Practical Machine Learning, Johns Hopkins University - Writeup

Andrea Berardo

Friday, May 22, 2015

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, data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants will be used. 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). The goal of this project is to predict the manner in which participants did the exercise.

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

Data Processing

The datasets are stored in a comma-separated-value (CSV). The dataset files need to be placed in the working directory, before being loaded.

The code below loads the datasets and transforms them replacing all empty cells and `#DIV/0!` with `NA`:

```r
TrainData <- read.csv("pml-training.csv", na.strings = c("#DIV/0!", ""))
TestData <- read.csv("pml-testing.csv", na.strings = c("#DIV/0!", ""))

dim(TrainData); dim(TestData)
```

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

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

In order to further process the data and keep only the potential predictors, the training and test dataset are further transformed removing the first 7 information columns (i.e. X, user_name, raw_timestamp_part_1, . . . , num_window) and removing all variables where the number of NA's is large (typically over 90% of the observations).

```
Train <- TrainData[, -c(1:7)]
colPr <- colSums(Train == "NA" | is.na(Train)) < 2000
Train <- Train[, colPr]

FinalTest <- TestData[, -c(1:7)]
FinalTest <- FinalTest[, colPr]

dim(Train); dim(FinalTest)
```

```
## [1] 19622    53
```

```
## [1] 20 53
```

The variables are now reduced from 160 to 53.

The **Train** dataset is now partitioned in two portions (85% and 15%), one for training purposes and one for testing purposes of the machine learning algorithm. Note that the `caret` package is loaded for this purpose. The Seed is set to an arbitrary value to have reproducible results.

```
library(caret)

set.seed(100)
inTrain <- createDataPartition(y=Train$classe, p=0.85, list=FALSE)
training <- Train[inTrain, ]
testing <- Train[-inTrain, ]
dim(training); dim(testing)
```

```
## [1] 16680    53
```

```
## [1] 2942   53
```

Model Selection and Training

We'll use the Random Forest machine learning algorithm, with a number of trees to grow limited to 100 in order to have a relatively high processing time, while maintaining a high level of accuracy.
For this selection a Cross-Validation does not seem to bring improvement in accuracy.

The code below builds the model and verify its accuracy on the **training** data set itself. In this case the accuracy is 100% resulting in a perfect match of the model with the training dataset. The library `randomForest` needs to be loaded prior to fitting the model.

```
library(randomForest)
modelFit <- randomForest(classe ~ ., data=training, importance = TRUE, ntree=100)
confusionMatrix(training$classe,predict(modelFit,training))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4743    0    0    0    0
##          B    0 3228    0    0    0
##          C    0    0 2909    0    0
##          D    0    0    0 2734    0
##          E    0    0    0    0 3066
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The validated model is now tested on the **testing** partition of the training dataset, achieving accuracy above 99.5%.

```
cf <- confusionMatrix(testing$classe,predict(modelFit,testing))
cf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 837   0   0   0   0
##          B   0 569   0   0   0
##          C   0   2 511   0   0
##          D   0   0   6 476   0
##          E   0   0   0   3 538
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9933, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9884   0.9937   1.0000
## Specificity            1.0000   1.0000   0.9992   0.9976   0.9988
## Pos Pred Value         1.0000   1.0000   0.9961   0.9876   0.9945
## Neg Pred Value         1.0000   0.9992   0.9975   0.9988   1.0000
## Prevalence             0.2845   0.1941   0.1757   0.1628   0.1829
## Detection Rate         0.2845   0.1934   0.1737   0.1618   0.1829
## Detection Prevalence   0.2845   0.1934   0.1744   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9982   0.9938   0.9957   0.9994
```

The out of sample error can be evaluated as (1 - Testing Accuracy) and is as shown below:

```
print(paste("Out of Sample Error: ", format(round((1-cf$overall[1]) * 100, 2), nsmall = 2), "%"))
```

```
## [1] "Out of Sample Error:  0.37 %"
```

Application of the model to the Test cases

The fitted model will now be applied to predict the 20 cases of the **FinalTest** dataset. The answers are stored in the character vector `answers` which will be used to create the files to be submitted for evaluation.

```
answers <- as.character(predict(modelFit, FinalTest))
summary(answers)
```

```
##    Length     Class      Mode 
##        20 character character
```