Practical Machine Learning Prediction Project

Removed For Privacy GitHub

Contents


```
## Run time: 2015-08-23 19:16:22
## R version: R version 3.1.2 (2014-10-31)
```

Prepare the datasets

```
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 3.1.3
```

```
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(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
```

```
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.

```
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.

```
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

```
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   131
```

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.

```
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
```

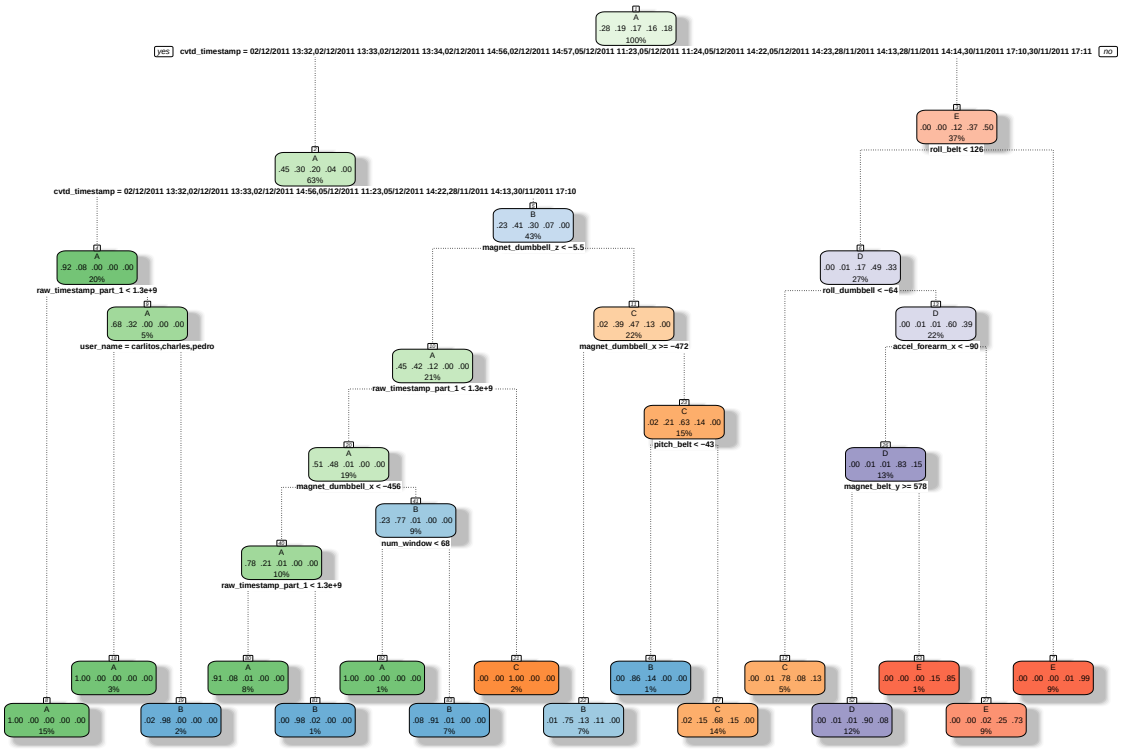
Harmonize the type of test and training data. Otherwise RandomForest throws an error that "Predictors in the new data do not match that of the training data"

```
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)
```



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 2139   83   10    1    0
##          B   66 1238   80   64    0
##          C   27  190 1251  209   57
##          D    0    7   20  776   73
##          E    0    0    7  236 1312
##
## Overall Statistics
##
##                Accuracy : 0.856
##                  95% CI : (0.848, 0.8637)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8177
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9583   0.8155   0.9145   0.6034   0.9098
## Specificity            0.9833   0.9668   0.9254   0.9848   0.9621
## Pos Pred Value         0.9579   0.8550   0.7215   0.8858   0.8437
## Neg Pred Value         0.9834   0.9562   0.9809   0.9268   0.9793
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2726   0.1578   0.1594   0.0989   0.1672
## Detection Prevalence   0.2846   0.1846   0.2210   0.1116   0.1982
## Balanced Accuracy      0.9708   0.8912   0.9200   0.7941   0.9360
```

Since accuracy is poor we try to use RandomForests

```
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    5    0    0
##          C    0    0 1361    7    0
##          D    0    0    2 1279    3
##          E    0    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.9977
##                  95% CI : (0.9964, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9971
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9949   0.9946   0.9979
## Specificity            0.9998   0.9992   0.9989   0.9992   1.0000
## Pos Pred Value         0.9996   0.9967   0.9949   0.9961   1.0000
## Neg Pred Value         1.0000   0.9998   0.9989   0.9989   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1735   0.1630   0.1834
## Detection Prevalence   0.2846   0.1940   0.1744   0.1637   0.1834
## Balanced Accuracy      0.9999   0.9993   0.9969   0.9969   0.9990
```

We obtain much better accuracy and pick this model. The estimated out of sample error is <0.3%

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)
```