Prediction Assignment Writeup

TCH

Saturday, December 02, 2017

Overview

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).

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.

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Loading Data and Exploratory Analysis

Firstly, it is necessary to load the R packages needed and download the training and testing data sets from the given URLs.

```
library(lattice);
library(ggplot2);
library(caret);
library(rattle);
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Entrez 'rattle()' pour secouer, faire vibrer, et faire défiler vos données.
```

```
library(rpart);
library(rpart.plot)
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:rattle':
##
##     importance
```

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

```
set.seed(7899)

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

We can observe that the training dataset has 19622 observations and 160 variables, and the testing dataset has 20 observations and also 160 variables. The "classe" variable in the training set is going to be predicted.

Now, we clean the data by deleting missing values and all unnecessary columns.

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]

datatrain <- training[, -c(1:7)]
datatest <- testing[, -c(1:7)]
```

Then, we split the cleaned training dataset into a training set (train 60%) for prediction and a validation set (valid 40%).

```
set.seed(7899)
inTrain <- createDataPartition(datatrain$classe, p = 0.6, list = FALSE)
train <- datatrain[inTrain, ]
valid <- datatrain[-inTrain, ]
dim(train)
```

```
## [1] 11776    53
```

```
dim(valid)
```

```
## [1] 7846   53
```

We can observe that we have now 11776 observations in the train and 7846 observations in the validation with 53 variables in both.

Prediction

We try to predict with : Regression, Decision Trees and Random Forests.

With Regression

```
set.seed(7899)
fit <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
gbm <- train(classe ~ ., data=train, method = "gbm", trControl = fit, verbose=FALSE)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
```

```
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
gbmFinal <- gbm$finalModel
gbmValid <- predict(gbm, newdata=valid)
gbmFinalValid <- confusionMatrix(gbmValid, valid$classe)
gbmFinalValid
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2204   43    0    0    1
##          B   17 1430   38    3   18
##          C    5   40 1309   35    6
##          D    6    2   17 1243   18
##          E    0    3    4    5 1399
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95% CI : (0.9625, 0.9706)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9579
##  Mcnemar's Test P-Value : 5.68e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9875   0.9420   0.9569   0.9666   0.9702
## Specificity            0.9922   0.9880   0.9867   0.9934   0.9981
## Pos Pred Value         0.9804   0.9495   0.9384   0.9666   0.9915
## Neg Pred Value         0.9950   0.9861   0.9909   0.9934   0.9933
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2809   0.1823   0.1668   0.1584   0.1783
## Detection Prevalence   0.2865   0.1919   0.1778   0.1639   0.1798
## Balanced Accuracy      0.9898   0.9650   0.9718   0.9800   0.9842
```

With Decision Trees

```
set.seed(7899)
fit2 <- trainControl(method = "cv", number = 5)
rpart <- train(classe ~ ., data = train, method = "rpart", trControl = fit2)
rpartValid <- predict(rpart, valid)
rpartFinalValid <- confusionMatrix(rpartValid, valid$classe)
rpartFinalValid
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    A    B    C    D    E
##          A 2004  647  630  563  180
##          B   35  502   44  231  198
##          C  160  369  694  492  367
##          D    0    0    0    0    0
##          E   33    0    0    0  697
##
## Overall Statistics
##
##                Accuracy : 0.4967
##                  95% CI : (0.4856, 0.5078)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3428
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8978  0.33070  0.50731   0.0000  0.48336
## Specificity            0.6402  0.91972  0.78574   1.0000  0.99485
## Pos Pred Value         0.4980  0.49703  0.33333      NaN  0.95479
## Neg Pred Value         0.9403  0.85138  0.88307   0.8361  0.89531
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2554  0.06398  0.08845   0.0000  0.08884
## Detection Prevalence   0.5129  0.12873  0.26536   0.0000  0.09304
## Balanced Accuracy      0.7690  0.62521  0.64652   0.5000  0.73910
```

With Random Forests

```
set.seed(7899)
random <- randomForest(classe ~ ., data = train)
randomValid <- predict(random, valid, type = "class")
randomFinalValid <- confusionMatrix(randomValid, valid$classe)
randomFinalValid
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   13    0    0    0
##          B    2 1497    8    0    0
##          C    0    8 1360    7    0
##          D    0    0    0 1279    4
##          E    1    0    0    0 1438
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.9926, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9862   0.9942   0.9946   0.9972
## Specificity            0.9977   0.9984   0.9977   0.9994   0.9998
## Pos Pred Value         0.9942   0.9934   0.9891   0.9969   0.9993
## Neg Pred Value         0.9995   0.9967   0.9988   0.9989   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1908   0.1733   0.1630   0.1833
## Detection Prevalence   0.2858   0.1921   0.1752   0.1635   0.1834
## Balanced Accuracy      0.9982   0.9923   0.9959   0.9970   0.9985
```

So we can see, from the confusion matrix of 3 methods, the accuracy rate of Random Forests method is 0.9945 compared to 0.4967 of Decision Trees and 0.9667 of GBM. So Random Forests method is the best way to predict in this case.

Prediction on Testing dataset

```
(predict(random, testing, 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
```