Practical Machine Learning Assignment

vesr

November 9, 2017

Prediction Assignment

Background

Smart wearable devices such as AppleWatch, Samsung, Fitbit etc is now allowing to collect a large amount of data about personal activities accurately. These devices are integral part of health and self motivaation - a study consisting of 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, the goal is 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: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). The raw training data has 19622 rows of observations and 158 features (predictors). Column `X` is unusable row number. While the testing data has 20 rows and the same 158 features. There is one column of target outcome named `classe`.

Preparing the data and R packages

Load packages, set caching

```r
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(knitr)
knitr::opts_chunk$set(cache=TRUE)
```

Preparing and downloading the data from the website. #### Getting Data

```r
# URL of the training and testing data
set.seed(12345)

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"



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

The raw training data has 19622 rows of observations and 158 features (predictors). Column `X` is unusable row number. While the testing data has 20 rows and the same 158 features. There is one column of target outcome named `classe`.

Data cleaning

First, extract target outcome (the activity quality) from training data, so now the training data contains only the predictors (the activity monitors).

```
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

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

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

Remove Column header from Training Data Set. Clean variables with more than 70% NA

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

trainingTemp <- myTraining
for(i in 1:length(myTraining)) {
  if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .7) {
    for(j in 1:length(trainingTemp)) {
      if( length( grep(names(myTraining[i]), names(trainingTemp)[j]) ) == 1)  {
        trainingTemp <- trainingTemp[ , -j]
      }
    }
  }
}

# Set back to the original variable name
myTraining <- trainingTemp
rm(trainingTemp)
```

Filter the myTesting and testing data sets and remove the 58 variable

```
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

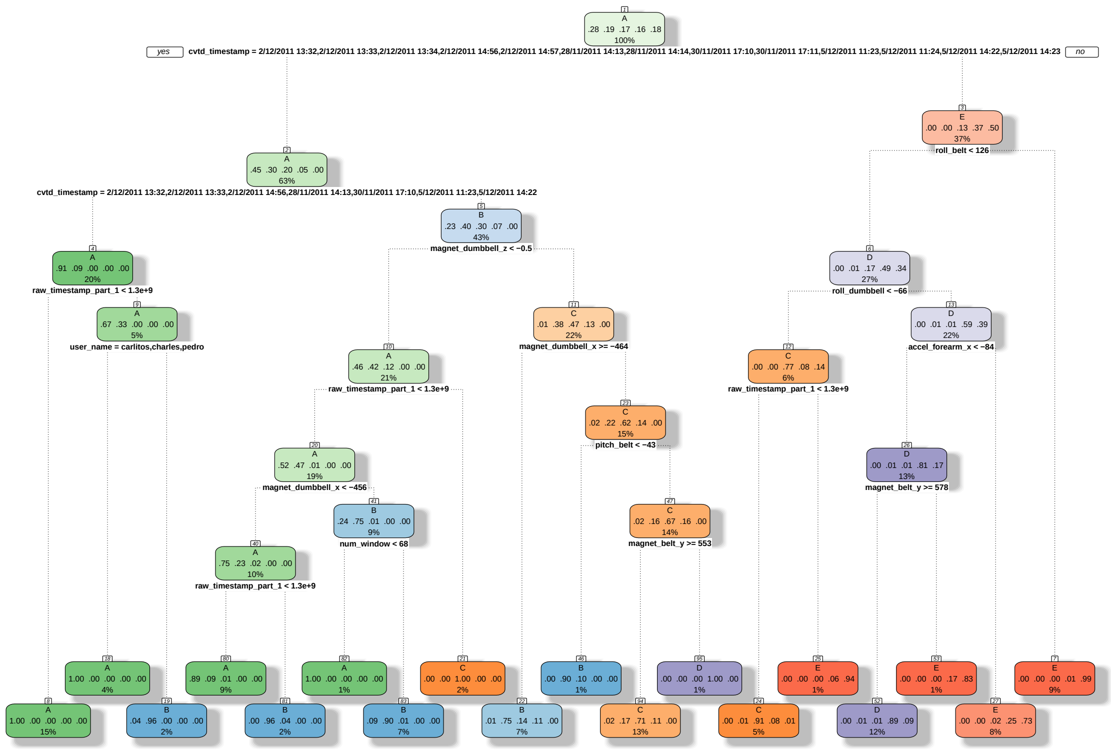
```
for (i in 1:length(testing) ) {
  for(j in 1:length(myTraining)) {
    if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
      class(testing[j]) <- class(myTraining[i])
    }
  }
}

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

Predicting with Fancy Plot.

FancyPlot are excellent tool for prediction due to the facts simple to understand (white box) from a tree we can extract interpretable results and make simple decisions they are helpful for exploratory analysis as binary structure of tree is simple to visualize very good prediction accuracy performance very fast they can be simply tuned by ensemble learning techniques

```
# convert character levels to numeric
set.seed(12345)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1, caption = "Prediction using FancyPlot")
```


1 A .28 .19 .17 .16 .18 100%
yes
cvtd_timestamp = 2/12/2011 13:32,2/12/2011 13:33,2/12/2011 13:34,2/12/2011 14:56,2/12/2011 14:57,28/11/2011 14:13,28/11/2011 14:14,30/11/2011 17:10,30/11/2011 17:11,5/12/2011 11:23,5/12/2011 11:24,5/12/2011 14:22,5/12/2011 14:23
no
2 A .45 .30 .20 .05 .00 63%
cvtd_timestamp = 2/12/2011 13:32,2/12/2011 13:33,2/12/2011 14:56,28/11/2011 14:13,30/11/2011 17:10,5/12/2011 11:23,5/12/2011 14:22
3 E .00 .00 .13 .37 .50 37%
roll_belt < 126
4 A .91 .09 .00 .00 .00 20%
raw_timestamp_part_1 < 1.3e+9
5 B .23 .40 .30 .07 .00 43%
magnet_dumbbell_z < −0.5
6 D .00 .01 .17 .49 .34 27%
roll_dumbbell < −66
9 A .67 .33 .00 .00 .00 5%
user_name = carlitos,charles,pedro
11 C .01 .38 .47 .13 .00 22%
magnet_dumbbell_x >= −464
13 D .00 .01 .01 .59 .39 22%
accel_forearm_x < −84
10 A .46 .42 .12 .00 .00 21%
raw_timestamp_part_1 < 1.3e+9
12 C .00 .00 .77 .08 .14 6%
raw_timestamp_part_1 < 1.3e+9
23 C .02 .22 .62 .14 .00 15%
pitch_belt < −43
20 A .52 .47 .01 .00 .00 19%
magnet_dumbbell_x < −456
26 D .00 .01 .01 .81 .17 13%
magnet_belt_y >= 578
41 B .24 .75 .01 .00 .00 9%
num_window < 68
47 C .02 .16 .67 .16 .00 14%
magnet_belt_y >= 553
40 A .75 .23 .02 .00 .00 10%
raw_timestamp_part_1 < 1.3e+9
18 A 1.00 .00 .00 .00 .00 4%
80 A .89 .09 .01 .00 .00 9%
82 A 1.00 .00 .00 .00 .00 1%
21 C .00 .00 1.00 .00 .00 2%
46 B .00 .90 .10 .00 .00 1%
95 D .00 .00 .00 1.00 .00 1%
25 E .00 .00 .00 .06 .94 1%
53 E .00 .00 .00 .17 .83 1%
7 E .00 .00 .00 .01 .99 9%
8 A 1.00 .00 .00 .00 .00 15%
19 B .04 .96 .00 .00 .00 2%
81 B .00 .96 .04 .00 .00 2%
83 B .09 .90 .01 .00 .00 7%
22 B .01 .75 .14 .11 .00 7%
94 C .02 .17 .71 .11 .00 13%
24 C .00 .01 .91 .08 .01 5%
52 D .00 .01 .01 .89 .09 12%
27 E .00 .00 .02 .25 .73 8%

Prediction using FancyPlot

Create Confusionmatrix. The outcome is removed from training data.

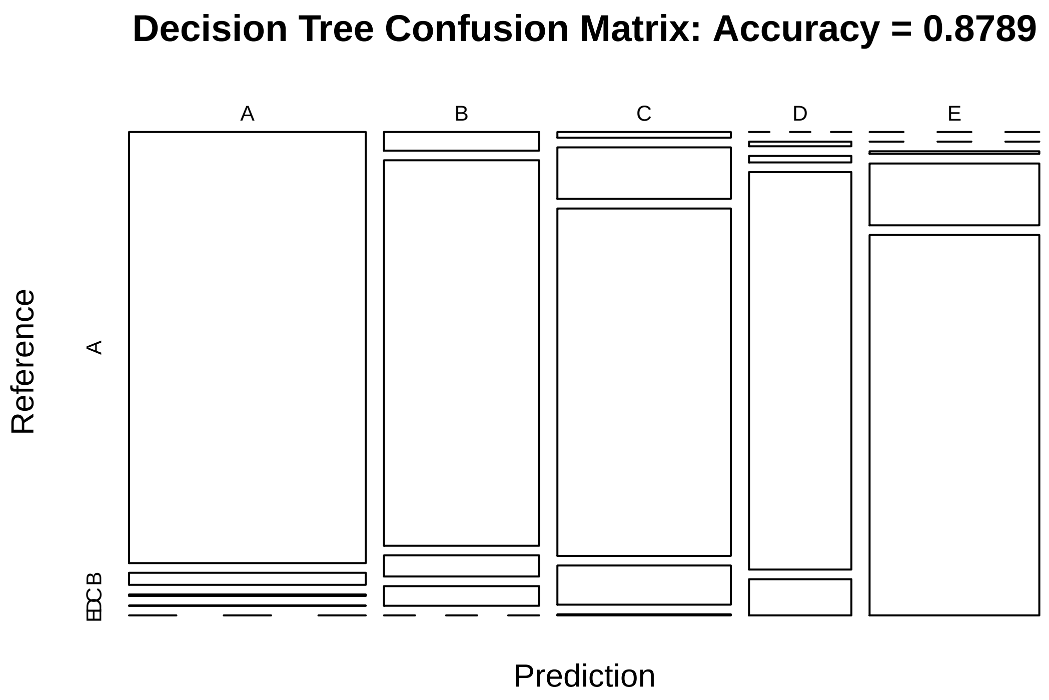
```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTesting$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

The assignment rubric asks to use data from accelerometers on the `belt`, `forearm`, `arm`, and `dumbell`, so the features are extracted based on these keywords.

```
# filter columns on: belt, forearm, arm, dumbell
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy =", rou
```



Prediction with Random Forests.

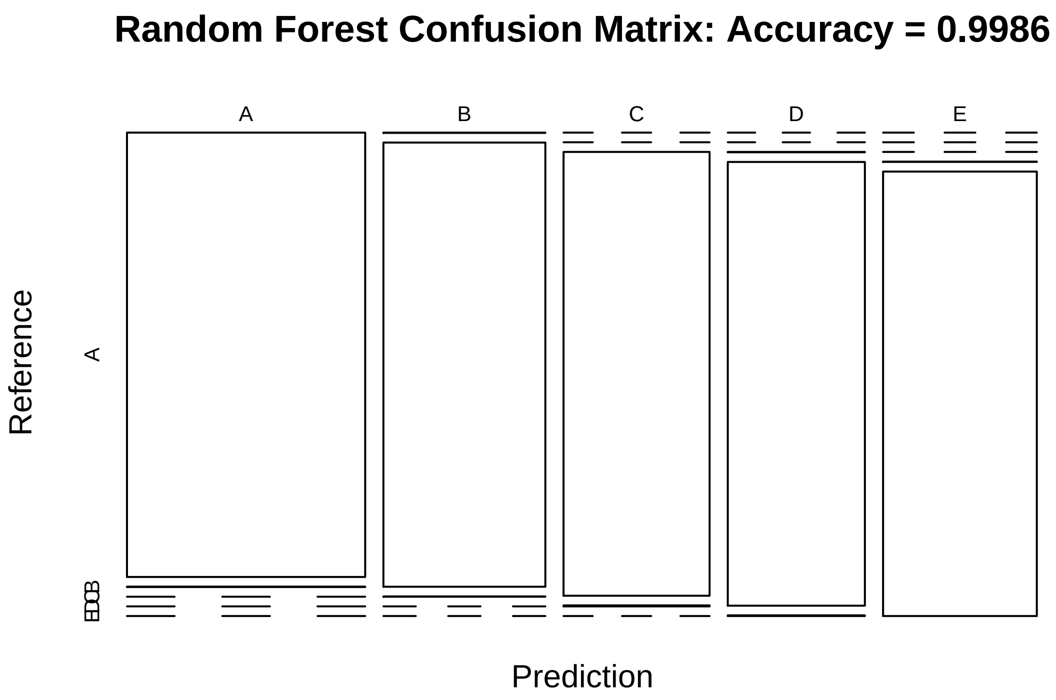
Random Forests can deal with “small and large problems, high-order interactions, correlated predictor variables are used. Method not only for prediction, but also to assess variable importance

```
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 2231    2    0    0    0
##          B    1 1516    1    0    0
##          C    0    0 1366    3    0
##          D    0    0    1 1282    2
##          E    0    0    0    1 1440
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9975, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9985   0.9969   0.9986
## Specificity            0.9996   0.9997   0.9995   0.9995   0.9998
## Pos Pred Value         0.9991   0.9987   0.9978   0.9977   0.9993
## Neg Pred Value         0.9998   0.9997   0.9997   0.9994   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1741   0.1634   0.1835
## Detection Prevalence   0.2846   0.1935   0.1745   0.1638   0.1837
## Balanced Accuracy      0.9996   0.9992   0.9990   0.9982   0.9992
```

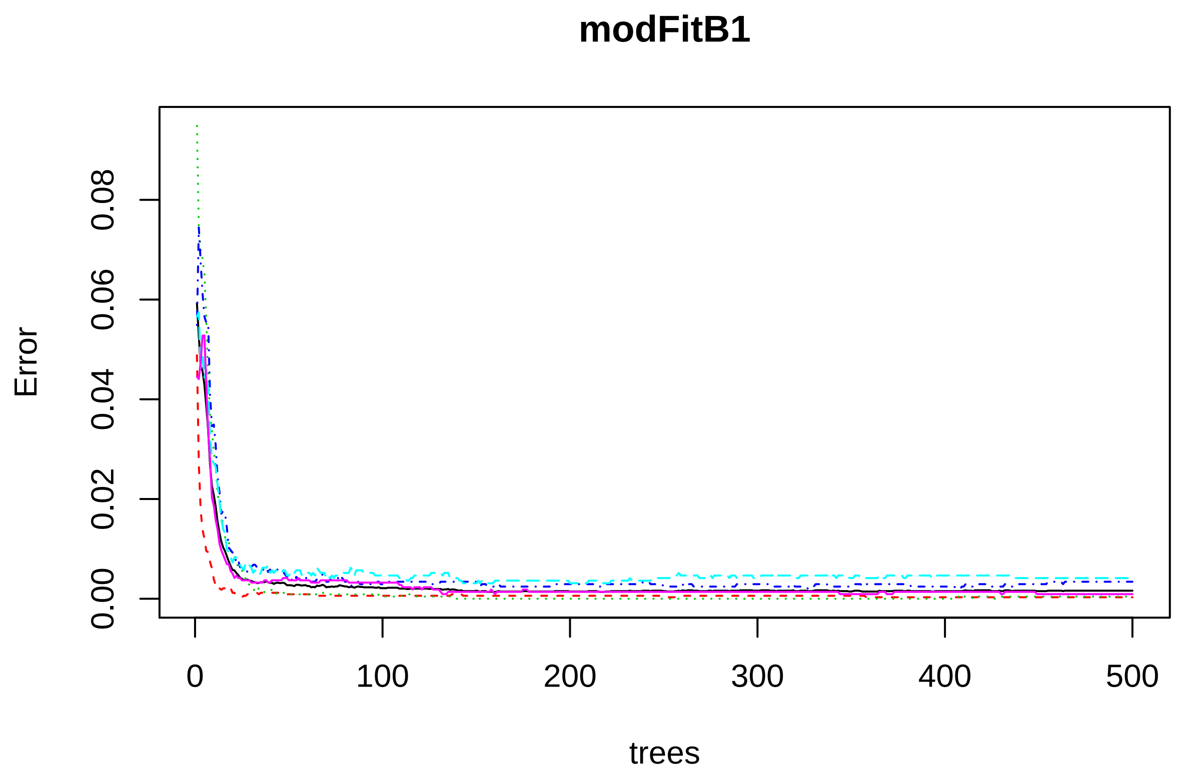
Plot Random Forest Confusion Matrix with Accuracy

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



###Plot modfit values - Error vs Tree

```
plot(modFitB1)
```



Prediction with Generalized Boosted Regression

```r
set.seed(12345)
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

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

```
## 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
```

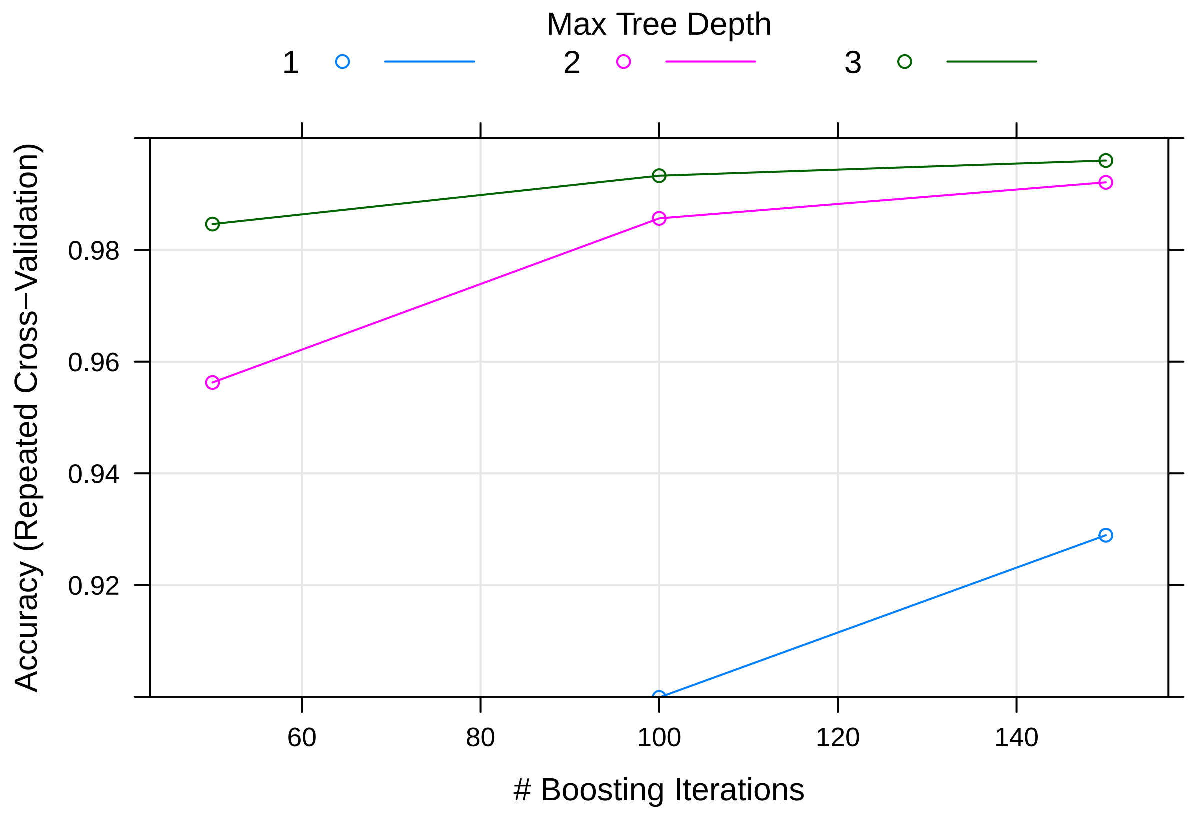
```
gbmFinMod1 <- gbmFit1$finalModel

gbmPredTest <- predict(gbmFit1, newdata=myTesting)
gbmAccuracyTest <- confusionMatrix(gbmPredTest, myTesting$classe)
gbmAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1515    0    0    0
##          C    0    1 1361    5    0
##          D    0    0    7 1272    0
##          E    0    0    0    9 1442
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.9953, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.996
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9980   0.9949   0.9891   1.0000
## Specificity            0.9996   0.9998   0.9991   0.9989   0.9986
## Pos Pred Value         0.9991   0.9993   0.9956   0.9945   0.9938
## Neg Pred Value         0.9998   0.9995   0.9989   0.9979   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1931   0.1735   0.1621   0.1838
## Detection Prevalence   0.2846   0.1932   0.1742   0.1630   0.1849
## Balanced Accuracy      0.9996   0.9989   0.9970   0.9940   0.9993
```

Plot Generalized Boosted Regression for the Max Tree Depth vs. Accuracy

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



Predicting Results on the Test Data

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

```
##  1  2 31  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
```

For this dataset, random forest method is way better than classification tree method. 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%. This may be due to the fact that many predictors are highly correlated. Random forests chooses a subset of predictors at each split and decorrelate the trees. This leads to high accuracy, although this algorithm is sometimes difficult to interpret and computationally inefficient.

The expected out-of-sample error is 100-99.89 = 0.11%.

```
#predictionB2[2]
#write.table(predictionB2[2], file="c:/test123.txt", row.names =  FALSE, col.names =FALSE)


#for (i in 1:20) {
#filename = paste0("c:/problem_id_",i,".txt")
#write.table(predictionB2[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
#}
```

```r
# Write the results to a text file for submission
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){

    filename = paste0("c:/problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(predictionB2)
```