Final Project Machine Learning

Geovanni Becerril

2018-02-15

Introduction

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.

Read the data, previously downloaded:

```
training_complete<-read.csv("pml-training.csv",na.strings=c("NA","#DIV/0!",""))
testing_complete<-read.csv("pml-testing.csv",na.strings=c("NA","#DIV/0!",""))
```

Cleaning the data sets

The data sets have a lot NA's. so it was decided to keep the variables that had complete information and don't have unique value.

```
#Loading libraries
library(caret)
```

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

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

```
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
```

```
library(rpart)
#Eliminate the predictors that have unique value.
unique.value<-nearZeroVar(training_complete)
training_aux<-training_complete[,-c(1,unique.value)]
#Keep columns with complete data
training<-training_aux[,c(2,3,5:9,32:44,46:54,65:67,80,91:102,112,114:123)]
```

Data partitioning

The training dataset contains the 75% of data, while the testing set, to evaluate the model, contains the 25%.

```
set.seed(11235)
inTrain<-createDataPartition(training$classe,p=0.75,list=FALSE)
training_set<-training[inTrain,]
testing_set<-training[-inTrain,]
```

The training dataset contains 14,718 observations. The testing dataset contains 4,904 observations.

Fitting the model

We use three different machine learning algorithms to select the best model to predict: Gradient Boosted Machine, Random Forest and Decision Trees.

1. Gradient Boosted Machine (GBM)

The code used is the following:

```
set.seed(11235)
fitGBM<-train(classe~.,data=training_set,method="gbm",
               trControl=trainControl(method="repeatedcv",
                                      number=5,repeats=1))
pred_GBM<-predict(fitGBM,newdata = testing_set)
cm_GBM<-confusionMatrix(pred_GBM, testing_set$classe)
```

The confusion matrix is:

```
cm_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1389    0    0    0    0
##          B    6  949    0    0    0
##          C    0    0  853    0    0
##          D    0    0    2  799    4
##          E    0    0    0    5  897
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9965
##                  95% CI : (0.9945, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9957   1.0000   0.9977   0.9938   0.9956
## Specificity            1.0000   0.9985   1.0000   0.9985   0.9988
## Pos Pred Value         1.0000   0.9937   1.0000   0.9925   0.9945
## Neg Pred Value         0.9983   1.0000   0.9995   0.9988   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2832   0.1935   0.1739   0.1629   0.1829
## Detection Prevalence   0.2832   0.1947   0.1739   0.1642   0.1839
## Balanced Accuracy      0.9978   0.9992   0.9988   0.9962   0.9972
```

2. Random Forests (RF)

The code used is the following:

```
set.seed(11235)
fitRF<-randomForest(classe ~ ., data=training_set)
pred_RF<-predict(fitRF, testing_set, type = "class")
cm_RF<-confusionMatrix(pred_RF, testing_set$classe)
cm_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    2    0
##          D    0    0    0  802    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9996
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9995
##  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   0.9975   1.0000
## Specificity            1.0000   1.0000   0.9995   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9977   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1743   0.1635   0.1837
## Detection Prevalence   0.2845   0.1935   0.1748   0.1635   0.1837
## Balanced Accuracy      1.0000   1.0000   0.9998   0.9988   1.0000
```

3. Decision Trees (DT)

The code used is the following

```
set.seed(11235)
fitDT<-rpart(classe~.,data=training_set,method="class")
pred_DT<-predict(fitDT, testing_set, type = "class")
cm_DT<-confusionMatrix(pred_DT, testing_set$classe)
cm_DT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1292   85    6   39   48
##          B   50  689   27    9   64
##          C    4   41  733   53   67
##          D   28   81   32  645   75
##          E   21   53   57   58  647
##
## Overall Statistics
##
##                Accuracy : 0.8169
##                  95% CI : (0.8058, 0.8276)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7681
##  Mcnemar's Test P-Value : 2.663e-15
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9262   0.7260   0.8573   0.8022   0.7181
## Specificity            0.9493   0.9621   0.9592   0.9473   0.9528
## Pos Pred Value         0.8789   0.8212   0.8163   0.7491   0.7739
## Neg Pred Value         0.9700   0.9360   0.9695   0.9607   0.9376
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2635   0.1405   0.1495   0.1315   0.1319
## Detection Prevalence   0.2998   0.1711   0.1831   0.1756   0.1705
## Balanced Accuracy      0.9377   0.8441   0.9083   0.8748   0.8354
```

Discussion

In the confusionMatrix it is observed that the algorithm with greater precision is the GBM and the RF, while the DT is bad for predicting the data. The accuracy obtained in each algorithm can be seen in the following table:

Algorithm	Accuracy
Gradient Boosted Machine	0.9965334
Random Forests	0.9995922
Decision Trees	0.8168842

Predicting in the Testing Data

The goal is to use the prediction model to predict 20 different test cases. the algorithm to use is the Random Forest because it has better accuracy. The following code makes the prediction for the 20 test cases.

```
predict_final<-predict(fitRF,newdata = testing_complete)
```

The predictions for the 20 test case are:

```
predictions<-data.frame(names=testing_complete[,2],class=predict_final)
predictions
```

```
##       names class
## 1     pedro     B
## 2    jeremy     A
## 3    jeremy     B
## 4    adelmo     A
## 5    eurico     A
## 6    jeremy     E
## 7    jeremy     D
## 8    jeremy     B
## 9  carlitos     A
## 10  charles     A
## 11 carlitos     B
## 12   jeremy     C
## 13   eurico     B
## 14   jeremy     A
## 15   jeremy     E
## 16   eurico     E
## 17    pedro     A
## 18 carlitos     B
## 19    pedro     B
## 20   eurico     B
```