Practical Machine Learning Coursera Project

Jagrut

26/08/2020

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

```
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*

```
library(caret)
library(rpart)
library(randomForest)
library(RCurl)
set.seed(2014)
```

Load and prepare the data and clean up the data

Load and prepare the data

```
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.

```
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.

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

[1] 100

```
# 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

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(as.factor(classe)~.,data=training)
print(model)
```

```
##
## Call:
##  randomForest(formula = as.factor(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: 0.18%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    2 2277    0    0    0 0.0008775779
## C    0    6 2046    2    0 0.0038948393
## D    0    0    4 1924    2 0.0031088083
## E    0    0    0    4 2161 0.0018475751
```

Model Evaluate

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

```r
importance(model)
```

```
##                      MeanDecreaseGini
## user_name                  45.4060433
## raw_timestamp_part_1     1050.5385348
## raw_timestamp_part_2       12.0545141
## cvtd_timestamp            578.2735528
## new_window                  0.3421283
## num_window                624.8824704
## roll_belt                 618.2179028
## pitch_belt                332.3663255
## yaw_belt                  383.7849303
## total_accel_belt          113.9257030
## gyros_belt_x               41.4222833
## gyros_belt_y               54.6478846
## gyros_belt_z              122.3010876
```

```
## accel_belt_x              60.8171364
## accel_belt_y              63.9603205
## accel_belt_z             206.8074365
## magnet_belt_x            114.6060566
## magnet_belt_y            197.6493184
## magnet_belt_z            192.5051508
## roll_arm                 141.0288983
## pitch_arm                 60.0608536
## yaw_arm                   92.0245359
## total_accel_arm           36.7324616
## gyros_arm_x               48.0298198
## gyros_arm_y               49.0599928
## gyros_arm_z               22.1497386
## accel_arm_x              103.7052303
## accel_arm_y               60.2523574
## accel_arm_z               50.9146767
## magnet_arm_x             116.2915190
## magnet_arm_y              97.7440407
## magnet_arm_z              64.3899906
## roll_dumbbell            191.8792765
## pitch_dumbbell            85.1779976
## yaw_dumbbell             130.7090362
## total_accel_dumbbell     112.4854498
## gyros_dumbbell_x          42.6894796
## gyros_dumbbell_y         104.1377433
## gyros_dumbbell_z          31.4840682
## accel_dumbbell_x         135.0174573
## accel_dumbbell_y         198.6534381
## accel_dumbbell_z         152.9469994
## magnet_dumbbell_x        226.7368108
## magnet_dumbbell_y        339.2889851
## magnet_dumbbell_z        366.2576628
## roll_forearm             275.0080538
## pitch_forearm            375.8467141
## yaw_forearm               68.0403539
## total_accel_forearm       37.2014906
## gyros_forearm_x           29.0992941
## gyros_forearm_y           46.0498360
## gyros_forearm_z           34.8071438
## accel_forearm_x          140.6172414
## accel_forearm_y           50.6697073
## accel_forearm_z           98.3638351
## magnet_forearm_x          85.8077511
## magnet_forearm_y          88.7622520
## magnet_forearm_z         105.2446691
```

Now we evaluate our model results through confusion Matrix.

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

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

```
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    3    0    0
##          C    0    0 1365    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            0.9996   0.9987   0.9978   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         0.9998   0.9997   0.9995   0.9998   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1740   0.1638   0.1835
## Detection Prevalence   0.2846   0.1937   0.1741   0.1640   0.1835
## Balanced Accuracy      0.9996   0.9990   0.9988   0.9995   0.9993
```

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

```
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)
```

```r
# 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

```r
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
```