Practical Machine Learning Peer Assignment

Buğra Balkaç

12/14/2020

Summary

This report uses machine learning algorithms to predict the manner in which users of exercise devices exercise.

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: (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

Set the work environment and knitr options

```r
rm(list=ls(all=TRUE)) #start with empty workspace
startTime <- Sys.time()
library(knitr)
opts_chunk$set(echo = TRUE, cache= TRUE, results = 'hold')
```

Load libraries and Set Seed

Load all libraries used, and setting seed for reproducibility. *Results Hidden, Warnings FALSE and Messages FALSE*

```r
library(caret)
library(rpart)
library(randomForest)
library(RCurl)
set.seed(2020)
```

Load and prepare the data and clean up the data

Load and prepare the data

```r
trainingLink <- getURL("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
pml_CSV  <- read.csv(text = trainingLink, header=TRUE, sep=",", na.strings=c("NA",""))
pml_CSV <- pml_CSV[,-1] # Remove the first column that represents a ID Row
```

Data Sets Partitions Definitions

Create data partitions of training and validating data sets.

```r
inTrain = createDataPartition(pml_CSV$classe, p=0.60, list=FALSE)
training = pml_CSV[inTrain,]
validating = pml_CSV[-inTrain,]
# number of rows and columns of data in the training set
dim(training)
# number of rows and columns of data in the validating set
dim(validating)
```

```
## [1] 11776   159
## [1] 7846  159
```

Data Exploration and Cleaning

Since we choose a random forest model and we have a data set with too many columns, first we check if we have many problems with columns without data. So, remove columns that have less than 60% of data entered.

```r
# Number of cols with less than 60% of data
sum((colSums(!is.na(training[,-ncol(training)])) < 0.6*nrow(training)))
```

[1] 100

```r
# apply our definition of remove columns that most doesn't have data, before its apply to the model.
Keep <- c((colSums(!is.na(training[,-ncol(training)])) >= 0.6*nrow(training)))
training   <-  training[,Keep]
validating <- validating[,Keep]
# number of rows and columns of data in the final training set
dim(training)
```

[1] 11776 59

```r
# number of rows and columns of data in the final validating set
dim(validating)
```

[1] 7846 59

```r
training$user_name<-factor(training$user_name)
training$new_window<-factor(training$new_window)
training$cvtd_timestamp<-factor(training$cvtd_timestamp)
training$classe<-factor(training$classe)

validating$user_name<-factor(validating$user_name)
validating$new_window<-factor(validating$new_window)
validating$cvtd_timestamp<-factor(validating$cvtd_timestamp)
validating$classe<-factor(validating$classe)
```

Modeling

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the execution. So, we proceed with the training the model (Random Forest) with the training data set.

```r
model <- randomForest(classe ~ .,data=training)
print(model)
```

```
##
## 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: 0.17%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3348    0    0    0    0 0.000000000
## B    2 2276    1    0    0 0.001316367
## C    0    4 2048    2    0 0.002921130
## D    0    0    6 1922    2 0.004145078
## E    0    0    0    3 2162 0.001385681
```

Model Evaluate

And proceed with the verification of variable importance measures as produced by random Forest:

```r
importance(model)
```

```
##                      MeanDecreaseGini
## user_name                  98.9089820
## raw_timestamp_part_1      944.0311078
## raw_timestamp_part_2       10.4029122
```

```
## cvtd_timestamp          1398.2720849
## new_window                 0.2038923
## num_window               559.6816291
## roll_belt                518.4798100
## pitch_belt               293.9582514
## yaw_belt                 332.2147175
## total_accel_belt         104.8849815
## gyros_belt_x              39.0173077
## gyros_belt_y              56.1821641
## gyros_belt_z             126.3018655
## accel_belt_x              65.9504258
## accel_belt_y              67.7366902
## accel_belt_z             197.1869906
## magnet_belt_x            115.3666945
## magnet_belt_y            209.1606202
## magnet_belt_z            194.1546410
## roll_arm                 116.7263888
## pitch_arm                 53.6317223
## yaw_arm                   81.4944530
## total_accel_arm           27.8216851
## gyros_arm_x               40.5539308
## gyros_arm_y               43.3927287
## gyros_arm_z               17.4678568
## accel_arm_x               95.3028424
## accel_arm_y               53.7147255
## accel_arm_z               41.0149602
## magnet_arm_x              95.1680960
## magnet_arm_y              73.2193934
## magnet_arm_z              58.7417701
## roll_dumbbell            189.7878975
## pitch_dumbbell            81.6941799
## yaw_dumbbell              99.5760087
## total_accel_dumbbell     116.2686929
## gyros_dumbbell_x          41.7980307
## gyros_dumbbell_y         115.9631661
## gyros_dumbbell_z          23.9479599
## accel_dumbbell_x         126.6783989
## accel_dumbbell_y         187.5211923
## accel_dumbbell_z         130.7202750
## magnet_dumbbell_x        231.1482167
## magnet_dumbbell_y        322.3210961
## magnet_dumbbell_z        300.8988505
## roll_forearm             229.2167244
## pitch_forearm            306.8954720
## yaw_forearm               52.6418923
## total_accel_forearm       32.8858437
## gyros_forearm_x           25.1902698
## gyros_forearm_y           39.3548614
## gyros_forearm_z           27.8813007
## accel_forearm_x          137.2949575
## accel_forearm_y           44.5834329
## accel_forearm_z           88.1606754
## magnet_forearm_x          70.2206732
## magnet_forearm_y          70.2193828
```

```
## magnet_forearm_z           86.3942445
```

Now we evaluate our model results through confusion Matrix.

```r
confusionMatrix(predict(model,newdata=validating[,-ncol(validating)]),validating$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1516    4    0    0
##          C    0    0 1364    1    0
##          D    0    0    0 1285    2
##          E    0    0    0    0 1440
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9989          
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9985          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9971   0.9992   0.9986
## Specificity            0.9996   0.9994   0.9998   0.9997   1.0000
## Pos Pred Value         0.9991   0.9974   0.9993   0.9984   1.0000
## Neg Pred Value         1.0000   0.9997   0.9994   0.9998   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1738   0.1638   0.1835
## Detection Prevalence   0.2847   0.1937   0.1740   0.1640   0.1835
## Balanced Accuracy      0.9998   0.9990   0.9985   0.9995   0.9993
```

And confirmed the accuracy at validating data set by calculate it with the formula:

```r
accuracy <-c(as.numeric(predict(model,newdata=validating[,-ncol(validating)])==validating$classe))
accuracy <-sum(accuracy)*100/nrow(validating)
```

Model Accuracy as tested over Validation set = **99.9%**.

Model Test

Finally, we proceed with predicting the new values in the testing csv provided, first we apply the same data cleaning operations on it and coerce all columns of testing data set for the same class of previous data set.

```
testingLink <- getURL("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
pml_CSV  <- read.csv(text = testingLink, header=TRUE, sep=",", na.strings=c("NA",""))
pml_CSV <- pml_CSV[,-1] # Remove the first column that represents a ID Row
pml_CSV <- pml_CSV[ , Keep] # Keep the same columns of testing dataset
pml_CSV <- pml_CSV[,-ncol(pml_CSV)] # Remove the problem ID
# Apply the Same Transformations and Coerce Testing Dataset
# Coerce testing dataset to same class and strucuture of training dataset
testing <- rbind(training[100, -59] , pml_CSV)
# Apply the ID Row to row.names and 100 for dummy row from testing dataset
row.names(testing) <- c(100, 1:20)
```

Getting Testing Dataset

```
predictions <- predict(model,newdata=testing[-1,])
print(predictions)
```

Predicting with testing dataset

```
##  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
```

```
write.table(predictions,file="problem.txt",quote=FALSE,row.names = FALSE,col.names=FALSE)
#get the time
```

```
endTime <- Sys.time()
```

The following function to create the files to answers the Prediction Assignment Submission: The analysis was completed on Mon Dec 14 12:59:14 PM 2020 in 3 seconds.