Practical Machine Learning - Course Project

Aswin Kumar T

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, 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.

The Goal of the project :

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.

Load the data

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(knitr)
library(e1071)
set.seed(100)

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!",""))
```

Creaing 2 training sets:

```
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 : nearzerovariance variables , the 1st column , clearing variables with NA values.

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

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

myTraining <- myTraining[c(-1)]

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)

clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])
myTesting <- myTesting[clean1]
testing <- testing[clean2]

dim(myTesting)
```

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

```
dim(testing)
```

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

Join the data into same data type.

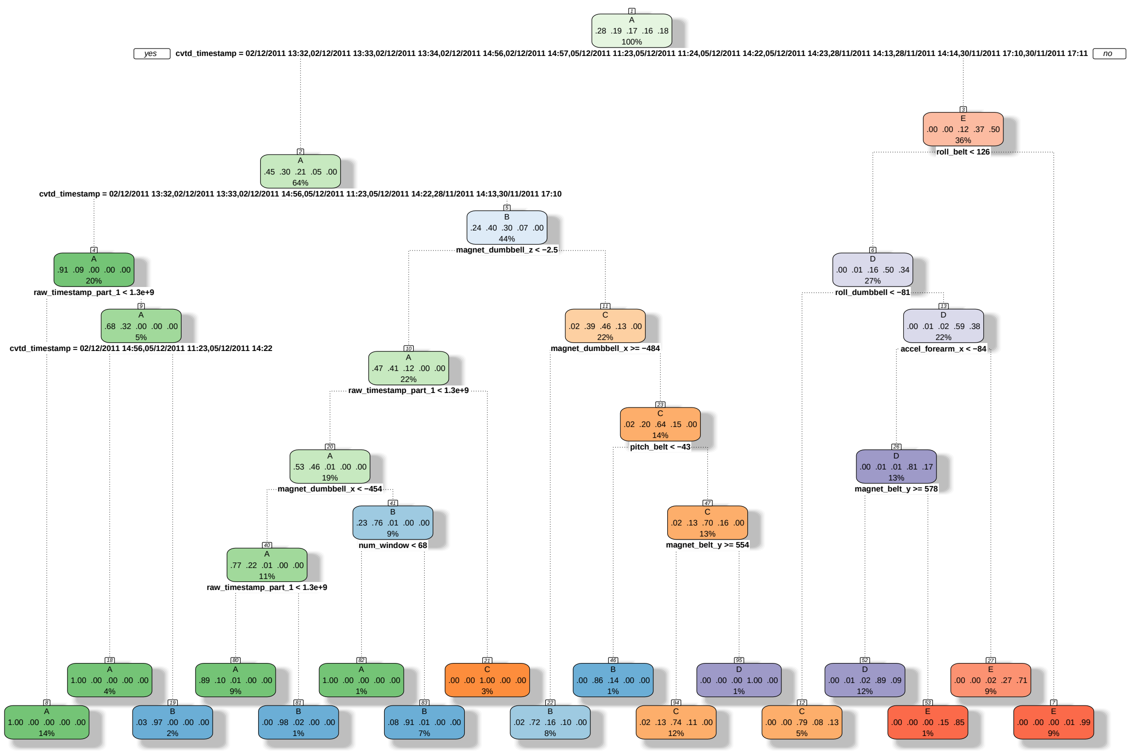
```
for (i in 1:length(testing) ) {
    for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}

# To get the same class between testing and myTraining
```

```r
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

Prediction with Decision Trees

```r
set.seed(100)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```



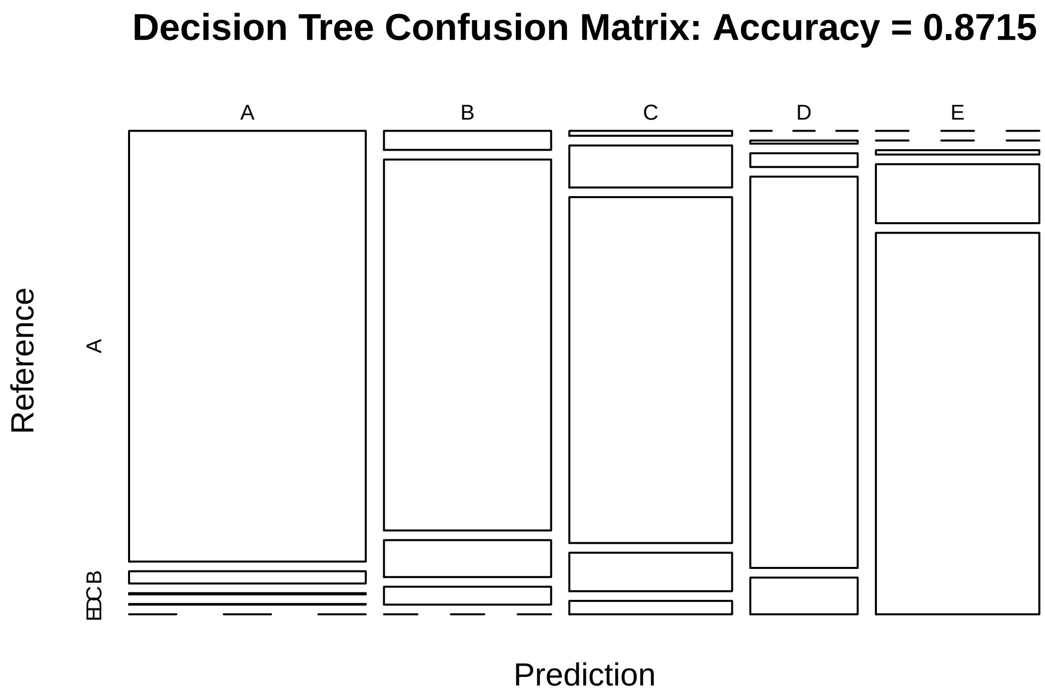
Rattle 2018−Jun−10 19:52:21 Aswin

```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 2148   61    6    3    0
##          B   67 1306  130   63    0
##          C   17  144 1186  132   46
##          D    0    7   31  885   83
##          E    0    0   15  203 1313
##
## Overall Statistics
##
```

```
##                Accuracy : 0.8715
##                  95% CI : (0.8639, 0.8789)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8374
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9624   0.8603   0.8670   0.6882   0.9105
## Specificity            0.9875   0.9589   0.9477   0.9816   0.9660
## Pos Pred Value         0.9684   0.8340   0.7777   0.8797   0.8576
## Neg Pred Value         0.9851   0.9662   0.9712   0.9414   0.9796
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2738   0.1665   0.1512   0.1128   0.1673
## Detection Prevalence   0.2827   0.1996   0.1944   0.1282   0.1951
## Balanced Accuracy      0.9749   0.9096   0.9073   0.8349   0.9382
```

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

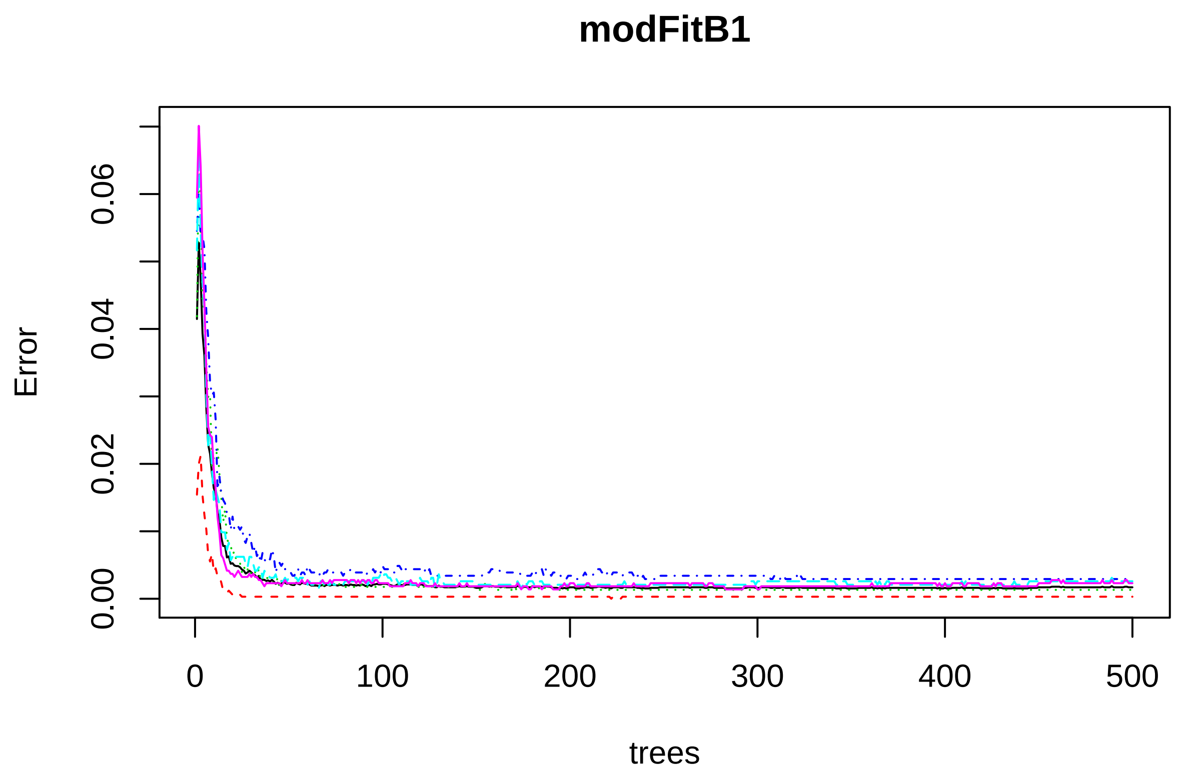


Prediction with Random Forests

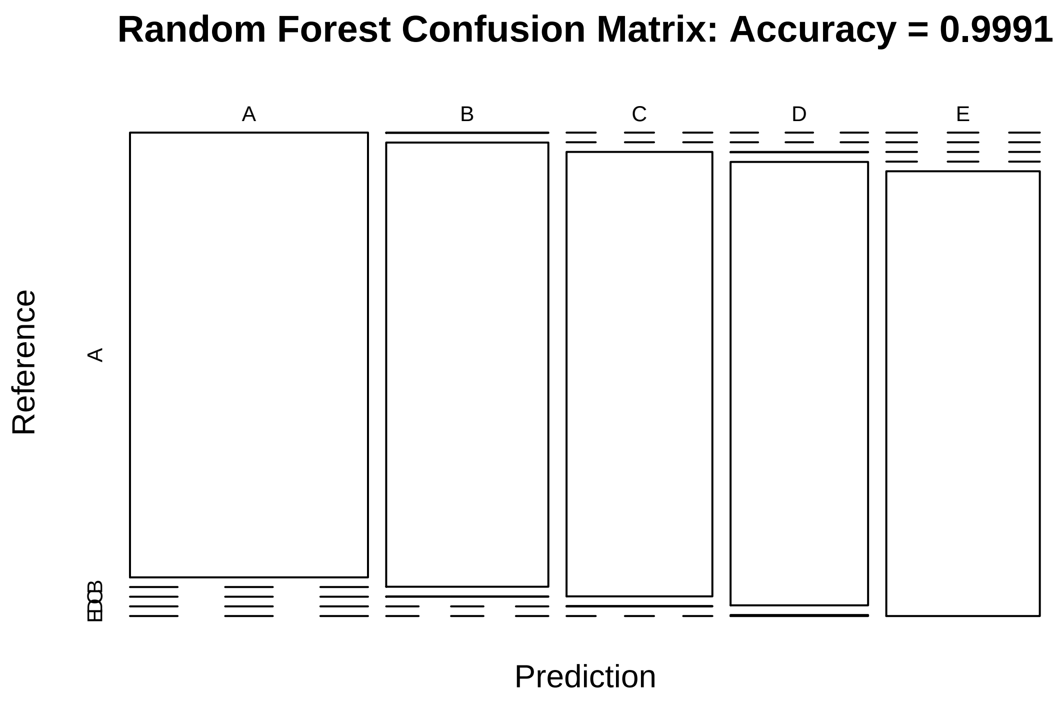
```r
set.seed(100)
modFitB1 <- randomForest(classe ~ ., data=myTraining)
predictionB1 <- predict(modFitB1, myTesting, type = "class")
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    0    0    0    0
##          B    1 1518    1    0    0
##          C    0    0 1366    1    0
##          D    0    0    1 1285    3
##          E    0    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.9991
##                  95% CI : (0.9982, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9989
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   1.0000   0.9985   0.9992   0.9979
## Specificity            1.0000   0.9997   0.9998   0.9994   1.0000
## Pos Pred Value         1.0000   0.9987   0.9993   0.9969   1.0000
## Neg Pred Value         0.9998   1.0000   0.9997   0.9998   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1741   0.1638   0.1834
## Detection Prevalence   0.2843   0.1937   0.1742   0.1643   0.1834
## Balanced Accuracy      0.9998   0.9998   0.9992   0.9993   0.9990
```

```r
plot(modFitB1)
```



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



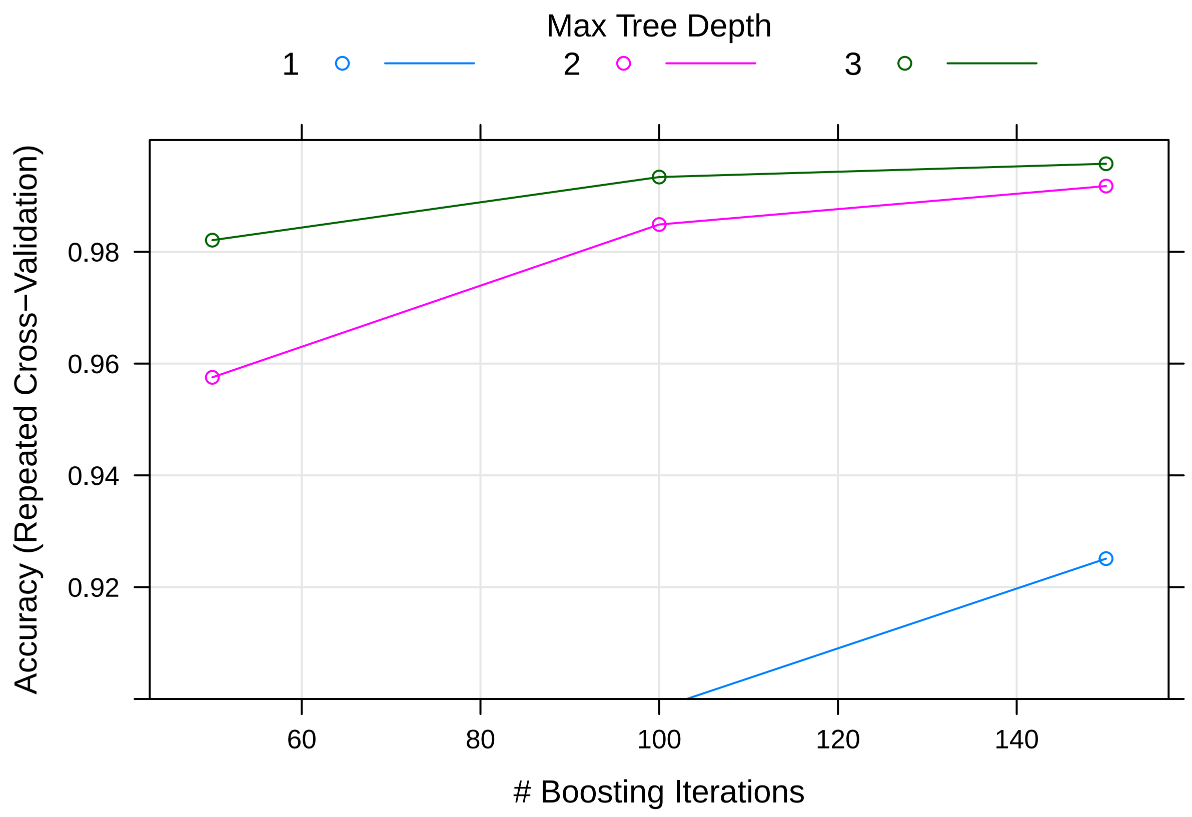
Prediction with Generalized Boosted Regression

```r
set.seed(100)
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 2229    0    0    0    0
##          B    3 1517    1    0    0
##          C    0    1 1359    1    0
##          D    0    0    8 1284    6
##          E    0    0    0    1 1436
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9959, 0.9983)
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                    Kappa : 0.9966
##   Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9993   0.9934   0.9984   0.9958
## Specificity            1.0000   0.9994   0.9997   0.9979   0.9998
## Pos Pred Value         1.0000   0.9974   0.9985   0.9892   0.9993
## Neg Pred Value         0.9995   0.9998   0.9986   0.9997   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1933   0.1732   0.1637   0.1830
## Detection Prevalence   0.2841   0.1939   0.1735   0.1654   0.1832
## Balanced Accuracy      0.9993   0.9994   0.9966   0.9982   0.9978
```

```
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%.

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

```
##  1  2  3 41  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
```