ML CP

Chris De Coster

Sunday, October 26, 2014

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 (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
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Packages used

```
library(caret)
library(randomForest)
library(data.table)
set.seed(1)
```

Getting and Cleaning Data:

Files are dowmloaded and imported using read.csv

```
if(!file.exists("pml-training.csv")){
  download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfi
le = "pml-training.csv")}
if(!file.exists("pml-testing.csv")){
  download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfil
e = "pml-testing.csv")}

df<- read.csv(file = file.path(getwd(),'pml-training.csv'), na.strings=c('NA','','#DIV/0!'))
df_test <- read.csv(file = file.path(getwd(), 'pml-testing.csv'), na.strings=c('NA','','#DIV/0!'))
```

Next, the columns with NA values are removed and also the first 7 columns because they do not contain data of any use for the prediction.

```
na <- apply(df, 2, function(x) {sum(is.na(x))})
df <- subset(df[, which(na == 0)], select=-c(X, user_name, new_window, num_window, raw_timestamp_p
art_1, raw_timestamp_part_2, cvtd_timestamp))
```

Building the Model

Split data:

I use 60% of the training data(file) to generate a training dataset and 40% for a testing dataset, partioned using the “classe” variable.

```
inTrain <- createDataPartition(y = df$classe, p=0.60, list=FALSE)
training <- df[inTrain,]
testing <- df[-inTrain,]
```

Prediction:

We train the dataset using the randowforest function of the caret package. The number of trees to grow is 20 to ensure that every input row gets predicted at least a few times.

```
model <- train(training$classe ~., data = training, method="rf", ntree = 20)
prediction <- predict(model,newdata=training)
confusionMatrix(prediction, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.194    0.174    0.164    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

The matrix shows an accurancy of 1, this means no error. Very good, so there is no need to use another function or adapt parameters.

Now we target the testing dataset only once to get our final results:

```
prediction <- predict(model,newdata=testing)
confusionMatrix(prediction, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2225    9    0    1    0
##          B    3 1494    5    0    4
##          C    3   14 1354   22    4
##          D    0    0    9 1262    6
##          E    1    1    0    1 1428
##
## Overall Statistics
##
##                Accuracy : 0.989
##                  95% CI : (0.987, 0.992)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.987
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.997    0.984    0.990    0.981    0.990
## Specificity             0.998    0.998    0.993    0.998    1.000
## Pos Pred Value          0.996    0.992    0.969    0.988    0.998
## Neg Pred Value          0.999    0.996    0.998    0.996    0.998
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.190    0.173    0.161    0.182
## Detection Prevalence    0.285    0.192    0.178    0.163    0.182
## Balanced Accuracy       0.998    0.991    0.992    0.990    0.995
```

This time the matrix shows a accurancy of 0.989, so an error of 0.011.

Submission part:

Apply algorithm to test data:

```
na <- apply(df_test, 2, function(x) {sum(is.na(x))})
df_test <- subset(df_test[, which(na == 0)], select=-c(X, user_name, new_window, num_window, raw_t
imestamp_part_1, raw_timestamp_part_2, cvtd_timestamp))

test_prediction <- predict(model, newdata = df_test)

pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

# write the predictons generated in the previsou step into files.
pml_write_files(test_prediction)
```