Project - final

SS

Friday, August 21, 2015

PRACTICAL MACHINE LEARNING COURSE PROJECT

SYNOPSIS

In the study we will discuss in this paper (http://groupware.les.inf.puc-rio.br/har), they investigated the use of computing to evaluate "proper" exercise form (possibly allowing computers to replace personal trainers to help us become better, faster, stronger.

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

moved data to the Data folder in the same repo

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.

item 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 :-).

item 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.

```
library(caret)

## Warning: package 'caret' was built under R version 3.1.3

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

library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.1.3

library(RColorBrewer)
library(rattle)

## Warning: package 'rattle' was built under R version 3.1.3

## Loading required package: RGtk2

## Warning: package 'RGtk2' was built under R version 3.1.3

## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.1.3

## randomForest 4.6-10
## 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 and testing data set can be found on the following URLs:

```
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

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

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

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

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]
```

```
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 observations
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) )
==1)  { #if the columns are the same:
                trainingV3 <- trainingV3[ , -j] #Remove that column
            }
        }
    }
}

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
```

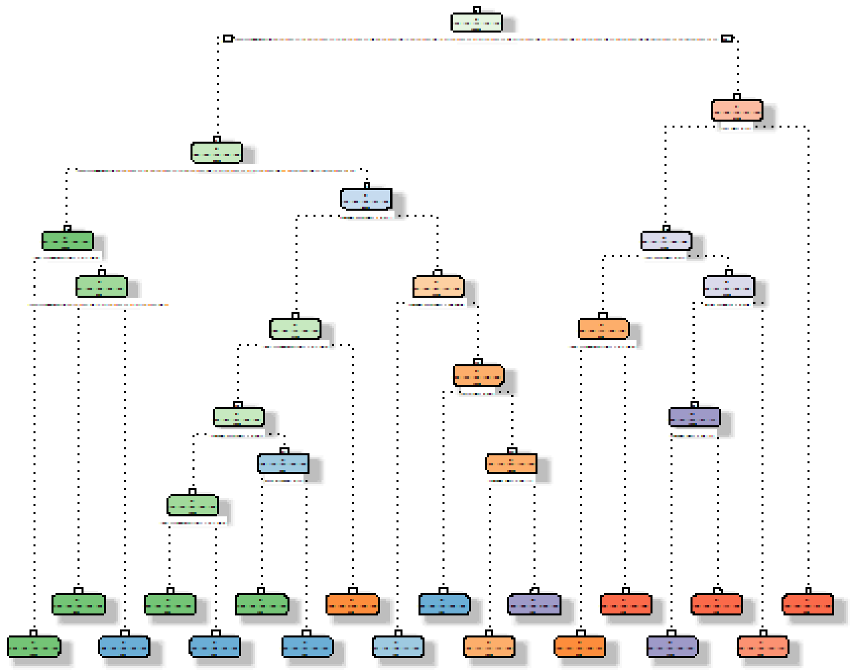
In order to ensure proper functioning of Decision Trees and especially RandomForest Algorithm with the Test data set (data set provided), we need to coerce the data into the same type.

```
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 right.. now:
testing <- testing[-1,]
```

Using ML algorithms for prediction: Decision Tree

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



Rattle 2015-Aug-21 19:09:22 ssouidi

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
```

CONCLUSION

I received three separate predictions by appling the 4 models against the actual 20 item training set:

A) Accuracy Rate 0.0286 Predictions: B A A A A E D B A A B C B A E E A B B B

B) Accuracy Rates 0.0366 and 0.0345 Predictions: B A B A A E D B A A B C B A E E A B B B

C) Accuracy Rate 0.0437 Predictions: B A B A A E D D A A B C B A E E A B B B