# Practical Machine Learning - Course Project

*Faruk Miguel*

*August 15, 2017*

#### Indications

## 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.

## Submition

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.

#### Solution

## Getting and loading the data

```
set.seed (54321)

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!",""))
```

## Dividing the training set in two subsets

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
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

## Removing the variable with variance near 0, the first column of the myTraining data set

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

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

## Cleaning variables with more than 60% of missing values

```
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]
            }
        }
    }
}

myTraining <- trainingV3
rm(trainingV3)
```

## Transforming myTesting and testing data set

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])  # removing classe column
myTesting <- myTesting[clean1]
testing <- testing[clean2]

dim(myTesting)
```

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

```
dim(testing)
```

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

## Coercing the data into the same type

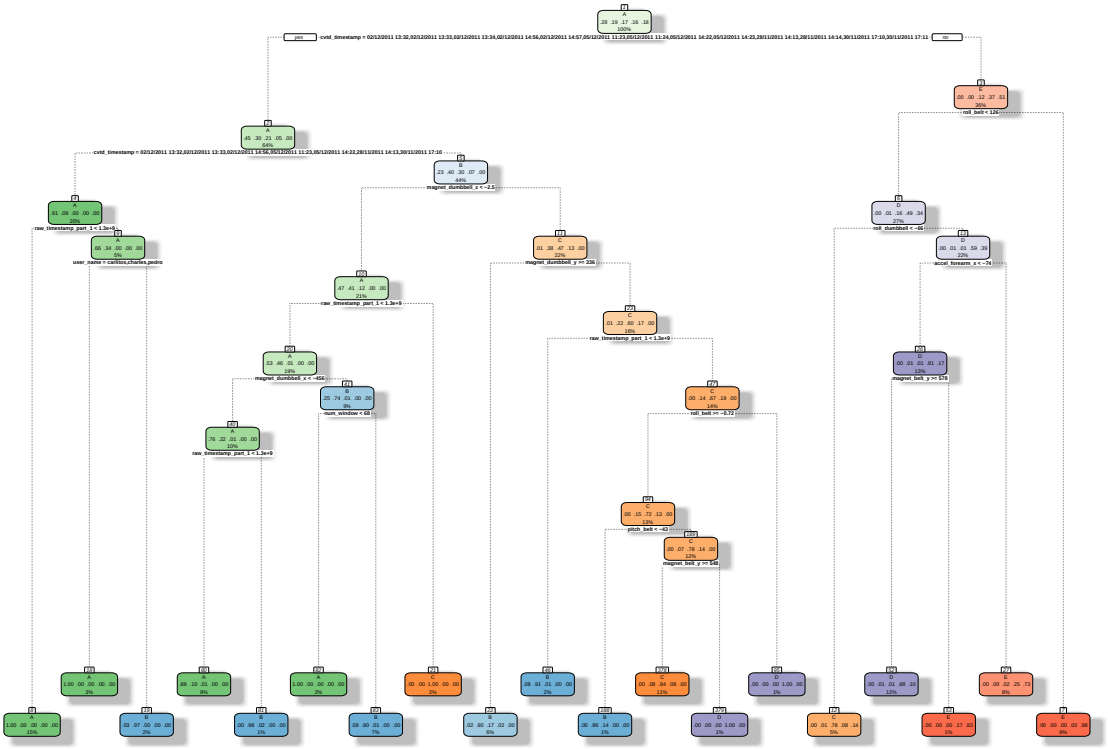
```
for (i in 1:length(testing) ) {
    for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}

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

### Prediction with Decision Trees

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
library(rpart.plot)
set.seed(54321)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```



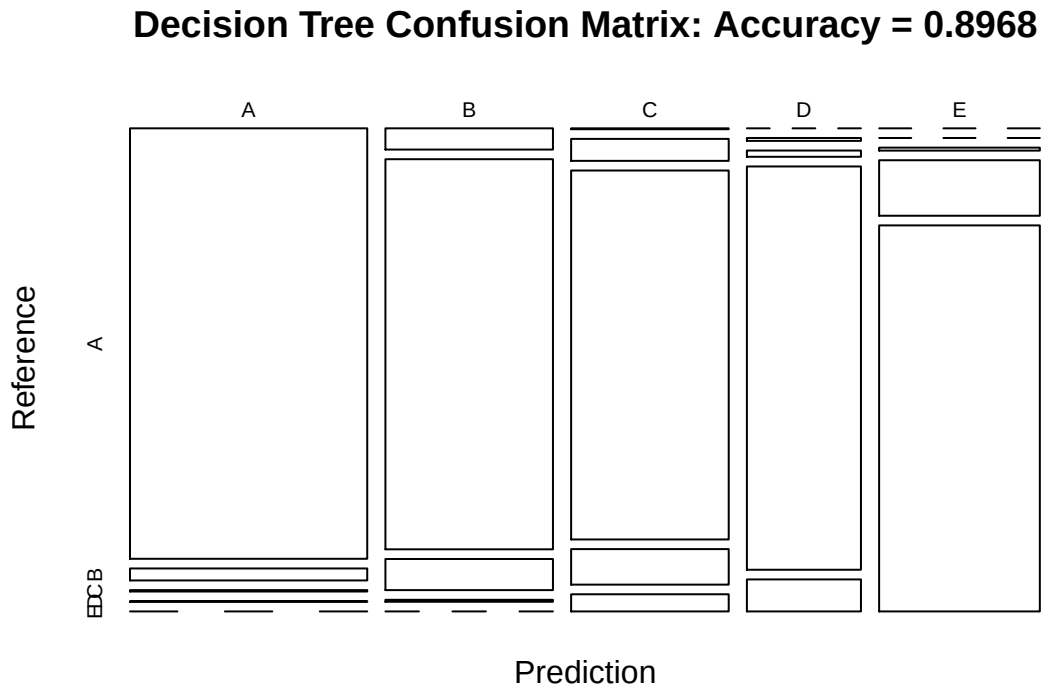
Rattle 2017−Aug−15 22:56:16 Faruk Miguel

```
library(caret)
library(e1071)
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTesting$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2154   60    7    3    0
##          B   75 1378  110    7    0
##          C    3   73 1226  118   57
##          D    0    7   15  970   77
##          E    0    0   10  188 1308
##
## Overall Statistics
##
##                Accuracy : 0.8968
##                  95% CI : (0.8898, 0.9034)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8694
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9651   0.9078   0.8962   0.7543   0.9071
## Specificity            0.9875   0.9697   0.9613   0.9849   0.9691
## Pos Pred Value         0.9685   0.8777   0.8301   0.9074   0.8685
## Neg Pred Value         0.9861   0.9777   0.9777   0.9534   0.9789
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2745   0.1756   0.1563   0.1236   0.1667
## Detection Prevalence   0.2835   0.2001   0.1882   0.1362   0.1919
## Balanced Accuracy      0.9763   0.9387   0.9287   0.8696   0.9381
```

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




### Prediction with Random Forests

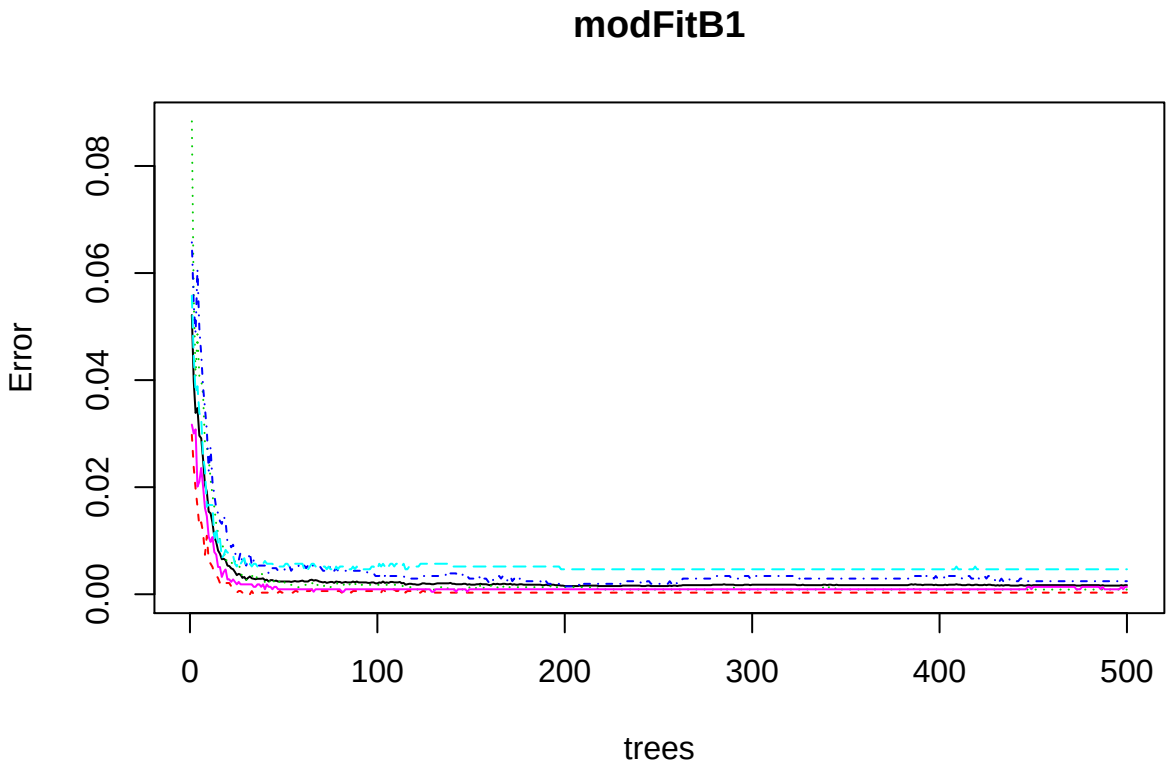
```
library(randomForest)
```

```
## 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
```

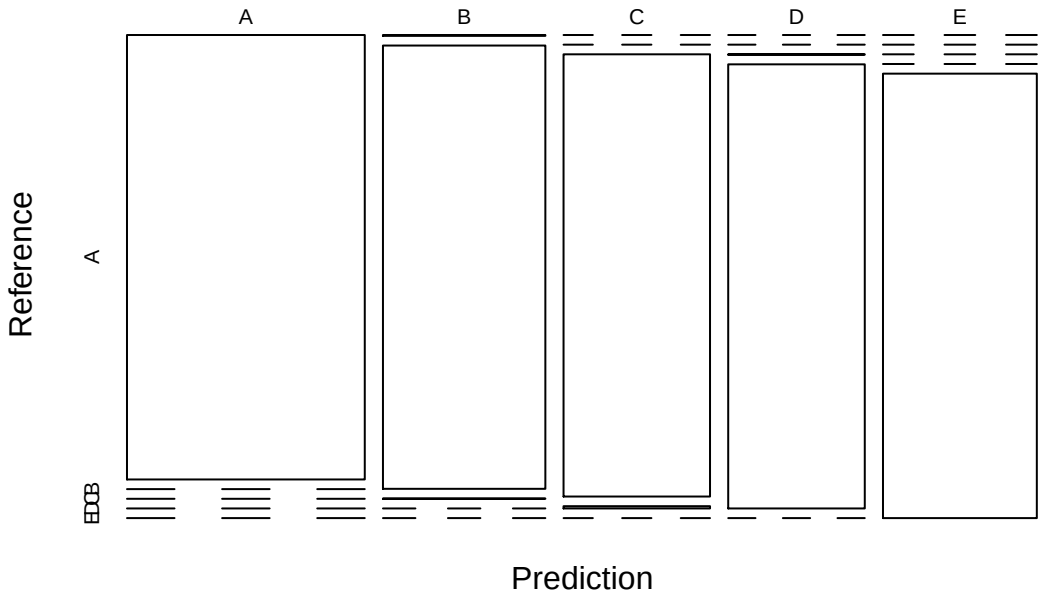
```r
set.seed(54321)
modFitB1 <- randomForest(classe ~ ., data=myTraining)
predictionB1 <- predict(modFitB1, myTesting, type = "class")
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    0    0    0    0
##          B    3 1518    1    0    0
##          C    0    0 1366    7    0
##          D    0    0    1 1279    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9985
##                  95% CI : (0.9973, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9981
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   1.0000   0.9985   0.9946   1.0000
## Specificity            1.0000   0.9994   0.9989   0.9998   1.0000
## Pos Pred Value         1.0000   0.9974   0.9949   0.9992   1.0000
## Neg Pred Value         0.9995   1.0000   0.9997   0.9989   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1935   0.1741   0.1630   0.1838
## Detection Prevalence   0.2841   0.1940   0.1750   0.1631   0.1838
## Balanced Accuracy      0.9993   0.9997   0.9987   0.9972   1.0000
```

```r
plot(modFitB1)
```




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




### Prediction with Generalized Boosted Regresion

```
set.seed(54321)
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

gbmFit1 <- train(classe ~ ., data=myTraining, method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)
```

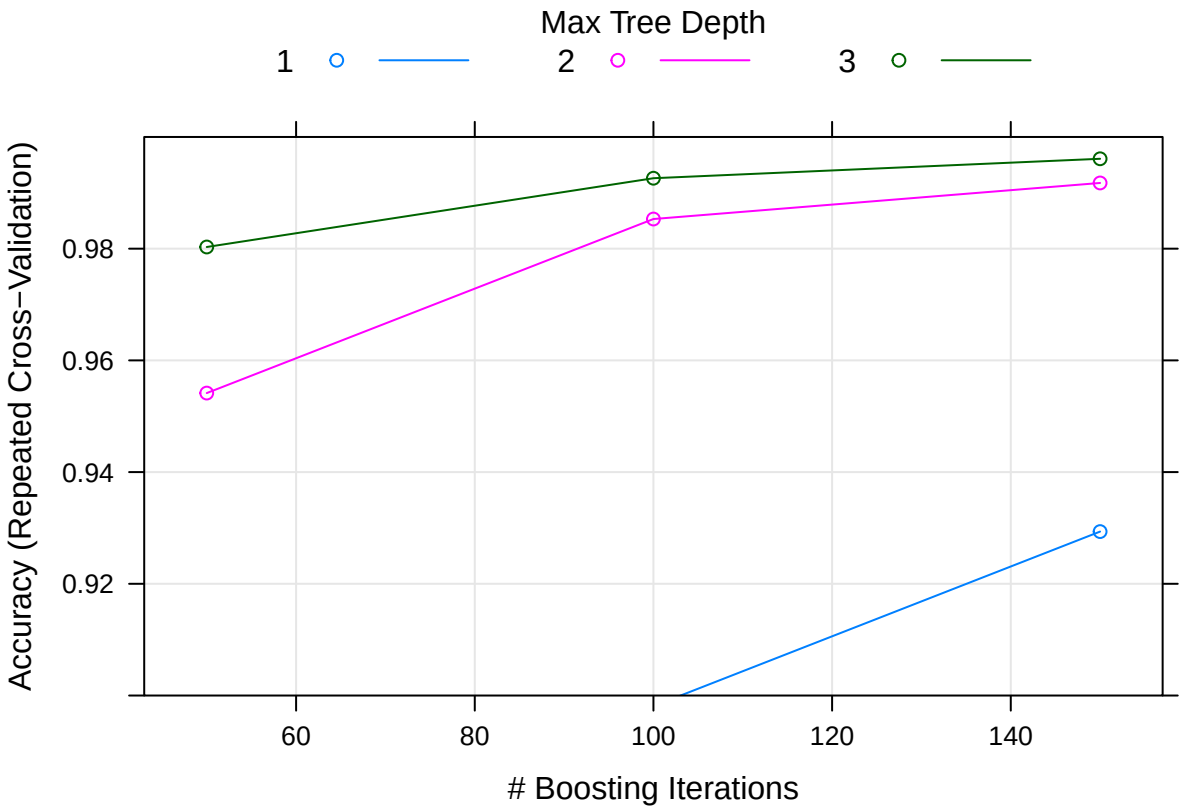
```
## Loading required package: gbm
## Loading required package: survival
##
## Attaching package: 'survival'
## The following object is masked from 'package:caret':
##
##     cluster
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.3
```

```
## Loading required package: plyr
```

```
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    2    0    0    0
##          B    0 1514    1    0    0
##          C    0    1 1359    3    0
##          D    0    1    8 1283    2
##          E    0    0    0    0 1440
##
## 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   0.9974   0.9934   0.9977   0.9986
## Specificity            0.9996   0.9998   0.9994   0.9983   1.0000
## Pos Pred Value         0.9991   0.9993   0.9971   0.9915   1.0000
## Neg Pred Value         1.0000   0.9994   0.9986   0.9995   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1930   0.1732   0.1635   0.1835
## Detection Prevalence   0.2847   0.1931   0.1737   0.1649   0.1835
## Balanced Accuracy      0.9998   0.9986   0.9964   0.9980   0.9993
```

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




### Prediction Results on the Test Data

The accuracy in the myTesting dataset is 99.89%, which was more accurate from the Decision Trees or GBM. The expected out-of-sample error is 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
```

```
# Results in a text file for submission
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(predictionB2)
```