Prediction Assignment Writeup

Matias Caggiani

May 22, 2016

Introduction

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, our 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.

Reproducibility

Load required libraries

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.4
```

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

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

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

```r
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
library(knitr)
library(caTools)
```

Load data

```r
setwd("~/Dropbox/Coursera/JohnHopkins/8_MachineLearning/courseProject")

training = read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))

testing = read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
```

Split the train data

```r
set.seed(1984)

spl = sample.split(training$classe, SplitRatio = 0.6)

train = subset(training, spl==T)

test = subset(training, spl==F)
```

Clean the data

```r
nearZerVar = nearZeroVar(train, saveMetrics=T)

train = train[,nearZerVar$nzv==F]

nearZerVar = nearZeroVar(test, saveMetrics=T)

test = test[,nearZerVar$nzv==F]
```

Remove the first column of the train data set since we don't need it

```r
train = train[c(-1)]
```

Clean variables with more than 70% NA to remove noise

```r
trainAux = train
for(i in 1:length(train)) {
    if( sum( is.na( train[, i] ) ) /nrow(train) >= .7) {
        for(j in 1:length(trainAux)) {
            if( length( grep(names(train[i]), names(trainAux)[j]) ) == 1)  {
                trainAux = trainAux[ , -j]
            }
        }
    }
}

train = trainAux

rm(trainAux)
```

Transform the myTesting and testing data sets

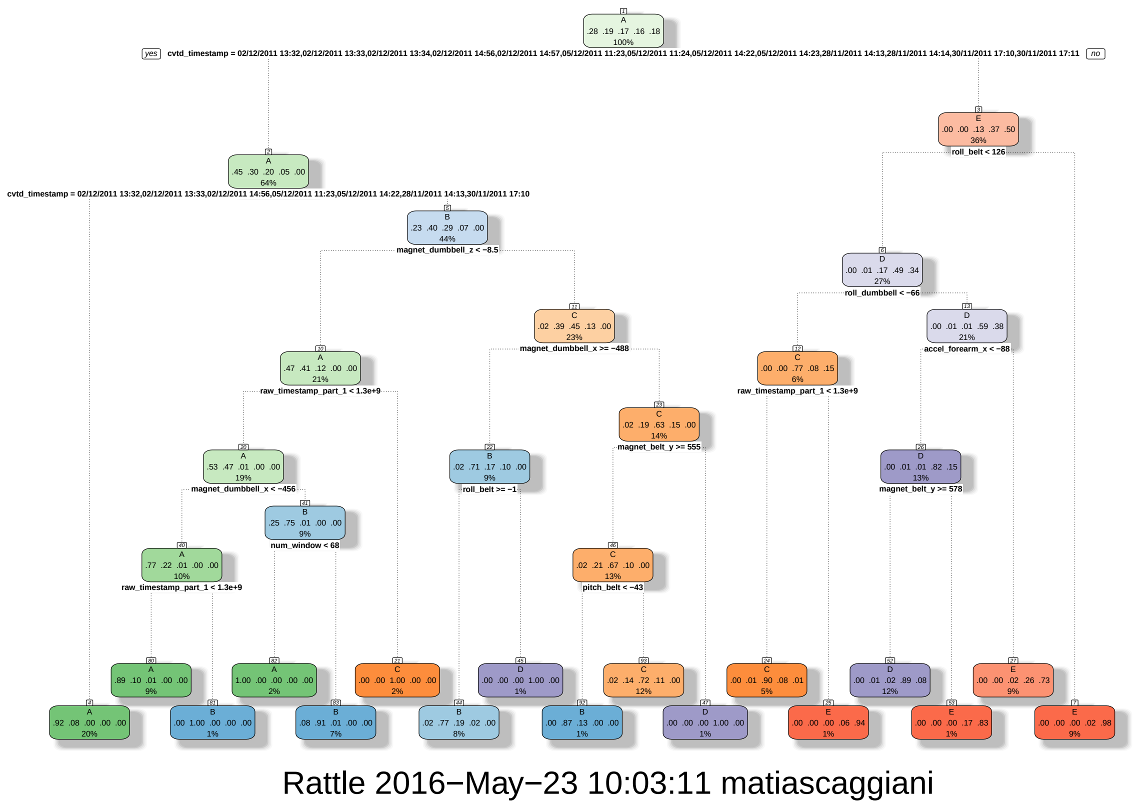
```r
cleanAux1 = colnames(train)

cleanAux2 = colnames(train[, -58])

test = test[cleanAux1]
testing = testing[cleanAux2]
```

Coerce the test and train data into the same type

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

testing = rbind(train[2, -58] , testing)

testing = testing[-1,]
```

Prediction with Decision Trees

```
set.seed(1984)

fitDecisionTree = rpart(classe ~ ., data=train, method="class")

fancyRpartPlot(fitDecisionTree)
```



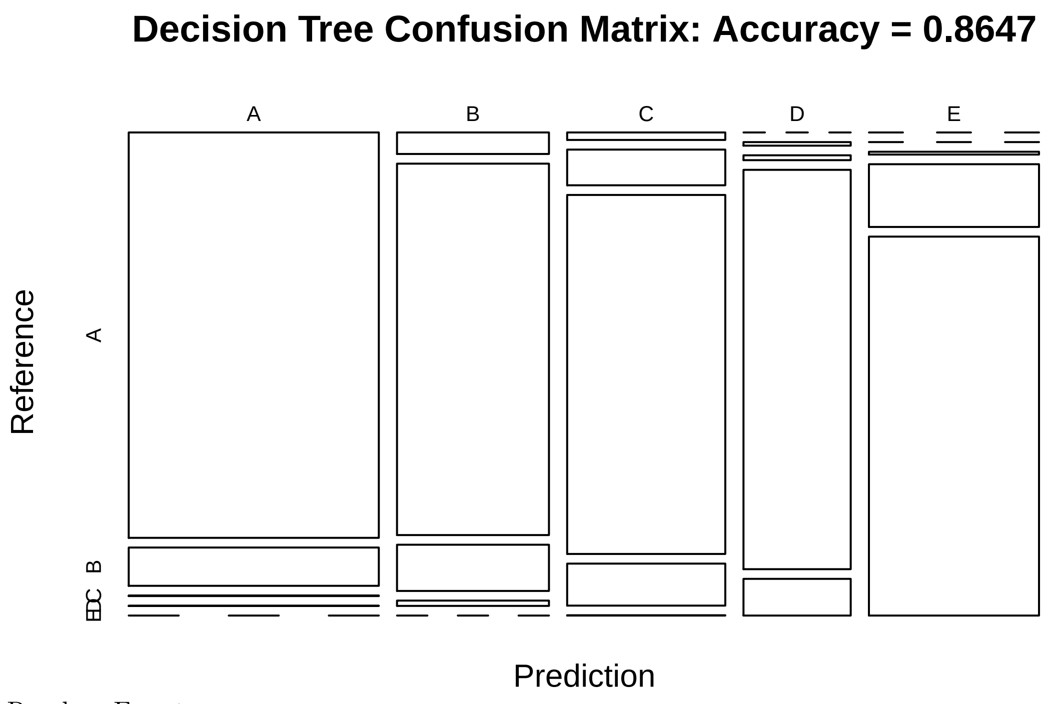
A .28 .19 .17 .16 .18 100%
yes
cvtd_timestamp = 02/12/2011 13:32,02/12/2011 13:33,02/12/2011 13:34,02/12/2011 14:56,02/12/2011 14:57,05/12/2011 11:23,05/12/2011 11:24,05/12/2011 14:22,05/12/2011 14:23,28/11/2011 14:13,28/11/2011 14:14,30/11/2011 17:10,30/11/2011 17:11
no
A .45 .30 .20 .05 .00 64%
cvtd_timestamp = 02/12/2011 13:32,02/12/2011 13:33,02/12/2011 14:56,05/12/2011 11:23,05/12/2011 14:22,28/11/2011 14:13,30/11/2011 17:10
E .00 .00 .13 .37 .50 36%
roll_belt < 126
B .23 .40 .29 .07 .00 44%
magnet_dumbbell_z < −8.5
D .00 .01 .17 .49 .34 27%
roll_dumbbell < −66
C .02 .39 .45 .13 .00 23%
magnet_dumbbell_x >= −488
D .00 .01 .01 .59 .38 21%
accel_forearm_x < −88
A .47 .41 .12 .00 .00 21%
raw_timestamp_part_1 < 1.3e+9
C .00 .00 .77 .08 .15 6%
raw_timestamp_part_1 < 1.3e+9
C .02 .19 .63 .15 .00 14%
magnet_belt_y >= 555
A .53 .47 .01 .00 .00 19%
magnet_dumbbell_x < −456
B .02 .71 .17 .10 .00 9%
roll_belt >= −1
D .00 .01 .01 .82 .15 13%
magnet_belt_y >= 578
B .25 .75 .01 .00 .00 9%
num_window < 68
A .77 .22 .01 .00 .00 10%
raw_timestamp_part_1 < 1.3e+9
C .02 .21 .67 .10 .00 13%
pitch_belt < −43
A .89 .10 .01 .00 .00 9%
A 1.00 .00 .00 .00 .00 2%
C .00 .00 1.00 .00 .00 2%
D .00 .00 .00 1.00 .00 1%
C .02 .14 .72 .11 .00 12%
C .00 .01 .90 .08 .01 5%
D .00 .01 .02 .89 .08 12%
E .00 .00 .02 .26 .73 9%
A .92 .08 .00 .00 .00 20%
B .00 1.00 .00 .00 .00 1%
B .08 .91 .01 .00 .00 7%
B .02 .77 .19 .02 .00 8%
B .00 .87 .13 .00 .00 1%
D .00 .00 .00 1.00 .00 1%
E .00 .00 .00 .06 .94 1%
E .00 .00 .00 .17 .83 1%
E .00 .00 .00 .02 .98 9%
Rattle 2016−May−23 10:03:11 matiasscaggiani

```
DecisionTreePred = predict(fitDecisionTree, test, type = "class")

DecisionTreeConfMatrix = confusionMatrix(DecisionTreePred, test$classe)

DecisionTreeConfMatrix
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2138  202    3    1    0
##          B   69 1190  148   17    0
##          C   25  119 1197  140    1
##          D    0    8   11  903   83
##          E    0    0   10  225 1359
##
## Overall Statistics
##
##                Accuracy : 0.8647
##                  95% CI : (0.8569, 0.8722)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8285
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9579   0.7834   0.8744   0.7022   0.9418
## Specificity            0.9633   0.9630   0.9560   0.9845   0.9633
## Pos Pred Value         0.9121   0.8357   0.8077   0.8985   0.8526
## Neg Pred Value         0.9829   0.9488   0.9730   0.9440   0.9866
## Prevalence             0.2844   0.1935   0.1744   0.1638   0.1838
## Detection Rate         0.2724   0.1516   0.1525   0.1150   0.1731
## Detection Prevalence   0.2986   0.1814   0.1888   0.1280   0.2031
## Balanced Accuracy      0.9606   0.8732   0.9152   0.8433   0.9526
```

```
plot(DecisionTreeConfMatrix$table, col = DecisionTreeConfMatrix$byClass, main = paste("Decision Tree Co
```

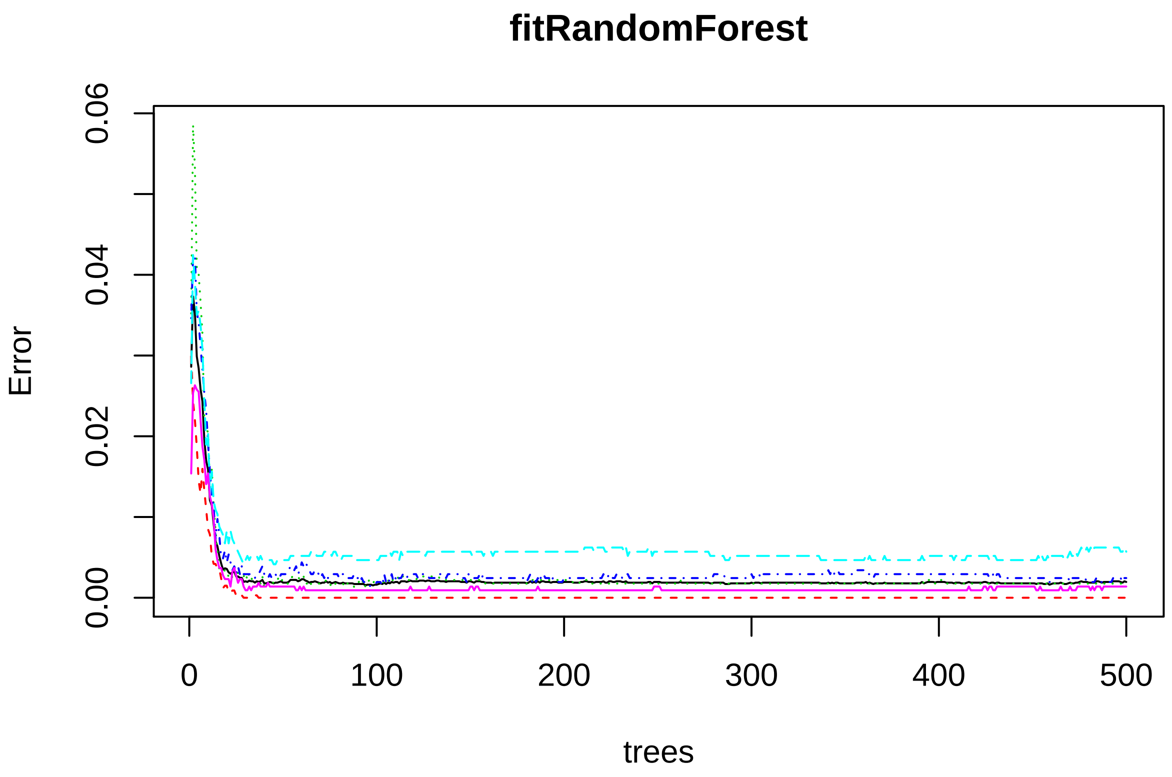


# Prediction with Random Forests

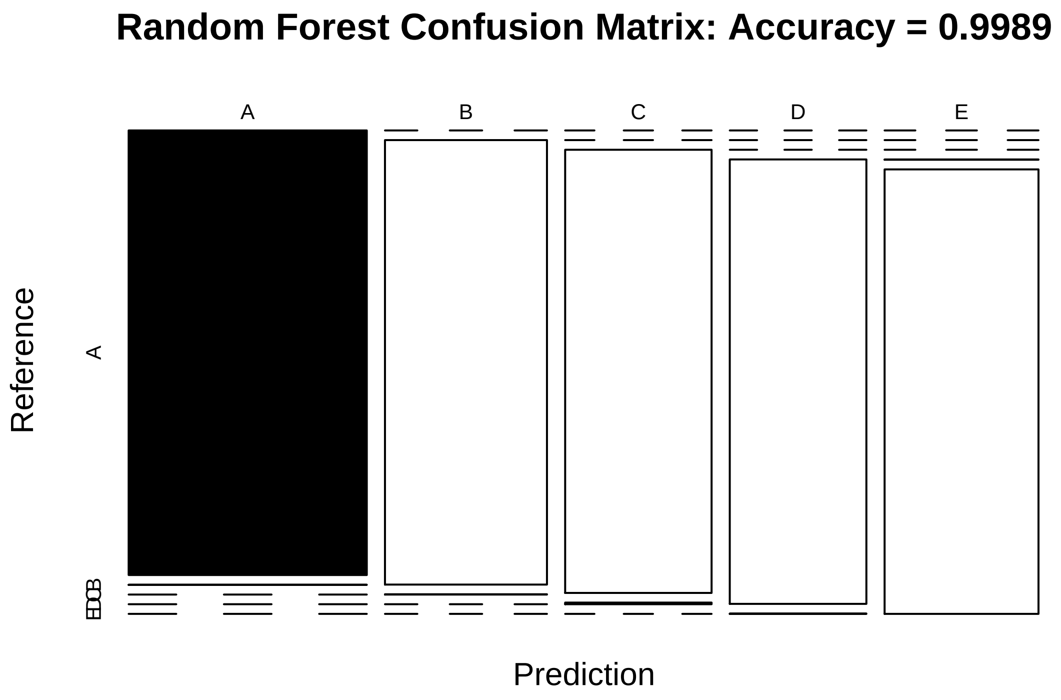
```r
set.seed(1984)

fitRandomForest = randomForest(classe ~ ., data=train)

RandomForestPred = predict(fitRandomForest, test, type = "class")

RandomForestConfMatrix = confusionMatrix(RandomForestPred, test$classe)

RandomForestConfMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1518    1    0    0
##          C    0    0 1368    5    0
##          D    0    0    0 1280    1
##          E    0    0    0    1 1442
##
## Overall Statistics
##
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2844
##     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   0.9953   0.9993
## Specificity            0.9998   0.9998   0.9992   0.9998   0.9998
## Pos Pred Value         0.9996   0.9993   0.9964   0.9992   0.9993
## Neg Pred Value         1.0000   0.9998   0.9998   0.9991   0.9998
## Prevalence             0.2844   0.1935   0.1744   0.1638   0.1838
## Detection Rate         0.2844   0.1934   0.1743   0.1631   0.1837
## Detection Prevalence   0.2845   0.1935   0.1749   0.1632   0.1838
## Balanced Accuracy      0.9999   0.9996   0.9992   0.9976   0.9996
```

```r
plot(fitRandomForest)
```



```
plot(RandomForestConfMatrix$table, col = RandomForestConfMatrix$byClass, main = paste("Random Forest Co
```



Prediction with Generalized Boosted Regression

```
set.seed(1984)

fitControl = trainControl(method = "repeatedcv",number = 5,repeats = 1)

fitGeneralizedBoostedRegression = train(classe ~ ., data=train, method = "gbm",trControl =fitControl,ve
```

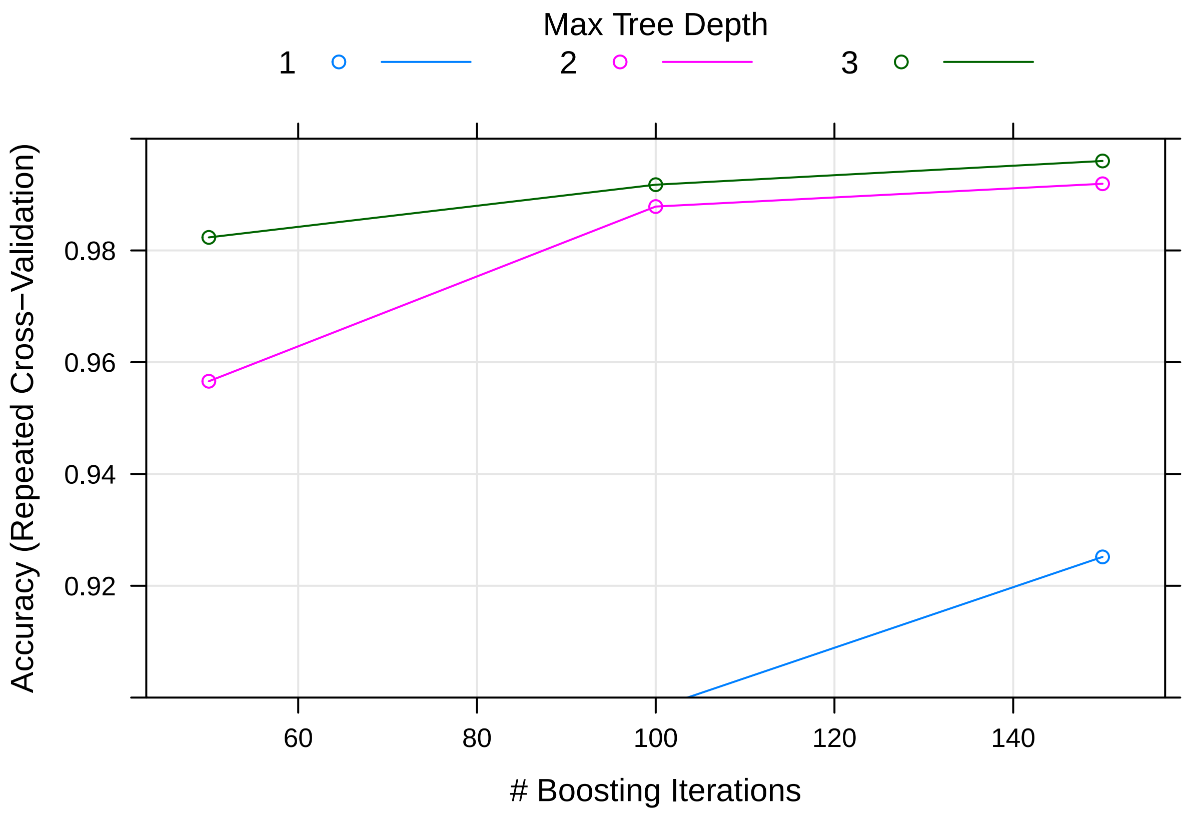
```
## Loading required package: gbm

## 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 2.1.1

## Loading required package: plyr
```

```
GeneralizedBoostedRegressionPred = predict(fitGeneralizedBoostedRegression, newdata=test,n.trees=fitGen

GeneralizedBoostedRegressionConfMatrix = confusionMatrix(GeneralizedBoostedRegressionPred, test$classe)

GeneralizedBoostedRegressionConfMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    4    0    0    0
##          B    0 1512    2    0    0
##          C    0    3 1362    3    0
##          D    0    0    5 1281    3
##          E    0    0    0    2 1440
##
## Overall Statistics
##
##                Accuracy : 0.9972
##                  95% CI : (0.9958, 0.9982)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9965
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9954   0.9949   0.9961   0.9979
## Specificity            0.9993   0.9997   0.9991   0.9988   0.9997
## Pos Pred Value         0.9982   0.9987   0.9956   0.9938   0.9986
## Neg Pred Value         1.0000   0.9989   0.9989   0.9992   0.9995
## Prevalence             0.2844   0.1935   0.1744   0.1638   0.1838
## Detection Rate         0.2844   0.1926   0.1735   0.1632   0.1835
## Detection Prevalence   0.2849   0.1929   0.1743   0.1642   0.1837
## Balanced Accuracy      0.9996   0.9975   0.9970   0.9974   0.9988
```

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



#

Conclusion Decision Tree = Accuracy : 0.8647 95% CI : (0.8569, 0.8722)

Random Forest = Accuracy : 0.9989 95% CI : (0.9978, 0.9995)

Generalized Boosted Regression = Accuracy : 0.9972 95% CI : (0.9958, 0.9982)

The confusion matrices show, that the Random Forest algorithm performens better than Decision Tree and Generalized Boosted Regression. Moreover, the accuracy for the Random Forest model was 0.9989 95% CI : (0.9978, 0.9995) compared to 0.9972 95% CI : (0.9958, 0.9982) for Generalized Boosted Regression and 0.8647 95% CI : (0.8569, 0.8722) for Decision Tree. The Random Forest model is choosen. ## Expected out-of-sample error The expected out-of-sample error is estimated at 0.0011, or 0.11%. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be missclassified. # Predicting Results on the Test Data

```
predictionOnTesting = predict(fitRandomForest, testing, type = "class")

predictionOnTesting
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

Submission

```
submission = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=F,row.names=F,col.names=F)
    }
}

submission(predictionOnTesting)
```