Prediction: Assignment Writeup

Allen Brown

2/18/2019

Summary

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

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.

```r
library(caret)
library(randomForest)
set.seed(2019)
```

Load and prepare the data and clean up the data

```
trainingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainingFile <- "pml-training.csv"
testingFile <- "pml-testing.csv"
download.file(url=trainingURL, destfile=trainingFile,method="curl")
download.file(url=testingURL, destfile=testingFile,method="curl")
training <- read.csv("pml-training.csv",row.names=1,na.strings=c("NA",""))
testing <- read.csv("pml-testing.csv",row.names=1,na.strings=c("NA",""))
dim(training)
```

[1] 19622 159

```
dim(testing)
```

[1] 20 159

Data Sets Partitions Definitions

Create data partitions of training and validating data sets.

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

[1] 11776 159

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

[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(trainingClean[,-ncol(trainingClean)])) < 0.6*nrow(trainingClean)))
```

[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(trainingClean[,-ncol(trainingClean)])) >= 0.6*nrow(trainingClean)))
trainingClean   <-  trainingClean[,Keep]
validationClean <- validationClean[,Keep]
# number of rows and columns of data in the final training set
dim(trainingClean)
```

[1] 11776 59

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

[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(classe~.,data=trainingClean)
print(model)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainingClean)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3348    0    0    0    0 0.0000000000
## B    1 2278    0    0    0 0.0004387889
## C    0    7 2045    2    0 0.0043816943
## D    0    0    7 1922    1 0.0041450777
## 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                 101.0456230
## raw_timestamp_part_1      967.6919965
## raw_timestamp_part_2        9.9076934
## cvtd_timestamp           1405.1811629
## new_window                  0.1823449
## num_window                561.4160406
## roll_belt                 517.6594776
## pitch_belt                290.5769685
## yaw_belt                  344.0239135
## total_accel_belt           96.9051811
## gyros_belt_x               37.1290440
## gyros_belt_y               52.6271451
## gyros_belt_z              127.6052379
## accel_belt_x               65.5611264
## accel_belt_y               66.1606503
## accel_belt_z              189.6467371
## magnet_belt_x             116.2987052
```

```
## magnet_belt_y              224.7442944
## magnet_belt_z              183.6308868
## roll_arm                   116.6565866
## pitch_arm                   53.3803982
## yaw_arm                     80.9782269
## total_accel_arm             27.0909066
## gyros_arm_x                 41.2496865
## gyros_arm_y                 41.4532790
## gyros_arm_z                 18.2907908
## accel_arm_x                 92.6749301
## accel_arm_y                 47.2455095
## accel_arm_z                 38.1378558
## magnet_arm_x               100.6991201
## magnet_arm_y                76.8098884
## magnet_arm_z                57.0403023
## roll_dumbbell              195.0492865
## pitch_dumbbell              87.8949921
## yaw_dumbbell               112.8225670
## total_accel_dumbbell       127.9647787
## gyros_dumbbell_x            40.0152791
## gyros_dumbbell_y           104.8218443
## gyros_dumbbell_z            24.0909258
## accel_dumbbell_x           128.4018773
## accel_dumbbell_y           187.3658238
## accel_dumbbell_z           134.9556019
## magnet_dumbbell_x          217.9288504
## magnet_dumbbell_y          308.1659870
## magnet_dumbbell_z          307.9549718
## roll_forearm               228.7027117
## pitch_forearm              294.9762929
## yaw_forearm                 51.7343233
## total_accel_forearm         31.0883499
## gyros_forearm_x             23.5842654
## gyros_forearm_y             37.9642259
## gyros_forearm_z             26.1209407
## accel_forearm_x            118.8447915
## accel_forearm_y             41.5289041
## accel_forearm_z             91.6680843
## magnet_forearm_x            81.6662876
## magnet_forearm_y            64.9396349
## magnet_forearm_z            90.5627445
```

Now we evaluate our model results through confusion Matrix.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1516    4    0    0
##          C    0    1 1363    3    0
##          D    0    0    1 1283    0
##          E    0    0    0    0 1442
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9987
##                  95% CI : (0.9977, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9984
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9963   0.9977   1.0000
## Specificity            0.9998   0.9994   0.9994   0.9998   1.0000
## Pos Pred Value         0.9996   0.9974   0.9971   0.9992   1.0000
## Neg Pred Value         1.0000   0.9997   0.9992   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1737   0.1635   0.1838
## Detection Prevalence   0.2846   0.1937   0.1742   0.1637   0.1838
## Balanced Accuracy      0.9999   0.9990   0.9979   0.9988   1.0000
```

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

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

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.

Getting Testing Dataset

```r
testing  <- testing [ , Keep] # Keep the same columns of testing dataset
testing  <- testing [,-ncol(testing)] # Remove the problem ID

# Coerce testing dataset to same class and structure of training dataset
testClean <- rbind(trainingClean[100, -59] , testing)

# Apply the ID Row to row.names and 100 for dummy row from testing dataset
row.names(testClean) <- c(100, 1:20)
```

Predicting with testing dataset

```r
predictions <- predict(model,newdata=testClean[-1,])
predictions
```

```
##  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
```

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

The analysis was completed on Mon Feb 18 21:32:19 2019 in 0 seconds.