Practical Machine Learning Course Project

César Fernández

7 de febrero de 2020

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Versión 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
library(gbm)
```

```
## Loaded gbm 2.1.5
```

Project Introduction

Background

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. 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

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Getting and loading the data

```r
set.seed(12345)
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!",""))
```

Partioning the training set into two

```r
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]
myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Cleaning the data

Remove NearZeroVariance variables

```r
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
myTraining <- myTraining[,nzv$nzv==FALSE]
nzv<- nearZeroVar(myTesting,saveMetrics=TRUE)
myTesting <- myTesting[,nzv$nzv==FALSE]
```

Remove the first column of the myTraining data set

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

Clean variables with more than 60% NA

```
trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
    if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .7) {
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) == 1)  {
                trainingV3 <- trainingV3[ , -j]
            }
        }
    }
}
# Set back to the original variable name
myTraining <- trainingV3
rm(trainingV3)
```

Transform the myTesting and testing data sets

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])  # remove the classe column
myTesting <- myTesting[clean1]         # allow only variables in myTesting that are also in myTraining
testing <- testing[clean2]             # allow only variables in testing that are also in myTraining
dim(myTesting)
```

```
## [1] 7846   58
```

```
dim(testing)
```

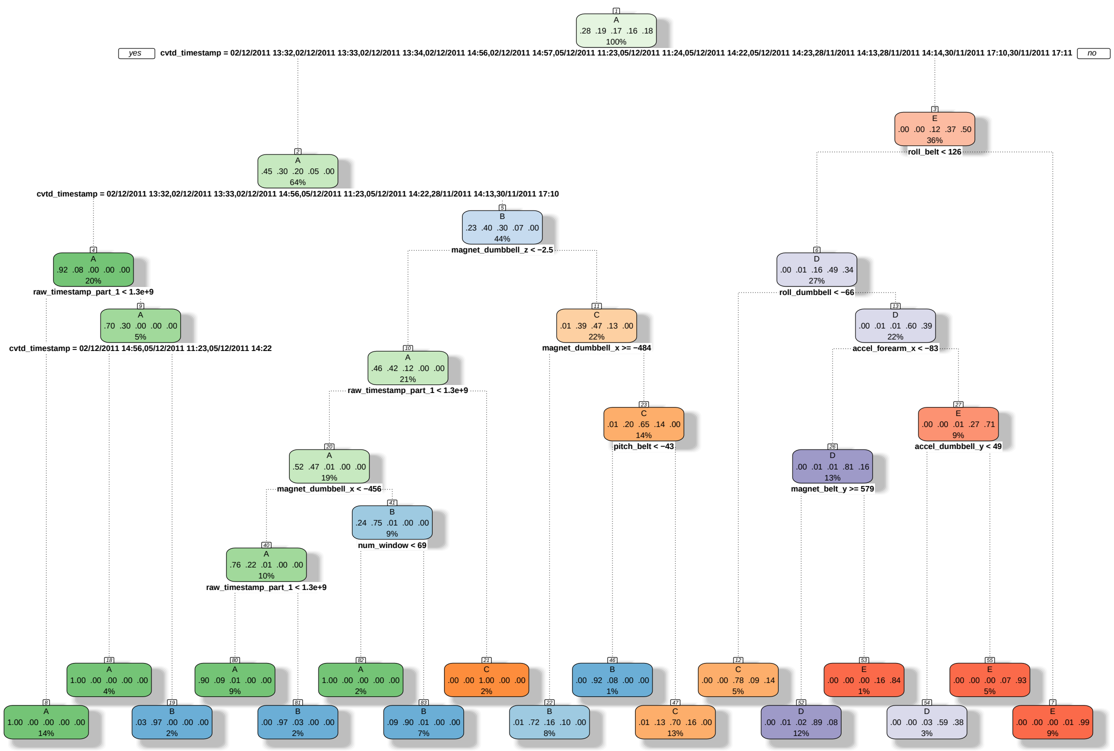
```
## [1] 20 57
```

Coerce the data into the same type

```
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])
        }
    }
}
# To get the same class between testing and myTraining
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

Prediction with Decision Trees

```
set.seed(12345)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```



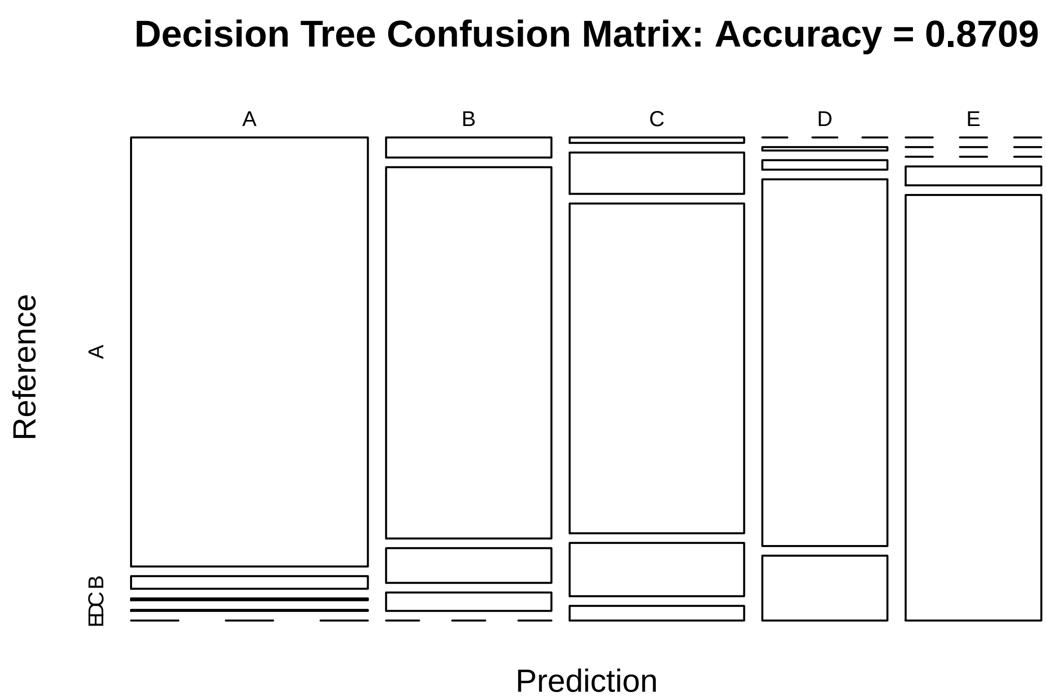
Rattle 2020−Feb.−07 18:09:10 Usuario

```r
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTesting$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2142   63    9    5    0
##          B   70 1294  121   64    0
##          C   20  152 1213  196   54
##          D    0    9   25  967  171
##          E    0    0    0   54 1217
##
## Overall Statistics
##
##                Accuracy : 0.8709
##                  95% CI : (0.8633, 0.8782)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8367
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9597   0.8524   0.8867   0.7519   0.8440
## Specificity            0.9863   0.9597   0.9349   0.9688   0.9916
## Pos Pred Value         0.9653   0.8354   0.7419   0.8251   0.9575
## Neg Pred Value         0.9840   0.9644   0.9750   0.9522   0.9658
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2730   0.1649   0.1546   0.1232   0.1551
## Detection Prevalence   0.2828   0.1974   0.2084   0.1494   0.1620
## Balanced Accuracy      0.9730   0.9061   0.9108   0.8603   0.9178
```

```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy =", rou
```



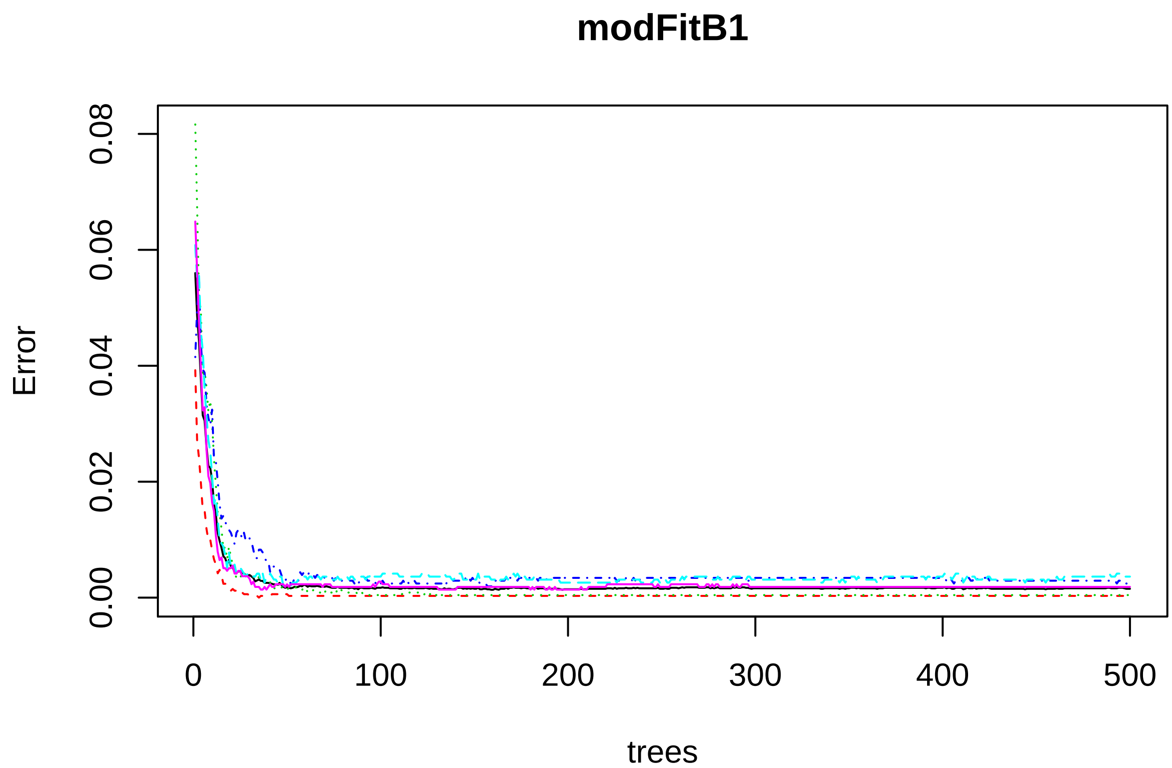
Prediction with Random Forests

```
set.seed(12345)
modFitB1 <- randomForest(classe ~ ., data=myTraining)
predictionB1 <- predict(modFitB1, myTesting, type = "class")
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf
```

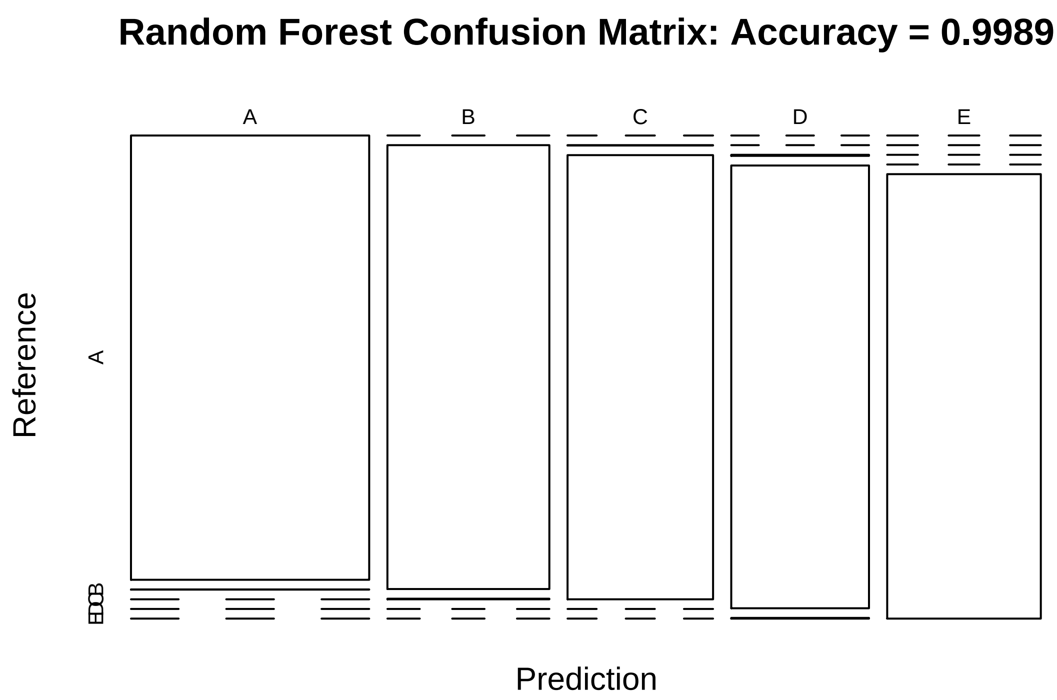
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1516    2    0    0
```

```
##          C    0    1 1363    0    0
##          D    0    0    3 1286    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   0.9987   0.9963   1.0000   0.9986
## Specificity            0.9998   0.9997   0.9998   0.9992   1.0000
## Pos Pred Value         0.9996   0.9987   0.9993   0.9961   1.0000
## Neg Pred Value         1.0000   0.9997   0.9992   1.0000   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1737   0.1639   0.1835
## Detection Prevalence   0.2846   0.1935   0.1738   0.1645   0.1835
## Balanced Accuracy      0.9999   0.9992   0.9981   0.9996   0.9993
```

```r
plot(modFitB1)
```




```
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", round
```



Prediction with Generalized Boosted

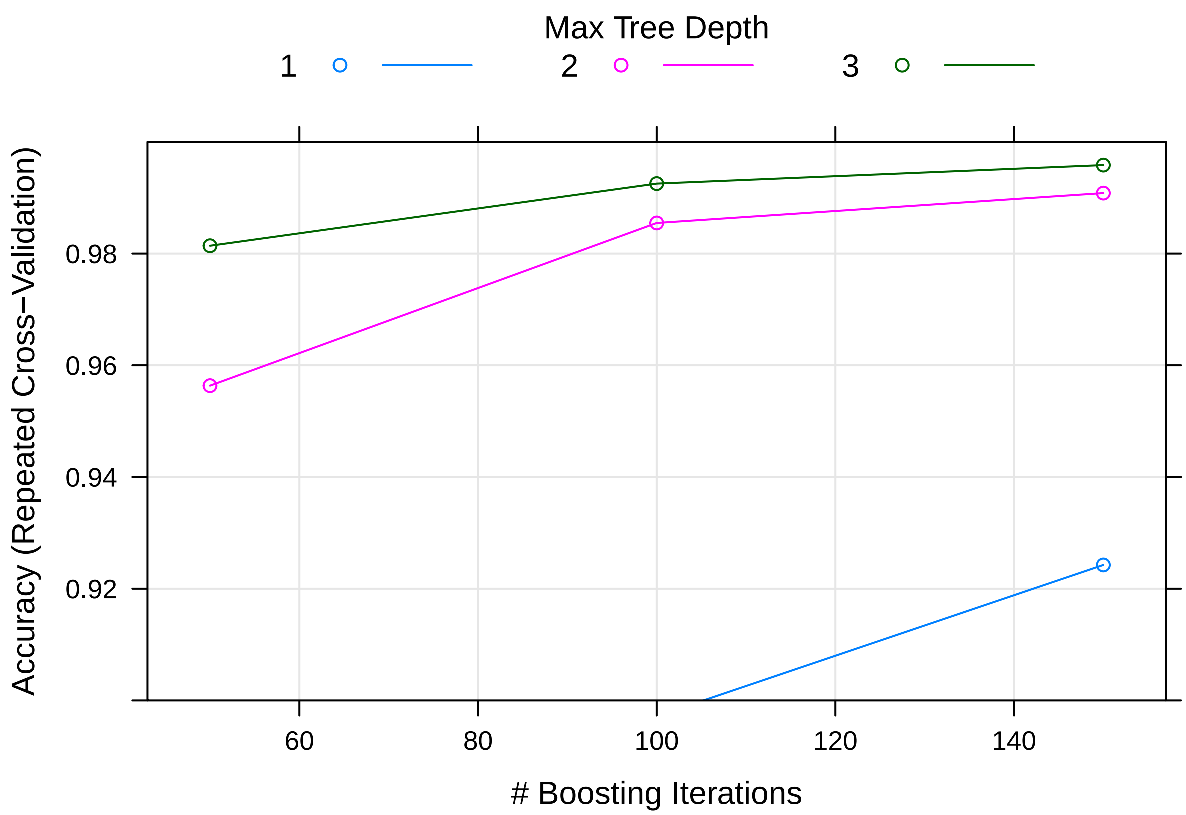
Regression

```
set.seed(12345)
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)
gbmFit1 <- train(classe ~ ., data=myTraining, method = "gbm",
                  trControl = fitControl,
                  verbose = FALSE)
gbmFinMod1 <- gbmFit1$finalModel
gbmPredTest <- predict(gbmFit1, newdata=myTesting)
gbmAccuracyTest <- confusionMatrix(gbmPredTest, myTesting$classe)
gbmAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1512    1    0    0
##          C    0    4 1360    0    0
##          D    0    1    7 1283    7
```

```
##          E    0    0    0    3 1435
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9955, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9960   0.9942   0.9977   0.9951
## Specificity            0.9998   0.9998   0.9994   0.9977   0.9995
## Pos Pred Value         0.9996   0.9993   0.9971   0.9884   0.9979
## Neg Pred Value         1.0000   0.9991   0.9988   0.9995   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1927   0.1733   0.1635   0.1829
## Detection Prevalence   0.2846   0.1928   0.1738   0.1654   0.1833
## Balanced Accuracy      0.9999   0.9979   0.9968   0.9977   0.9973
```

```
plot(gbmFit1, ylim = c(0.9,1))
```



Predicting Results on the Test Data

Random Forests gave an Accuracy in the myTesting dataset of 99.89%, which was more accurate that what I got from the Decision Trees or GBM. The expected out-of-sample error is 100-99.89 = 0.11%.

```
predictionB2 <- predict(modFitB1, testing, type = "class")
predictionB2
```

```
##  1 21  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```