Practical Machine Learning Coursera Project

r1

Saturday, June 20, 2015

# Summary

This is an R Markdown document. For a Coursera project for Practical Machine Learning. The goal of this project is to predict and classify the type of CLASS (A,B,C,D,E) on the record based on the training and test sets provided by the course project.In essence the predition will establish a pattern based on existing data and predict the CLASS for 20 records in a testing set.

# Background notes Provided by the course Assisgnment

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.

# Data

The training data for this project was initially available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>

The test data are available here:

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

# Initial Talking Point

This data has been downloaded by the developer in the working directory already. The data attributes in the training set with values "NA" have already been analyzed and the list is prepared.

# The Sequence of this assisgnment

1. Load the Data Training and Test dataset from url
2. Cleaning Data for both "Training" and "Test" 2a. Replace DIV/0 with NA 2b. Remove near zero values and Nulls
3. Create Data Partition for "Training" set 3a. Training 3b. Valuation
4. PreProc, Train and Predict with Training set
5. Evaluate and Crossvalidate with valuation data.
6. Apply Test data and Predict

# Reproducible Code:

The list of libraries to be imported

library(rpart)  
library(rpart.plot)

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

library(RColorBrewer)

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

library(rattle)

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

## 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(caret)

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

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

## Warning: package 'ggplot2' was built under R version 3.1.2

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.

## Load Training and test set

THis loads the data and replaces any values with #DIV/0

## Cleaning the data

Removing Near Zero values and nulls from the training and test datasets, This set of null varibles were pre-analyzed. In addition to the null columns I have removed the first 7 columns that do not add any value for prediction and remove noise.

mynulls <- names(training) %in% c("X","user\_name","raw\_timestamp\_part\_1","raw\_timestamp\_part\_2","cvtd\_timestamp","new\_window","num\_window","kurtosis\_roll\_belt","kurtosis\_picth\_belt","kurtosis\_yaw\_belt","skewness\_roll\_belt","skewness\_roll\_belt.1","skewness\_yaw\_belt","max\_roll\_belt","max\_picth\_belt","max\_yaw\_belt","min\_roll\_belt","min\_pitch\_belt","min\_yaw\_belt","amplitude\_roll\_belt","amplitude\_pitch\_belt","amplitude\_yaw\_belt","var\_total\_accel\_belt","avg\_roll\_belt","stddev\_roll\_belt","var\_roll\_belt","avg\_pitch\_belt","stddev\_pitch\_belt","var\_pitch\_belt","avg\_yaw\_belt","stddev\_yaw\_belt","var\_yaw\_belt","var\_accel\_arm","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","max\_picth\_arm","max\_yaw\_arm","min\_roll\_arm","min\_pitch\_arm","min\_yaw\_arm","amplitude\_roll\_arm","amplitude\_pitch\_arm","amplitude\_yaw\_arm","kurtosis\_roll\_dumbbell","kurtosis\_picth\_dumbbell","kurtosis\_yaw\_dumbbell","skewness\_roll\_dumbbell","skewness\_pitch\_dumbbell","skewness\_yaw\_dumbbell","max\_roll\_dumbbell","max\_picth\_dumbbell","max\_yaw\_dumbbell","min\_roll\_dumbbell","min\_pitch\_dumbbell","min\_yaw\_dumbbell","amplitude\_roll\_dumbbell","amplitude\_pitch\_dumbbell","amplitude\_yaw\_dumbbell","var\_accel\_dumbbell","avg\_roll\_dumbbell","stddev\_roll\_dumbbell","var\_roll\_dumbbell","avg\_pitch\_dumbbell","stddev\_pitch\_dumbbell","var\_pitch\_dumbbell","avg\_yaw\_dumbbell","stddev\_yaw\_dumbbell","var\_yaw\_dumbbell","kurtosis\_roll\_forearm","kurtosis\_picth\_forearm","kurtosis\_yaw\_forearm","skewness\_roll\_forearm","skewness\_pitch\_forearm","skewness\_yaw\_forearm","max\_roll\_forearm","max\_picth\_forearm","max\_yaw\_forearm","min\_roll\_forearm","min\_pitch\_forearm","min\_yaw\_forearm","amplitude\_roll\_forearm","amplitude\_pitch\_forearm","amplitude\_yaw\_forearm","var\_accel\_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 <- training[!mynulls]  
myTest <- test[!mynulls]

## Creating data partition, I have taken 70% for "intrain" and the rest for the "valuation", We will drop "classe" column from our train set before preproc

set.seed(123456)  
  
inTrain <- createDataPartition(y = myTraining$classe, p = 0.7, list = F)  
train <- myTraining[inTrain,]  
valuation <- myTraining[-inTrain,]  
  
numericreturnofremovedvalues <- function(x, classcolumn)  
{  
 indices <- -(which(names(x) %in% classcolumn))  
 return (indices)  
}  
  
dropclasse <- names(train) %in% c("classe")  
  
dropclasseP <- myTraining[!dropclasse]

## PreProc, Train and Predict with Training set

preProc <- preProcess(train[, numericreturnofremovedvalues(train, c("classe"))], method="pca", pcaComp="30")  
  
train\_main <- predict(preProc, train[, numericreturnofremovedvalues(train, c("classe"))])  
  
modfit <- randomForest(train$classe ~ ., data = train\_main, ntree=200)  
  
valid <- predict(preProc, valuation[, numericreturnofremovedvalues(valuation, c("classe"))])  
  
prediction <- predict(modfit, valid)

## The requirement of the written assignement is to provide a crossvalidation and outofsampleerror, below are the three analysis for the three aspects

## Printing Confusion Matrix and crossvalidation and outofsampleerror

print(confusionMatrix(prediction, valuation$classe), digits=4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1670 13 2 0 0  
## B 0 1116 13 0 9  
## C 2 9 1002 38 10  
## D 1 0 6 923 6  
## E 1 1 3 3 1057  
##   
## Overall Statistics  
##   
## Accuracy : 0.9801   
## 95% CI : (0.9762, 0.9835)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9748   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9976 0.9798 0.9766 0.9575 0.9769  
## Specificity 0.9964 0.9954 0.9879 0.9974 0.9983  
## Pos Pred Value 0.9911 0.9807 0.9444 0.9861 0.9925  
## Neg Pred Value 0.9990 0.9952 0.9950 0.9917 0.9948  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2838 0.1896 0.1703 0.1568 0.1796  
## Detection Prevalence 0.2863 0.1934 0.1803 0.1590 0.1810  
## Balanced Accuracy 0.9970 0.9876 0.9822 0.9774 0.9876

print(table(prediction, valuation$classe))

##   
## prediction A B C D E  
## A 1670 13 2 0 0  
## B 0 1116 13 0 9  
## C 2 9 1002 38 10  
## D 1 0 6 923 6  
## E 1 1 3 3 1057

print(1 - (sum(diag(table(prediction, valuation$classe)))/ length(prediction)))

## [1] 0.01988105

## Apply Test data and Predict

FinalTest <- predict(preProc, myTest[, numericreturnofremovedvalues(myTest, c("problem\_id"))])  
myanswers <- predict(modfit, FinalTest)  
myanswers

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

## Conclusion

Since the out of error value is 0.0198811 which sugggests that the accuracy rate is close to 99%. I feel very confident using the randomforest ML process.

## Function to generate the files provided by Coursera

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)  
 }  
}

## Generating Files

pml\_write\_files(myanswers)