Practical Machine Learning

GHyee

Sunday, February 08, 2015

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. In this project, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict how well they perform barbell lifts. 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 Processing

The data source comes from <http://groupware.les.inf.puc-rio.br/har> and are already split into Train and Test set beforehand. The Train dataset can be downloaded from <https://d396qusza40orc.cloudfront.net/predmachlearn/pml-Train.csv> while the Test dataset can be downloaded from <https://d396qusza40orc.cloudfront.net/predmachlearn/pml-Test.csv>. The data are processed after some exploratory analysis before models are fitted.

#Load required library  
library(caret)

## Warning: package 'caret' was built under R version 3.1.2

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

library(rpart)  
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.1.2

library(RColorBrewer)  
library(rattle)

## Warning: package 'rattle' was built under R version 3.1.2

## 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.

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.1.2

## randomForest 4.6-10  
## Type rfNews() to see new features/changes/bug fixes.

####################################  
#############Load Data#############  
####################################  
#Download data from URL  
#trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-Train.csv"  
#testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-Test.csv"  
#Data have been downloaded to reduce runtime of the code  
#Import data into R  
Train <- read.csv("./pml-training.csv", na.strings=c("NA","#DIV/0!",""))  
Test <- read.csv("./pml-testing.csv", na.strings=c("NA","#DIV/0!",""))  
  
#Create partition on Train set  
inTrain <- createDataPartition(y=Train$classe, p=0.6, list=FALSE)  
myTrain <- Train[inTrain, ]; myTest <- Train[-inTrain, ]  
dim(myTrain); dim(myTest)

## [1] 11776 160

## [1] 7846 160

#Remove variables with near zero variance   
myDataNZV <- nearZeroVar(myTrain, saveMetrics=TRUE)  
  
myNZVvars <- names(myTrain) %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")  
#Remove variables with near zero variance  
myTrain <- myTrain[!myNZVvars]  
#Check dimension of subset  
dim(myTrain)

## [1] 11776 100

#Remove ID column (1)  
myTrain <- myTrain[c(-1)]  
#Remove variables with more than 60% of NAs  
TrainTemp <- myTrain #creating another subset to iterate in loop  
for(i in 1:length(myTrain)) { #for every column in the Train dataset  
 if( sum( is.na( myTrain[, i] ) ) /nrow(myTrain) >= .6 ) { #if n?? NAs > 60% of total observations  
 for(j in 1:length(TrainTemp)) {  
 if( length( grep(names(myTrain[i]), names(TrainTemp)[j]) ) ==1) { #if the columns are the same:  
 TrainTemp <- TrainTemp[ , -j] #Remove that column  
 }   
 }   
 }  
}  
#To check the new N?? of observations  
dim(TrainTemp)

## [1] 11776 58

#Seting back to our set:  
myTrain <- TrainTemp  
rm(TrainTemp)  
#Obtain variable names that are not removed earlier  
clean1 <- colnames(myTrain)  
clean2 <- colnames(myTrain[, -58]) #already with classe column removed  
#Remove the variables from myTest set  
myTest <- myTest[clean1]  
Test <- Test[clean2]  
  
#To check the new N?? of observations  
dim(myTest)

## [1] 7846 58

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

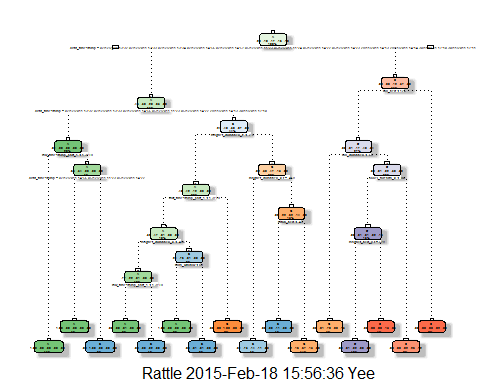
## [1] 20 57

#Coerce Test set to match Train set  
for (i in 1:length(Test) ) {  
 for(j in 1:length(myTrain)) {  
 if( length( grep(names(myTrain[i]), names(Test)[j]) ) ==1) {  
 class(Test[j]) <- class(myTrain[i])  
 }   
 }   
}  
#And to make sure Coertion really worked, simple smart ass technique:  
Test <- rbind(myTrain[2, -58] , Test) #note row 2 does not mean anything, this will be removed right.. now:  
Test <- Test[-1,]  
#########################################################################  
#######################End of Data Processing############################  
#########################################################################

# Model Fitting

Two machine learning method, namely the decision tree and Random forest, will be used to fit the model. The method with the better accuracy will be chosen. Decision tree is fitted using the rpart function from the rpart library. Random forest is fitted using the randomForest function from the randomForest package.

#Set set for reproducibility  
set.seed(20151802)  
modFitA1 <- rpart(classe ~ ., data=myTrain, method="class")  
  
fancyRpartPlot(modFitA1)



predictionsA1 <- predict(modFitA1, myTest, type = "class")  
  
confusionMatrix(predictionsA1, myTest$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2152 70 8 2 0  
## B 59 1260 71 67 0  
## C 21 180 1275 198 62  
## D 0 8 10 801 78  
## E 0 0 4 218 1302  
##   
## Overall Statistics  
##   
## Accuracy : 0.865   
## 95% CI : (0.858, 0.873)  
## No Information Rate : 0.284   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.83   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.964 0.830 0.932 0.623 0.903  
## Specificity 0.986 0.969 0.929 0.985 0.965  
## Pos Pred Value 0.964 0.865 0.734 0.893 0.854  
## Neg Pred Value 0.986 0.960 0.985 0.930 0.978  
## Prevalence 0.284 0.193 0.174 0.164 0.184  
## Detection Rate 0.274 0.161 0.163 0.102 0.166  
## Detection Prevalence 0.284 0.186 0.221 0.114 0.194  
## Balanced Accuracy 0.975 0.899 0.930 0.804 0.934

modFitB1 <- randomForest(classe ~. , data=myTrain)  
  
predictionsB1 <- predict(modFitB1, myTest, type = "class")  
  
confusionMatrix(predictionsB1, myTest$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2232 1 0 0 0  
## B 0 1517 1 0 0  
## C 0 0 1366 6 0  
## D 0 0 1 1280 3  
## E 0 0 0 0 1439  
##   
## Overall Statistics  
##   
## Accuracy : 0.998   
## 95% CI : (0.997, 0.999)  
## No Information Rate : 0.284   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.998   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.000 0.999 0.999 0.995 0.998  
## Specificity 1.000 1.000 0.999 0.999 1.000  
## Pos Pred Value 1.000 0.999 0.996 0.997 1.000  
## Neg Pred Value 1.000 1.000 1.000 0.999 1.000  
## Prevalence 0.284 0.193 0.174 0.164 0.184  
## Detection Rate 0.284 0.193 0.174 0.163 0.183  
## Detection Prevalence 0.285 0.193 0.175 0.164 0.183  
## Balanced Accuracy 1.000 1.000 0.999 0.997 0.999

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

The second model, Random Forest, will be chosen as it yields a much better accuracy compared to the first.

The test data set is used to validate the accuracy of the Random forest model.

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)