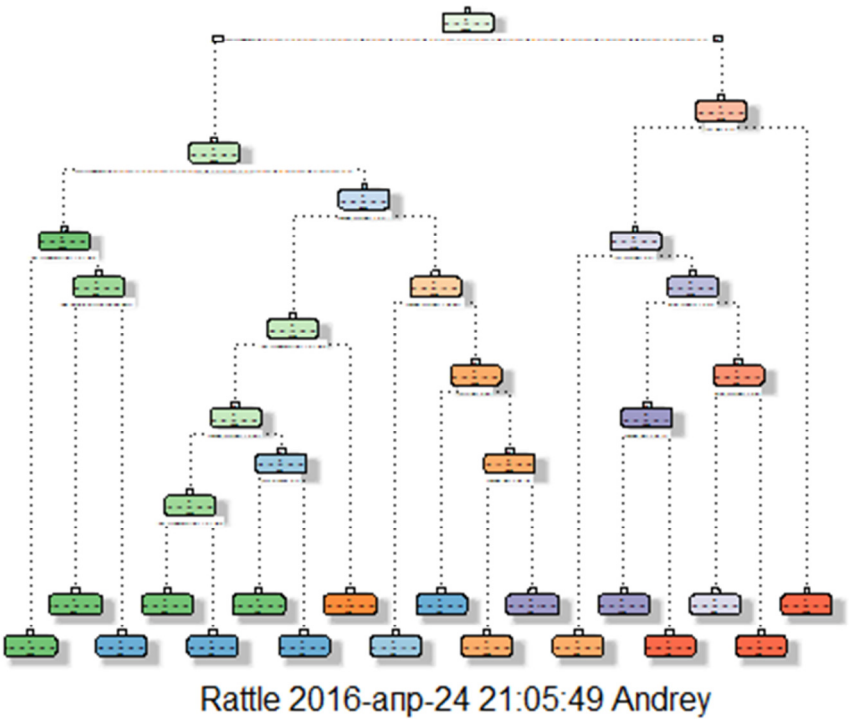
```
for (i in 1:length(testing) ) {
        for(j in 1:length(training1)) {
        if( length( grep(names(training1[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(training1[i])
        }
    }
}

testing <- rbind(training1[2, -58] , testing)
testing <- testing[-1,]
```

Predicion using Decision Trees

```
modFitA1 <- rpart(classe ~ ., data=training1, method="class")

fancyRpartPlot(modFitA1)
```



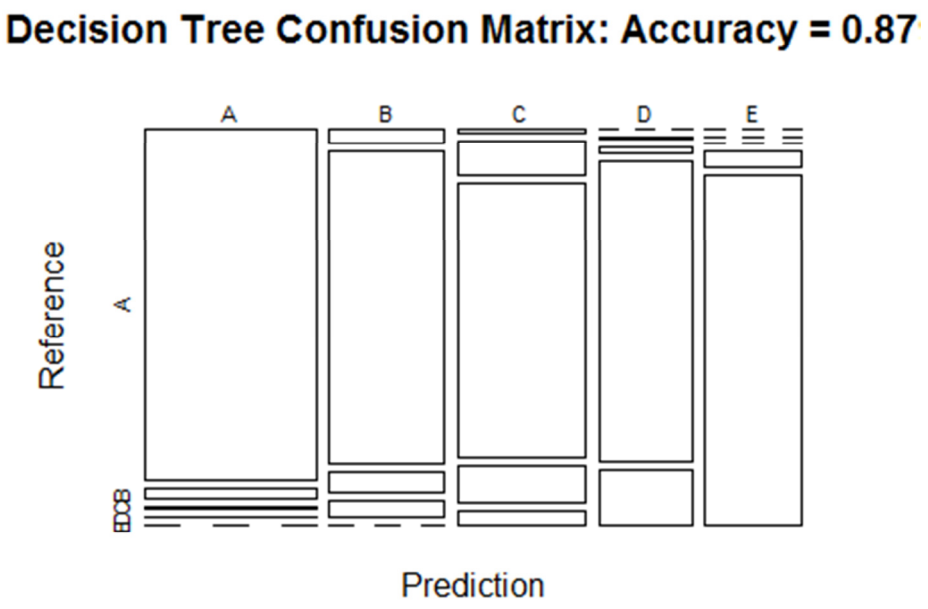
Predicting and checking accuracy using confusion matrix

```
predictionsA1 <- predict(modFitA1, testing1, type = "class")
confmatrix_tree <- confusionMatrix(predictionsA1, testing1$classe)
confmatrix_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1886   55    8    2    0
##          B   51 1124   71   57    0
##          C   16  141 1099  146   51
##          D    0    8   19  873  159
##          E    0    0    0   47 1052
##
## Overall Statistics
##
##                Accuracy : 0.879
##                  95% CI : (0.871, 0.8866)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8469
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9657   0.8464   0.9181   0.7760   0.8336
## Specificity            0.9868   0.9677   0.9375   0.9676   0.9916
## Pos Pred Value         0.9667   0.8626   0.7564   0.8244   0.9572
## Neg Pred Value         0.9864   0.9633   0.9819   0.9566   0.9636
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2747   0.1637   0.1601   0.1272   0.1532
## Detection Prevalence   0.2842   0.1898   0.2117   0.1543   0.1601
## Balanced Accuracy      0.9762   0.9070   0.9278   0.8718   0.9126
```

```
plot(confmatrix_tree$table, col = confmatrix_tree$byClass, main =
paste("Decision Tree Confusion Matrix: Accuracy =",
round(confmatrix_tree$overall['Accuracy'], 4)))
```



Accuracy for the Decision trees methid for dataset testing1 is 0.879

The expected OUT OF SAMPLE ERROR (for testing1 dataset) for the Decision Trees method is 100-87.9 = 12.1%

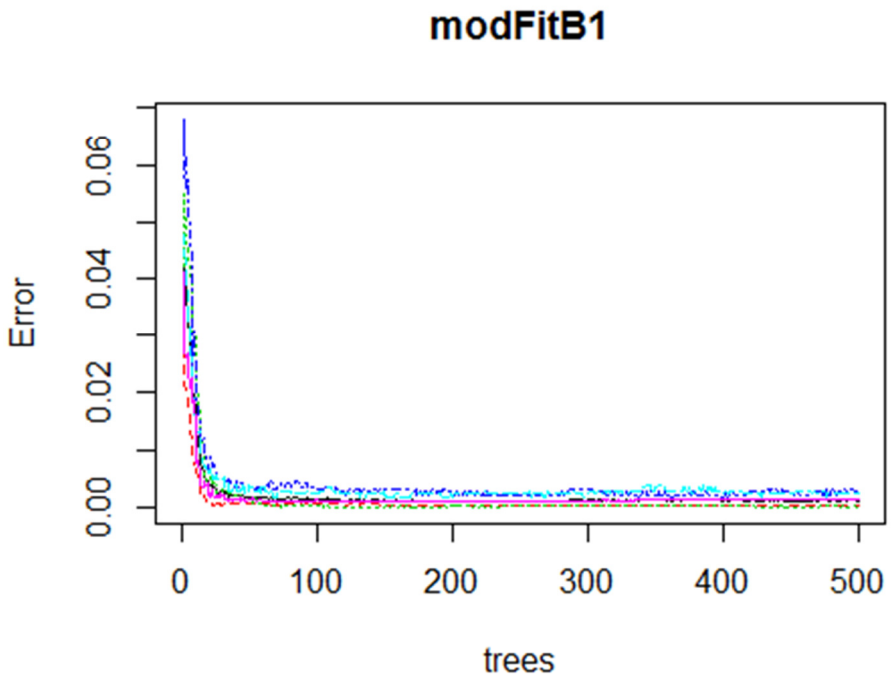
Predictions using Random Forests

```
modFitB1 <- randomForest(classe ~. , data=training1)
predictionsB1 <- predict(modFitB1, testing1, type = "class")
```

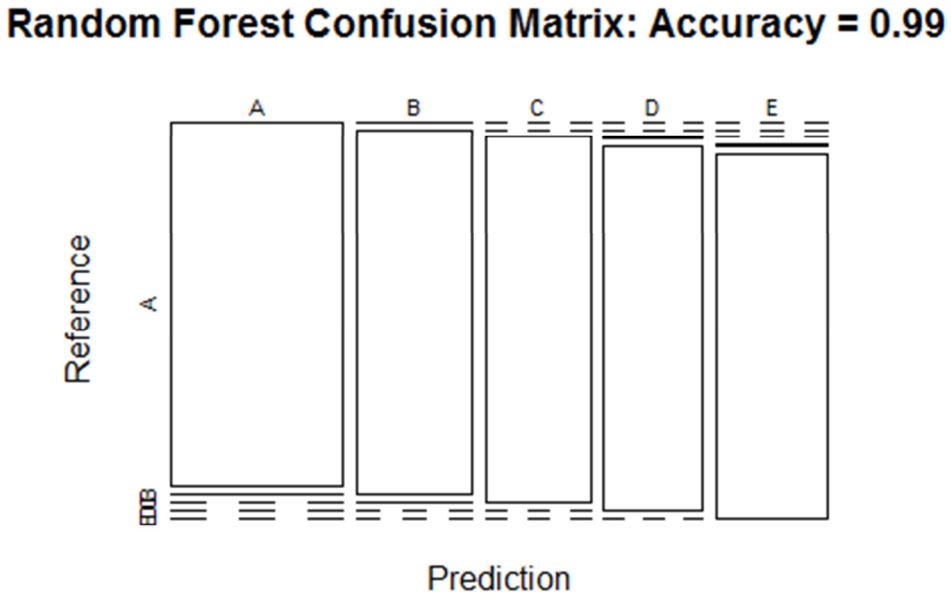
```
confmatrix_randfor <- confusionMatrix(predictionsB1, testing1$classe)
confmatrix_randfor
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1952    2    0    0    0
##          B    1 1326    1    0    0
##          C    0    0 1194    0    0
##          D    0    0    2 1122    0
##          E    0    0    0    3 1262
##
## Overall Statistics
##
##                Accuracy : 0.9987
##                  95% CI : (0.9975, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9983
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9995   0.9985   0.9975   0.9973   1.0000
## Specificity            0.9996   0.9996   1.0000   0.9997   0.9995
## Pos Pred Value         0.9990   0.9985   1.0000   0.9982   0.9976
## Neg Pred Value         0.9998   0.9996   0.9995   0.9995   1.0000
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1739   0.1634   0.1838
## Detection Prevalence   0.2846   0.1934   0.1739   0.1637   0.1843
## Balanced Accuracy      0.9995   0.9991   0.9987   0.9985   0.9997
```

```
plot(modFitB1)
```




```
plot(confmatrix_randfor$table, col = confmatrix_tree$byClass, main =
paste("Random Forest Confusion Matrix: Accuracy =",
round(confmatrix_randfor$overall['Accuracy'], 4)))
```



Accuracy for the Random forests methid for dataset testing1 is 0.879

The expected OUT OF SAMPLE ERROR (for testing1 dataset) for the random forests method is 100-99.87 = 0.13%

Expectation for out of sample error / Predicting of CLASSE variable for the data in the Test dataset:

Random Forests gave a better accuracy on the testing1 dataset of 99.87% (compared to 86.6% for Decision Trees), hence OUT OF SAMPLE ERROR is smaller for Random Forests method (0.13% vs 12.1%)

so we'll select Random Forests method to predict CLASSE variable for the Test sample.

Since we letft (testing) dataset apart when training the model, we expect (testing) dataset to have unbiased estimate of RF prediction. Based on small sample size of 20 measurements in (testing) dataset we expect out of sample error to be the same as for (testing1) sample, i e 0.13%. We expect NO ERRORS when predicting on small sample of 20 measurements (20 x 0.0013 = 0.026 < 1 => so we expect all variables predicted accurately with Fandom forests)

```
predictionsB2 <- predict(modFitB1, testing, type = "class")
predictionsB2
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```