Final Project Report - Practical Machine Learning Course

Author: Jared Aubry

Background Introduction

These are the files produced during a homework assignment of Coursera's MOOC Practical Machine Learning from Johns Hopkins University. Here is the introduction of the exercise:

"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: 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

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project comes from this original source: 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.

Please Note that I the code I use loads the data directly from the URL provided, so that you are not required to download the file to your environment. Please customize the code to your specific needs.

Project Intended Results

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).
2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Author initial note

Please consult the explorAnalysis.R (in code/raw code) file in the gitHub repo in order to better understand the reasoning behind the tatics choosen. For instance and as a very simple example, the initial loading of data to memory involves assuming some values as NA. For obvious reasons, this is only possible after you have already pooked around the data initially.

Reproduceablity

In order to reproduce the same results, you need a certain set of packages, as well as setting a pseudo random seed equal to the one I used. *Note:To install, for instance, the caret package in R, run this command: install.packages("caret")

The following Libraries were used for this project, which you should install - if not done yet - and load on your working environment.

```
library(caret)
```

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

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.0.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

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

Finally, load the same seed with the following line of code:

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

Getting the data

The training data set can be found on the following URL:

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

The testing data set can be found on the following URL:

```
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Procedure 1) Procedure 1) assumes that you only want to store the data files in memory.

Load data to memory solely

```
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Procedure 2) Procedure 2) assumes that you want to store the data files in memory and on disk. (Thus, it involves downloading data directly to your hard drive.)

You can use following function to download the data:

```
#getDataFiles <- function(filesDirectory = "./") {
#        if (!file.exists(filesDirectory)) {
#     dir.create(filesDirectory)
#           }
#        testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#        trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#        trainFile <- "train.csv"
#        testFile <- "test.csv"
#   trainFilePath <- paste(filesDirectory, trainFile, sep = "/")
#        testFilePath <- paste(filesDirectory, testFile, sep = "/")
#        download.file(trainUrl, destfile = trainFilePath, method="curl")
#        download.file(testUrl, destfile = testFilePath, method="curl")
#        training <- read.csv(trainFilePath, na.strings=c("NA","#DIV/0!",""))
#        testing <- read.csv(testFilePath, na.strings=c("NA","#DIV/0!",""))
#}
```

Run the function, for example, as follows:

```
#getDataFiles("/data")
```

Note that you can simply run the function without passing any argument, which means the file will be downloaded to your current working directory. Note: To view your current working directory run the following command:

```
getwd()
```

```
## [1] "C:/Users/Jared/Desktop/Coursera/Practical Machine Learning/PracticalMachineLearning"
```

Partioning the training set into two

Partioning Training data set into two data sets, 60% for myTraining, 40% for myTesting:

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Cleaning the data

The following transformations were used to clean the data:

Transformation 1: Cleaning NearZeroVariance Variables Run this code to view possible NZV Variables:

```
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
```

Run this code to create another subset without NZV variables:

```
myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll_dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_dumbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]
#To check the new N?? of observations
dim(myTraining)
```

```
## [1] 11776   100
```

Transformation 2: Killing first column of Dataset - ID Removing first ID variable so that it does not interfer with ML Algorithms:

```
myTraining <- myTraining[c(-1)]
```

Transformation 3: Cleaning Variables with too many NAs. For Variables that have more than a 60% threshold of NA's I'm going to leave them out:

```
trainingV3 <- myTraining #creating another subset to iterate in loop
for(i in 1:length(myTraining)) { #for every column in the training dataset
        if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 ) { #if n?? NAs > 60% of total obse
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==1)  { #if the columns are
                trainingV3 <- trainingV3[ , -j] #Remove that column
            }
        }
    }
}
#To check the new N?? of observations
dim(trainingV3)
```

```
## [1] 11776    58
```

```
#Seting back to our set:
myTraining <- trainingV3
rm(trainingV3)
```

Now let us do the exact same 3 transformations but for our myTesting and testing data sets.

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58]) #already with classe column removed
myTesting <- myTesting[clean1]
testing <- testing[clean2]

#To check the new N?? of observations
dim(myTesting)
```

```
## [1] 7846   58
```

```
#To check the new N?? of observations
dim(testing)
```

```
## [1] 20 57
```

```
#Note: The last column - problem_id - which is not equal to training sets, was also "automagically" rem
#No need for this code:
#testing <- testing[-length(testing)]
```

In order to ensure proper functioning of Decision Trees and especially RandomForest Algorithm with the Test data set (data set provided), we need to coerce the data into the same type.

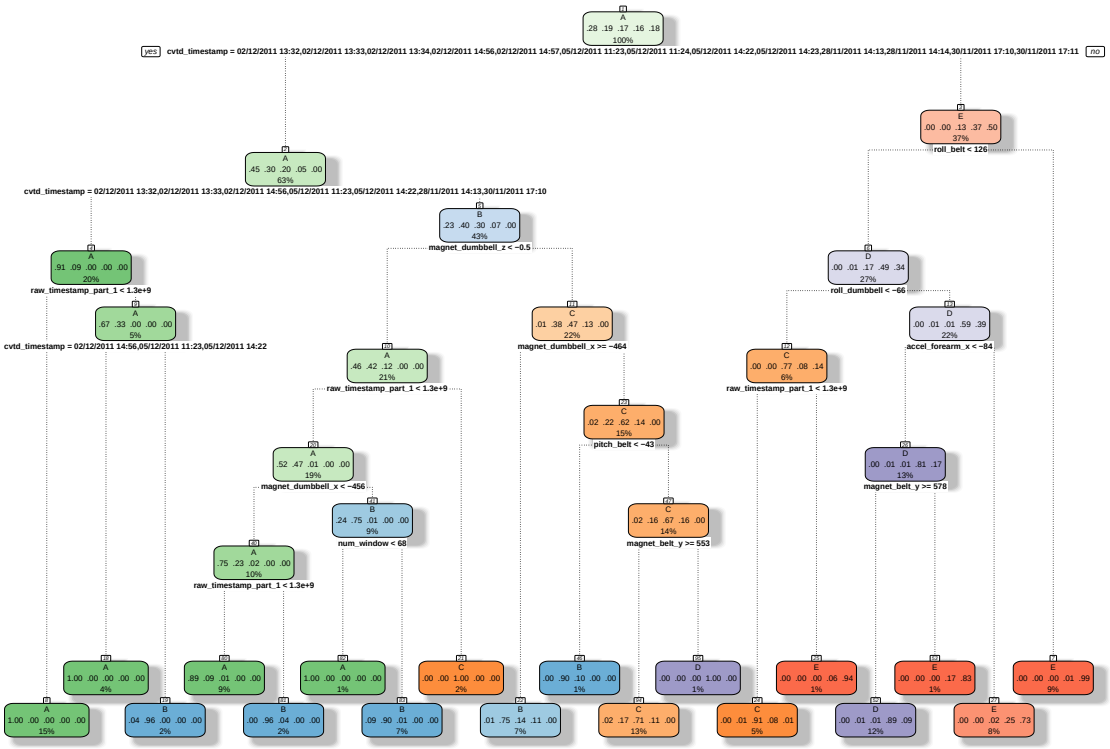
```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}
#And to make sure Coertion really worked, simple smart ass technique:
testing <- rbind(myTraining[2, -58] , testing) #note row 2 does not mean anything, this will be removed
testing <- testing[-1,]
```

Using ML algorithms for prediction: Decision Tree

```
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
```

Note: to view the decision tree with fancy run this command:

```
fancyRpartPlot(modFitA1)
```



Rattle 2015−Nov−21 15:53:09 Jared

Predicting:

```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
```

(Moment of truth) Using confusion Matrix to test results:

```
confusionMatrix(predictionsA1, myTesting$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
```

```
#Overall Statistics

#               Accuracy : 0.8683
#                 95% CI : (0.8607, 0.8757)
#    No Information Rate : 0.2845
#    P-Value [Acc > NIR] : < 2.2e-16

#                  Kappa : 0.8335
```

Using ML algorithms for prediction: Random Forests

```
modFitB1 <- randomForest(classe ~. , data=myTraining)
```

Predicting:

```
predictionsB1 <- predict(modFitB1, myTesting, type = "class")
```

(Moment of truth) Using confusion Matrix to test results:

```
confusionMatrix(predictionsB1, myTesting$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
```

```
#Overall Statistics

 #               Accuracy : 0.999
 #                 95% CI : (0.998, 0.9996)
 #    No Information Rate : 0.2845
 #    P-Value [Acc > NIR] : < 2.2e-16

 #                  Kappa : 0.9987
 #Mcnemar's Test P-Value : NA
```

Random Forests yielded better Results, as expected!

Generating Files to submit as answers for the Assignment:

Finally, using the provided Test Set: Note:

For Decision Tree would be like this, but not going to use it: predictionsA2 <- predict(modFitA1, testing, type = "class")

For Random Forests is, which yielded a much better prediction:

```
predictionsB2 <- predict(modFitB1, testing, type = "class")
```

Function to generate files with predictions to submit for assignment

```
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)
  }
}

pml_write_files(predictionsB2)
```