Practical Machine Learning Peer Assessments

Pei-Chun Su

Saturday, February 13, 2016

Executive Summary

This document presents the results of the Practical Machine Learning Peer Assessments in a report using a single R markdown document that can be processed by knitr and be transformed into an HTML file.

Introduction

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, the 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).

Data

The training data for this project are downloaded from:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are downloaded from:
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.

It is worth citing Groupware@LES for being generous in allowing their data to be used for this assignment.

Goal of the assignment

Predicting the manner in which the participants did the exercise. Refer to the "classe" variable in the training set. All other variables can be used as predictor.

Show how the model was built, performed cross validation, and expectation of the sample error and reasons of choices made.

Use the prediction model to predict 20 different test cases.

Data Preprocessing

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
library(rpart)
library(knitr)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(ElemStatLearn)
library(corrplot)
set.seed(666) # For research reproducibility purpose
setwd("C:/Users/Pei-Chun/Documents/R/Machine learning")
```

Preparation of Datasets

```
trainRaw <- read.csv("./pml-training.csv",header=T,sep=",",na.strings=c
("NA",""))
testRaw <- read.csv("./pml-testing.csv",header=T,sep=",",na.strings=c("
NA",""))
```

Data Sets Partitioning Definitions

The data partitions of training and validating data sets are created as below:

```
trainRaw <- trainRaw[,-1] # Remove the first column that represents a ID
 Row
inTrain = createDataPartition(trainRaw$classe, p=0.60, list=F)
training = trainRaw[inTrain,]
validating = trainRaw[-inTrain,]
```

Data Cleaning

Since a random forest model is chosen and the data set must first be checked on possibility of columns without data.

The decision is made whereby all the columns that having less than 60% of data filled are removed.

```
sum((colSums(!is.na(training[,-ncol(training)])) < 0.6*nrow(training)))
```

```
## [1] 100
```

Next, the criteria to remove columns that do not satisfy is applied before applying to the model.

```
Keep <- c((colSums(!is.na(training[,-ncol(training)])) >= 0.6*nrow(trai
ning)))
training   <-  training[,Keep]
validating <- validating[,Keep]
```

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. Therefore, the training of the model (Random Forest) is proceeded using the training data set.

```
model <- randomForest(classe~.,data=training)
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: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    3 2276    0    0    0 0.0013163668
## C    0    4 2046    4    0 0.0038948393
## D    0    0    5 1924    1 0.0031088083
## E    0    0    0    4 2161 0.0018475751
```

Model Evaluation

Verification of the variable importance measures as produced by random Forest is as follows:

```
importance(model)
```

```
##                     MeanDecreaseGini
## user_name                 86.6501781
## raw_timestamp_part_1     958.1625357
## raw_timestamp_part_2      10.4197058
## cvtd_timestamp          1403.7206581
## new_window                 0.2099771
## num_window               556.6817620
## roll_belt                547.9525674
## pitch_belt               305.4059679
## yaw_belt                 337.5319011
```

```
## total_accel_belt         121.3632181
## gyros_belt_x              39.6053619
## gyros_belt_y              48.7136869
## gyros_belt_z             127.2129618
## accel_belt_x              62.0404653
## accel_belt_y              68.5993222
## accel_belt_z             164.8149573
## magnet_belt_x            109.7122624
## magnet_belt_y            202.2257954
## magnet_belt_z            178.0977536
## roll_arm                 117.5298012
## pitch_arm                 50.3171955
## yaw_arm                   66.6723589
## total_accel_arm           28.3955950
## gyros_arm_x               43.3969180
## gyros_arm_y               40.3596444
## gyros_arm_z               19.1558967
## accel_arm_x               93.6602350
## accel_arm_y               50.0627278
## accel_arm_z               41.3949660
## magnet_arm_x              84.8326552
## magnet_arm_y              78.9193052
## magnet_arm_z              56.3410701
## roll_dumbbell            197.9402373
## pitch_dumbbell            88.1126847
## yaw_dumbbell             114.8917737
## total_accel_dumbbell     103.5833876
## gyros_dumbbell_x          38.6548790
## gyros_dumbbell_y          97.9766291
## gyros_dumbbell_z          24.8102676
## accel_dumbbell_x         116.4273466
## accel_dumbbell_y         188.2112109
## accel_dumbbell_z         141.6149128
## magnet_dumbbell_x        242.4551840
## magnet_dumbbell_y        326.1604375
## magnet_dumbbell_z        318.1683512
## roll_forearm             234.4663771
## pitch_forearm            305.3700749
## yaw_forearm               51.7827778
## total_accel_forearm       31.2361864
## gyros_forearm_x           24.2570230
## gyros_forearm_y           36.8098275
## gyros_forearm_z           25.4767234
## accel_forearm_x          134.2375365
## accel_forearm_y           44.3531671
## accel_forearm_z           85.8674199
## magnet_forearm_x          73.4708187
## magnet_forearm_y          71.6173900
## magnet_forearm_z          92.0223654
```

Next, the model results is evaluated through the confusion Matrix.

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

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    A    B    C    D    E
##         A 2230    1    0    0    0
##         B    2 1517    6    0    0
##         C    0    0 1359    5    0
##         D    0    0    3 1281    0
##         E    0    0    0    0 1442
##
## Overall Statistics
##
##               Accuracy : 0.9978
##                 95% CI : (0.9965, 0.9987)
##    No Information Rate : 0.2845
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.9973
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                     Class: A Class: B Class: C Class: D Class: E
## Sensitivity           0.9991   0.9993   0.9934   0.9961   1.0000
## Specificity           0.9998   0.9987   0.9992   0.9995   1.0000
## Pos Pred Value        0.9996   0.9948   0.9963   0.9977   1.0000
## Neg Pred Value        0.9996   0.9998   0.9986   0.9992   1.0000
## Prevalence            0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate        0.2842   0.1933   0.1732   0.1633   0.1838
## Detection Prevalence   0.2843   0.1944   0.1738   0.1637   0.1838
## Balanced Accuracy     0.9995   0.9990   0.9963   0.9978   1.0000
```

The accurancy for the validating data set is calculated with the following formula:

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

Model Accuracy as tested over Validation set = 99.73% The out-of-sample error is only 0.17%.

Model Test

For the model testing, the new values are predicted using the testing dataset provided which was loaded earlier. Data cleaning was first performed and all columns of Testing data set are coerced for the same class of previous data set.

```
testRaw <- testRaw[,-1] # Remove the first column that represents a ID R
ow
testRaw <- testRaw[ , Keep] # Keep the same columns of testing dataset
testRaw <- testRaw[,-ncol(testRaw)] # Remove the problem ID
```

Transformations and Coercing of Testing Dataset

```
# Coerce testing dataset to same class and structure of training dataset

testing <- rbind(training[100, -59] , testRaw)

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

Prediction with the Testing Dataset

```
predictions <- predict(model,newdata=testing[-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
```

We submitted these prediction ot the quzz and all are correct.