Practical Machine Learning Prediction Project

Removed For Privacy GitHub

Contents


```
## Run time: 2015-08-23 18:38:29
## R version: R version 3.1.2 (2014-10-31)
```

Prepare the datasets

```r
library(rpart)
library(rpart.plot)
```

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

```r
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.1.3
```

```r
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.
```

```r
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 3.1.3
```

```r
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 the training and testing data into a data table.

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

We are given an overwhelmingly large training data. We simply partition and generate a testing set rather than use only 20 observation in the given test set. Partition into training and testing is done with a 60/40 split in the code below.

```r
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
```

Remove zero variance predictors

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

myNZVvars <- names(myTraining) %in% row.names(myDataNZV[myDataNZV$nzv==TRUE,])
myTraining <- myTraining[!myNZVvars]
#To check the new N?? of observations
dim(myTraining)
```

```
## [1] 11776   128
```

We remove the first column of the data and also clean Variables with too many NAs. For Variables that have more than a 60% threshold of NA's we leave them out.

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

Do the same for the testing set of variables (as well as the original testing set.)

```
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
```

```
dim(testing)
```

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

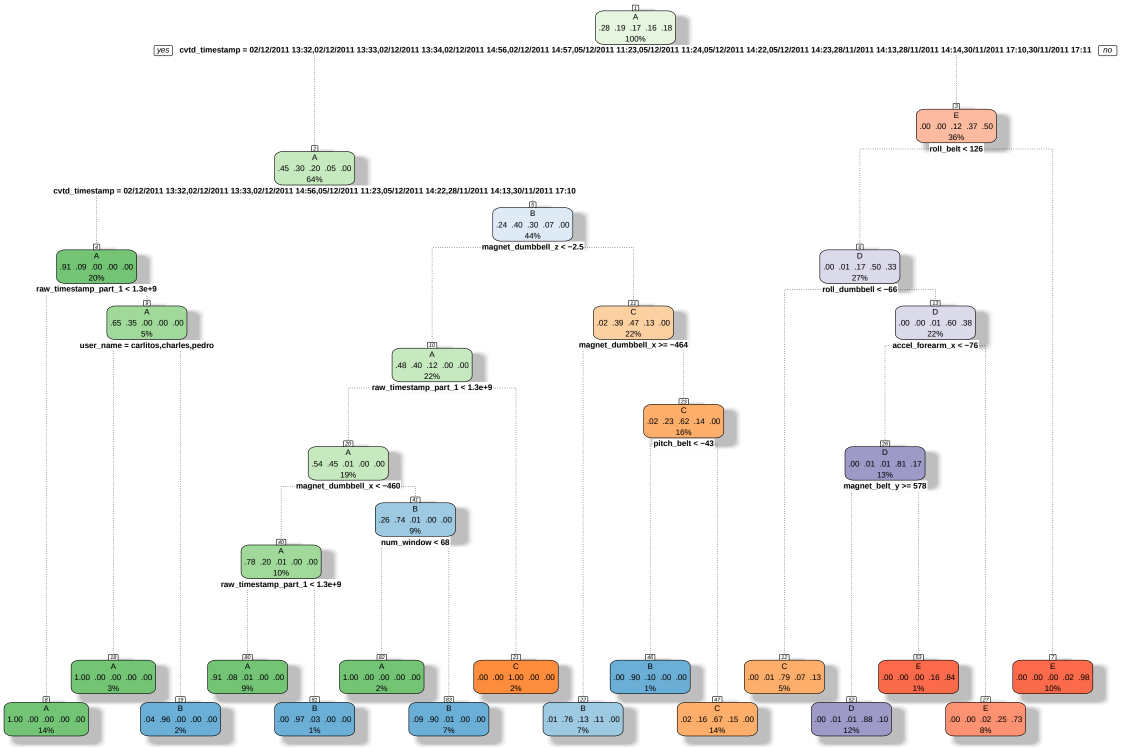
Harmonize the type of test and training data. Otherwise RandomForest throws an error that “Predictors in the new data do not match that of the training data”

```
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,]
```

Train a prediction model

We first use decision trees.

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



Rattle 2015−Aug−23 18:39:04 siva

We check how good the prediciton is

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

We look at the cross-tabulation of observed and predicted calsses with associated statistics.

```
confusionMatrix(predictionsA1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2143   64    7    3    0
##          B   69 1269   77   70    0
##          C   20  174 1263  198   63
##          D    0   11   12  823   80
##          E    0    0    9  192 1299
##
## Overall Statistics
##
##                Accuracy : 0.8663
##                  95% CI : (0.8586, 0.8738)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8308
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9601   0.8360   0.9232   0.6400   0.9008
## Specificity            0.9868   0.9659   0.9298   0.9843   0.9686
## Pos Pred Value         0.9666   0.8545   0.7352   0.8888   0.8660
## Neg Pred Value         0.9842   0.9609   0.9829   0.9331   0.9775
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2731   0.1617   0.1610   0.1049   0.1656
## Detection Prevalence   0.2826   0.1893   0.2190   0.1180   0.1912
## Balanced Accuracy      0.9735   0.9009   0.9265   0.8121   0.9347
```

Since accuracy is poor we try to use RandomForests

```r
modFitB1 <- randomForest(classe ~. , data=myTraining)
predictionsB1 <- predict(modFitB1, myTesting, type = "class")
confusionMatrix(predictionsB1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    0    0    0
##          C    0    0 1367    5    0
##          D    0    0    1 1281    2
##          E    0    0    0    0 1440
##
## Overall Statistics
##
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9993   0.9961   0.9986
## Specificity            0.9998   1.0000   0.9992   0.9995   1.0000
## Pos Pred Value         0.9996   1.0000   0.9964   0.9977   1.0000
## Neg Pred Value         1.0000   0.9998   0.9998   0.9992   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1742   0.1633   0.1835
## Detection Prevalence   0.2846   0.1933   0.1749   0.1637   0.1835
## Balanced Accuracy      0.9999   0.9997   0.9992   0.9978   0.9993
```

We obtain much better accuracy and pick this model.

Generate the files for submission.

We use the randome forest model to generate the required files.

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

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)
```