Use of accelerometers to predict personal activity

Hernando Consuegra A.

10/6/2020

1. Abstract

1.1 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, the 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).

1.2. 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.

1.3. Objective The goal of the project is to predict the manner in which the participants did the exercise -this is the "classe" variable in the training set- using any of the other variables to predict with. This report describes how the model was built, how cross validation was used, what was the expected out of sample error, and why the choices were done. Finally, the prediction model build is used to predict 20 different test cases.

2. Read and transform training data

After reading the data from the web, the following transformations are done:

- Discard the first 7 variables (columns) as they contain identification values for the data record: *X*, *user_name*, *raw_timestamp_part_1*, *raw_timestamp_part_2*, *cvtd_timestamp*, *new_window*, and *num_window*.
- Define *classe* as factor.
- Discard the remaining predictors with proportion of NAs greater that 90%.

```r
library(caret)
library(rpart)
library(randomForest)
library(readr)

# Read training data
pml.link <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
pml.data <- read.csv(pml.link, na.strings=c("NA","#DIV/0!", "", " "))

# Discard non relevant predictors for model building and define 'classe' as factor
m <- ncol(pml.data)
pml.data <- pml.data[ , 8:m]
pml.data$classe <- as.factor(pml.data$classe)

# Use the predictors with the proportion of NAs smaller that 90%
n <- nrow(pml.data)
pml.data <-pml.data[,colSums(is.na(pml.data))/n <= .90]
```

3. Data partition for Cross validation

The data is now partitioned into 2 data sets, one for training (70%) and one for testing (30%).

```r
set.seed(1234)
inTrain <- createDataPartition(pml.data$classe, p = 0.7)[[1]]

pml.train <- pml.data[inTrain, ]
pml.test  <- pml.data[-inTrain, ]
```

4. Model selection

4.1 Decision tree The first model to explore is based on decision tree built with the function *rpart*. The *train* function of the *caret* package was tried but it required the use of PCA for predictors reduction and parallel processing in order to get reasonable response times. The results of these trials are not included to keep the length of the report between the limits. However, the results are very similar. For detail about *caret* performance, see Caret Performance Analysis by Len Greski.

```r
modelFit.tree <- rpart(classe ~ ., data=pml.train, method="class")
pred.tree     <- predict(modelFit.tree, pml.test, type = "class")
accurac.tree  <- confusionMatrix(pred.tree, pml.test$classe)$overall[1]
accurac.tree
```

```
##  Accuracy
## 0.7541206
```

The **accuracy of the decision tree model** on the testing partition is **0.7541206**.

4.2 Random Forests The second model to explore is based on random forest built with the function *randomForest*. The same performance consideration explained above apply in this case.

```r
modelFit.forest <- randomForest(classe ~ ., data=pml.train, method="class")
pred.forest     <- predict(modelFit.forest, pml.test, type = "class")
accurac.forest  <- confusionMatrix(pred.forest, pml.test$classe)$overall[1]
accurac.forest
```

```
## Accuracy
## 0.995582
```

The **accuracy of the random forest model** on the testing partition is **0.995582**.

4.3 Selection based on accuracy Based on the accuracy of the models on the *training* partition, the **random forest model is selected**.

4.4 Expected out of sample error The expected out of sample error for the **random forest model** is equal to 1 minus its accuracy over the *testing* partition: **0.004418**.

5. Prediction based on the downloaded test data

The selected model is used to predict the *classe* for the 20 samples in the downloaded testing data set.

```r
pml.link    <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
pml.testing <- read.csv(pml.link)

pred.testing.forest <- predict(modelFit.forest, pml.testing)
pred.testing.forest
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

6. Appendix 1 - Complete output of random forest model

```r
modelFit.forest
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = pml.train, method = "class")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.52%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   10 2643    5    0    0 0.005643341
## C    0   15 2377    4    0 0.007929883
## D    1    0   24 2225    2 0.011989343
## E    0    0    1    6 2518 0.002772277
```

7. Appendix 2 - Confusion Matrix for random forest model on training set

```r
confusionMatrix(pred.forest, pml.test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    2    0    0    0
##          B    1 1133   11    0    0
##          C    0    4 1014    6    0
##          D    0    0    1  957    0
##          E    0    0    0    1 1082
## 
## Overall Statistics
## 
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9944
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9947   0.9883   0.9927   1.0000
## Specificity            0.9995   0.9975   0.9979   0.9998   0.9998
## Pos Pred Value         0.9988   0.9895   0.9902   0.9990   0.9991
## Neg Pred Value         0.9998   0.9987   0.9975   0.9986   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1925   0.1723   0.1626   0.1839
## Detection Prevalence   0.2846   0.1946   0.1740   0.1628   0.1840
## Balanced Accuracy      0.9995   0.9961   0.9931   0.9963   0.9999
```