Practical Machine Learning Project

Sergey Ostrovsky

July 19, 2017

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).

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.

How model was built

The outcome variable is classe, a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E)

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. All other available variables after cleaning will be used for prediction.

Three models will be tested with Bootstrapped Aggregation, Boosted C5.0, and Random Forest algorithms. The model with the highest accuracy will be chosen for our final model.

Use of cross validation

Cross-validation will be used by splitting our training data into two subsamples without replacement using 75% for training set and 25% for testing set. Our models will be fitted on the training sample data and tested on the testing sample data. Finally, the most accurate model will be tested on the original Testing data set.

Expected out of sample error

The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the subTesting data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set). Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Loading And Preparing Datasets

```r
set.seed(123567)
## Downloads and extacts data from given dataset
wd <- getwd()
training_url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing_url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training_file <- "pml-training.csv"
if (!file.exists(training_file)) {
    download.file(training_url, file.path(wd, training_file))
}

testing_file <- "pml-testing.csv"
if (!file.exists(testing_file)) {
    download.file(testing_url, file.path(wd, testing_file))
}

train <- read.csv(training_file, na.strings=c("NA","#DIV/0!", ""))
test <- read.csv(testing_file, na.strings=c("NA","#DIV/0!", ""))
```

Cleaning Datasets From Useless Columns

```r
# Delete columns where missing values exist
train <- train[,colSums(is.na(train)) == 0]
test <- test[,colSums(is.na(test)) == 0]

# Delete descriptive columns
train <- train[, -c(1:7)]
test <- test[, -c(1:7)]
```

Partitioning the training dataset for cross-validation

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
inTrain <- createDataPartition(y = train$classe, p = 0.75, list = FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]
```

First prediction model: Bootstrapped Aggregation

```
library(ipred)
baFit <- bagging(classe ~. , data=training, method="class")
baPredict <- predict(baFit, testing, type = "class")
confusionMatrix(baPredict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1387   12    0    0    0
##          B    6  931    4    0    4
##          C    0    5  847   12    3
##          D    1    1    4  791    3
##          E    1    0    0    1  891
##
## Overall Statistics
##
##                Accuracy : 0.9884
##                  95% CI : (0.985, 0.9912)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9853
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9943   0.9810   0.9906   0.9838   0.9889
## Specificity            0.9966   0.9965   0.9951   0.9978   0.9995
## Pos Pred Value         0.9914   0.9852   0.9769   0.9888   0.9978
## Neg Pred Value         0.9977   0.9955   0.9980   0.9968   0.9975
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2828   0.1898   0.1727   0.1613   0.1817
## Detection Prevalence   0.2853   0.1927   0.1768   0.1631   0.1821
## Balanced Accuracy      0.9954   0.9887   0.9929   0.9908   0.9942
```

Second prediction model: Using Boosted C5.0

```
library(C50)
bC50Fit <- C5.0(classe ~. , data=training, trials=10)
bc50Predict <- predict(bC50Fit, testing, type = "class")
confusionMatrix(bc50Predict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    1    0
##          B    1  946   10    0    0
##          C    0    2  841    8    1
##          D    0    0    4  793    4
##          E    0    0    0    2  896
##
## Overall Statistics
##
##                Accuracy : 0.9931
##                  95% CI : (0.9903, 0.9952)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9968   0.9836   0.9863   0.9945
## Specificity            0.9994   0.9972   0.9973   0.9980   0.9995
## Pos Pred Value         0.9986   0.9885   0.9871   0.9900   0.9978
## Neg Pred Value         0.9997   0.9992   0.9965   0.9973   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1929   0.1715   0.1617   0.1827
## Detection Prevalence   0.2847   0.1951   0.1737   0.1633   0.1831
## Balanced Accuracy      0.9994   0.9970   0.9905   0.9922   0.9970
```

Third prediction model: Using Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rfFit <- randomForest(classe ~. , data=training, method="class")
rfPredict <- predict(rfFit, testing, type = "class")
confusionMatrix(rfPredict, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    2    0    0    0
##          B    0  946    2    0    0
##          C    0    1  851   11    0
##          D    0    0    2  792    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9957
##                  95% CI : (0.9935, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9953   0.9851   0.9978
## Specificity            0.9994   0.9995   0.9970   0.9990   0.9998
## Pos Pred Value         0.9986   0.9979   0.9861   0.9950   0.9989
## Neg Pred Value         1.0000   0.9992   0.9990   0.9971   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1735   0.1615   0.1833
## Detection Prevalence   0.2849   0.1933   0.1760   0.1623   0.1835
## Balanced Accuracy      0.9997   0.9982   0.9962   0.9920   0.9988
```

Decision

From the result above we can see that Random Forest algorithm has highest accuracy. Thus, this model will be chosed for final prediction with original Testing set.

Submission

```r
predict(rfFit, test, type = "class")
```

```
##  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
```