Human Activity Recognition

JacobaDS

18 december 2017

Introduction

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, my 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of my project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set.

Load the data and remove missing values

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har).

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

```
Train <- read.csv("pml-training.csv", na.strings = c("NA", ""))
#remove missing values, using the following subsetting:
Train_use <- Train[, colSums(is.na(Train)) == 0]
```

Cross Validation

1. Firstly, I split the Training data into a sub-training set (60%) and a sub-test set (remaining 40%)
2. Secondly, I build the model on the sub-training set
3. Thirdly, I evaluate my model on the sub-test set

```
#Cross validation => create a sub-training set (60%) and a sub-test set (rem
aining 40%)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
set.seed(1000)
inTrain <- createDataPartition(Train_use$classe, times = 1, p=0.60, list=FAL
SE)
Training <- Train_use[inTrain,]
Testing <-Train_use[-inTrain,]
```

Building the model (using the Training data)

I choose a random forest algorithm, because this is one of the most used/accurate algorithms. I prefer randomForest(), because this algorithm runs much faster than train(classe ~ ., data = Training, method="rf").

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.3
```

```
## 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
```

```
rf <- randomForest(classe ~ ., data=Training)
rf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = Training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.01%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3348    0    0    0    0 0.0000000000
## B    0 2279    0    0    0 0.0000000000
## C    0    0 2054    0    0 0.0000000000
## D    0    0    1 1929    0 0.0005181347
## E    0    0    0    0 2165 0.0000000000
```

Expected out-of-sample error

As can be seen from the output above, the in-sample-error rate = 0.01%. This is extremely small. I expect the out-of-sample error to be above 0.01% (=> above the in-sample-error rate, which is usually the case, but not necessarily).

Cross Validation: prediction on the Testing data

Finally, I evaluate my model on the sub-test set.

```
Pred_rf <- predict(rf,Testing)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.3
```

```
confusionMatrix(Testing$classe,Pred_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1518    0    0    0
##          C    0    0 1368    0    0
##          D    0    0    0 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9995, 1)
##     No Information Rate : 0.2845
##     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             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The accuary is 1. In other words, the out-of-sample error rate = 0%. As can be seen in the Confusion Matrix, all predictions are exactly right.