Machine Learning

Ratan Jha

December 25, 2015

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

#Machine Learning Project Work  
#Author Ratan Jha  
#First set directory to downloaded folder  
# Source of this data is "http://groupware.les.inf.puc-rio.br/har"  
#http://groupware.les.inf.puc-rio.br/har#sbia\_paper\_section  
# Download and save the file in a working directory  
setwd("~/RforRatan/practicalmachinelearning")  
# read training data  
maindata <- read.csv("pml-training.csv")  
#set seed fro repeatable results  
set.seed(32343)  
#load caret library  
library(caret)

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

library(randomForest)

## randomForest 4.6-12  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:ggplot2':  
##   
## margin

library(e1071)  
#Let Us turnoff warning messages for knitting to avoid seeing those messages  
options(warn=-1)  
# Upon inspecting the data, we find that first 5 columns which contains  
# Since the number of predictors are huge (150) in the file,  
# There is no way to plot and assess which variables are related more closely  
# The other challenge since the output is a factor, it will not be right to do a   
# a correlation analysis to remove variables which are strongly correlated with each   
# other as well as weak relationship with the output variable  
# use name, time stamp, window fields which is not a predictor variable  
# first let us remove the first 5 fields  
sourcedata <- maindata [,6:160]  
# further investigation shows that data has blanks, NA and "#Div/o!"  
# Let us remove these values with "NA"  
sourcedata[sourcedata == ""] <- "NA"  
sourcedata[sourcedata == "#DIV/0!"] <- "NA"  
# Only keep columns that have less than 40% NA in the columns  
sourcedata <-  
 sourcedata[, colSums(is.na(sourcedata)) < nrow(sourcedata) \* 0.40]  
#sourcedata <- sapply(sourcedata,as.numeric)  
sourcedata$classe <- maindata$classe  
write.csv(sourcedata,"InterimResults.csv")  
#Split the training data into training and test set (75 to 25 ratio - since the sample size is large)  
inTrain <-  
 createDataPartition(sourcedata$classe,p = 0.75,list = FALSE)  
training <- sourcedata[inTrain,]  
testing <- sourcedata[-inTrain,]  
# Set the class as a factor for analysis  
training$classe <- as.factor(as.character(training$classe))  
testing$classe <- as.factor(as.character(testing$classe))  
# used randomForest function instead of train function as it happened to be many times faster  
RFModel <- randomForest(classe ~ .,data = training)  
#Check Class Error and Estimate of Error Rate  
print(RFModel)

##   
## 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: 0.24%  
## Confusion matrix:  
## A B C D E class.error  
## A 4184 0 0 0 1 0.0002389486  
## B 6 2841 1 0 0 0.0024578652  
## C 0 9 2558 0 0 0.0035060382  
## D 0 0 15 2395 2 0.0070480929  
## E 0 0 0 2 2704 0.0007390983

# Based on the model estimate of error on the test data will be 0.24%  
# That would translate accurace of 99.76% ( A very high accuracy)  
RFR <- predict(RFModel,testing)  
RFC <- confusionMatrix(RFR,testing$classe)$overall[1]  
# Check Accuracy for out   
print(RFC)

## Accuracy   
## 0.9963295

#random forest provided 99.6% accurate model while rpart gave only 49.8%. (removed code for rpart)  
#Hence we would leverage random forest for predicting  
#Read test Data  
testdata <- read.csv("pml-testing.csv")  
newtestdata <- training  
#Only keep columns that are present in trainign data  
newtestdata <- testdata[,names(testdata) %in% names(training)]  
#since test data doesn't have classe column defined, we will add the columne  
newtestdata$classe <- training$classe[1:1]  
levels(newtestdata) <- levels(training)  
#This code is to ensure that new\_window has 2 factor variable as the training data has two values  
levels(newtestdata$new\_window) <- c("yes","no")  
RPartFinalResult <- predict(RFModel,newtestdata)  
#getoutput in just problem\_id and Classes  
answers <- data.frame()  
# get problem\_id from testdata & results from RPartFinalResult  
answers <- subset(testdata,select = (problem\_id))  
answers$classes <- RPartFinalResult  
print(answers)

## problem\_id classes  
## 1 1 B  
## 2 2 A  
## 3 3 B  
## 4 4 A  
## 5 5 A  
## 6 6 E  
## 7 7 D  
## 8 8 B  
## 9 9 A  
## 10 10 A  
## 11 11 B  
## 12 12 C  
## 13 13 B  
## 14 14 A  
## 15 15 E  
## 16 16 E  
## 17 17 A  
## 18 18 B  
## 19 19 B  
## 20 20 B

# End of Programming

You can also embed plots, for example:

Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.