Prediction Assignment Writeup

S.T.S. Lin

Sunday, February 22, 2015

Background Introduction

This is the written report part of the final project of Coursera¡¦s MOOC Practical Machine Learning from Johns Hopkins University. Here is the background provided: "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 ¡V 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 Sources

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)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Method

- Clean up the data and select and remove columns that contain too much invalid or incomplete data (Steps 5 ~ 10).
- Then determine whether Decision Tree (Steps 11 ~ 14) or Random Forests (Steps 15 ~ 17) should be used by choosing one with the lower out of sample error.
- Use the choesen method (Decision Tree or Random Forest) to predict the test data.

Out of Sample error and Cross-validation

The training data set is provided is seperated into 60% for training and 40% for interal testing in "Step 4". I expect the out of sample error to be well less than 20% (accuracy rate of 80% or higher) due to the extensive efforts put into cleaning up the data. Confusion Matrix was used in Steps 14 and 17 to determine to accuracy rates (the opposite of error rates) with 0.8789 accuracy (1 - 0.8789 error) for Decision Tree and 0.9986 accuracy (1 - 0.9986 error) for Random Forests.

Data Proccessing

0. Set system locale to English.

```
Sys.setlocale(category = "LC_ALL", locale = "English")
```

```
## [1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=Engl
ish_United States.1252;LC_NUMERIC=C;LC_TIME=English_United States.1252"
```

1. Load Libraries

```
if (!require("caret")) {
  install.packages("caret")
}
```

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

```
if (!require("RGtk2")) {
  install.packages("RGtk2")
}
```

```
## Loading required package: RGtk2
```

```
if (!require("rattle")) {
  install.packages("rattle")
}
```

```
## Loading required package: rattle
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
if (!require("rpart")) {
  install.packages("rpart")
}
```

```
## Loading required package: rpart
```

```
if (!require("rpart.plot")) {
  install.packages("rpart.plot")
}
```

```
## Loading required package: rpart.plot
```

```
if (!require("randomForest")) {
  install.packages("randomForest")
}
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
if (!require("e1071")) {
  install.packages("e1071")
}
```

```
## Loading required package: e1071
```

```
require("caret")
require("RGtk2")
require("rattle")
require("rpart")
require("rpart.plot")
require("randomForest")
require("e1071")

library(caret)
library(RGtk2)
library(rattle)
library(rpart)
library(rpart.plot)
library(randomForest)
library(e1071)
```

2. Set seed value

```
set.seed(12345)
```

3. Read data to memoruy.

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

4. Partioning training data set into:

- 60% for interal training (intTrain)
- 40% for interal testing (intTest) a.k.a. Validation Testing

```
dataPartTrain <- createDataPartition(y=dataTrain$classe, p=0.6, list=FALSE)
intTrain <- dataTrain[dataPartTrain, ]
intTest <- dataTrain[-dataPartTrain, ]
```

Cleaning The Data

5. Generating list of NZV (Near Zero Variables) columns

```
intDataNZV <- nearZeroVar(dataTrain, saveMetrics=TRUE)
intDataNZVList <- intDataNZV[intDataNZV$nzv,]
```

6. Removing the NZV (Near Zero Variables) columns from data:

```
intNZVColumns <- names(intTrain) %in% rownames(intDataNZVList)
intTrain <- intTrain[!intNZVColumns]
```

7. Removing first column (ID) so that it does not interfer with ML Algorithms:

```
intTrain$X <- NULL
```

8. Remove columns that have too many NA values (more than a threshold of 60%)

```
intTrain <- intTrain[,colSums(is.na(intTrain)) < (nrow(intTrain)*0.6)]
```

9. Remove columns in test data that do not exist in training data

```
intTest <- intTest[, names(intTest) %in% names(intTrain)]
dataTest <- dataTest[, names(dataTest) %in% names(intTrain)]
```

10. “Syncronize” the data type/category in the data and initTrain data frames.

```
dataTest$classe <- 0
dataTest <- rbind(intTrain[1,] , dataTest)
```

```
## Warning in `[<-.factor`(`*tmp*`, ri, value = c(0, 0, 0, 0, 0, 0, 0, 0, 0,
## : invalid factor level, NA generated
```

```
dataTest <- dataTest[-1,]
dataTest$classe <- NULL
```

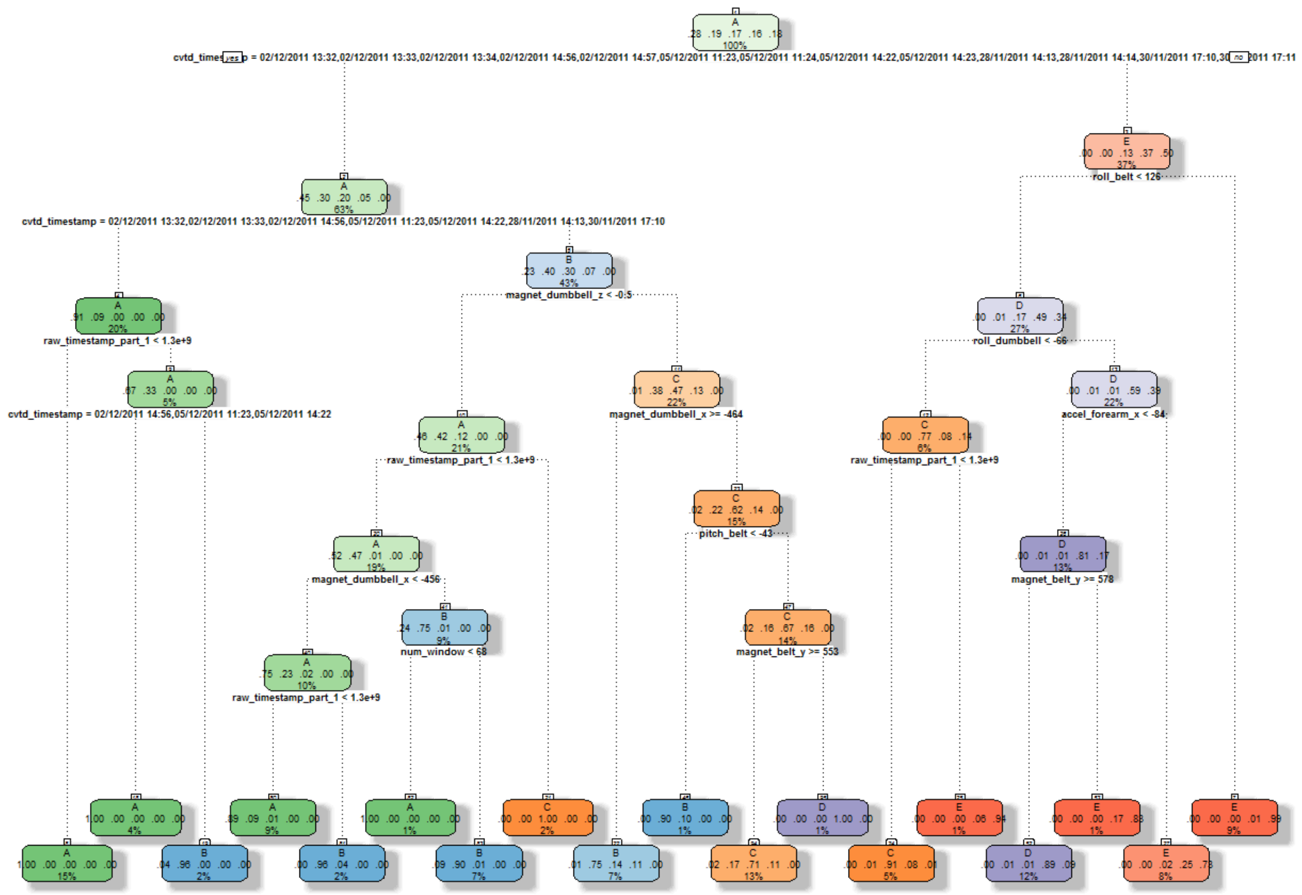
Machine Learning - Decision Tree

11. Apply decision tree machine learning.

```
modFitDT <- rpart(classe ~ ., data=intTrain, method="class")
```

12. View of the decision tree with fancy.

```
fancyRpartPlot(modFitDT)
```


cvtd_timestamp = 02/12/2011 13:32,02/12/2011 13:33,02/12/2011 13:34,02/12/2011 14:56,02/12/2011 14:57,05/12/2011 11:23,05/12/2011 11:24,05/12/2011 14:22,05/12/2011 14:23,28/11/2011 14:13,28/11/2011 14:14,30/11/2011 17:10,30/11/2011 17:11
yes
no
roll_belt < 126
cvtd_timestamp = 02/12/2011 13:32,02/12/2011 13:33,02/12/2011 14:56,05/12/2011 11:23,05/12/2011 14:22,28/11/2011 14:13,30/11/2011 17:10
magnet_dumbbell_z < -0.5
raw_timestamp_part_1 < 1.3e+9
roll_dumbbell < -66
cvtd_timestamp = 02/12/2011 14:56,05/12/2011 11:23,05/12/2011 14:22
magnet_dumbbell_x >= -464
accel_forearm_x < -84
raw_timestamp_part_1 < 1.3e+9
pitch_belt < -43
magnet_dumbbell_x < -456
magnet_belt_y >= 578
num_window < 68
magnet_belt_y >= 553
raw_timestamp_part_1 < 1.3e+9

Rattle 2015-Feb-23 04:21:04 shawnlin

13. Apply decision tree machine prediction.

```
predictionsDT <- predict(modFitDT, intTest, type = "class")
```

14. Test Results of decision tree with Confusion Matrix.

```
confusionMatrix(predictionsDT, intTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

Machine Learning - Random Forests

15. Apply random forests machine learning.

```
modFitRF <- randomForest(classe ~. , data=intTrain)
```

16. Apply random forests machine prediction.

```
predictionsRF <- predict(modFitRF, intTest, type = "class")
```

17. Test Results of random forests with Confusion Matrix.

```
confusionMatrix(predictionsRF, intTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    2    0    0
##          C    0    0 1366    3    0
##          D    0    0    0 1282    2
##          E    0    0    0    1 1440
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9975, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9985   0.9969   0.9986
## Specificity            0.9996   0.9995   0.9995   0.9997   0.9998
## Pos Pred Value         0.9991   0.9980   0.9978   0.9984   0.9993
## Neg Pred Value         0.9998   0.9997   0.9997   0.9994   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1741   0.1634   0.1835
## Detection Prevalence   0.2846   0.1936   0.1745   0.1637   0.1837
## Balanced Accuracy      0.9996   0.9991   0.9990   0.9983   0.9992
```

Predicting Test Data

Confusion Matrix was used in Steps 14 and 17 to determine to accuracy rates (the opposite of error rates) with 0.8789 accuracy (1 - 0.8789 error) for Decision Tree and 0.9986 accuracy (1 - 0.9986 error) for Random Forests.

Thus, Random Forest is selected to predict the test data to to its lower out of sample error.

```
predictionsRF <- predict(modFitRF, dataTest, type = "class")
```

Generate Files with Predictions to Submit for Assignment:

Provided by Assignment to Generate Solutions Files for Uploading

```
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)
  }
}
```

Call function Provided by Assignment to Generate Solutions Files for Uploading

```
pml_write_files(predictionsRF)
```