Final project

Sadykova Sabina 2018-10-14

# 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).

# Getting and loading the data

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)  
library(rpart.plot)  
library(RColorBrewer)  
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

library(knitr)

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

inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)  
myTraining <- training[inTrain, ]  
myTesting <- training[-inTrain, ]  
dim(myTraining); dim(myTesting)

## [1] 13737 160

## [1] 5885 160

# Data cleaning

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

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]  
 }   
 }   
 }  
}  
  
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] 5885 58

dim(testing)

## [1] 20 57

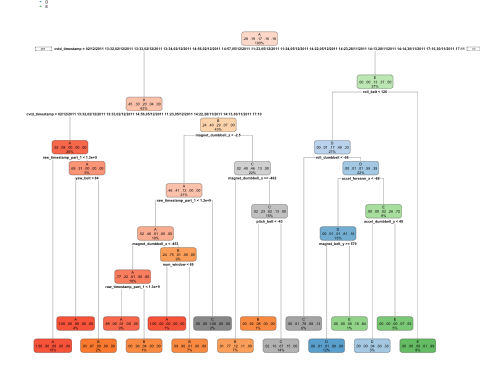
Coerce the data into the same type

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

# Prediction Algorithms

### Prediction with Decision Trees

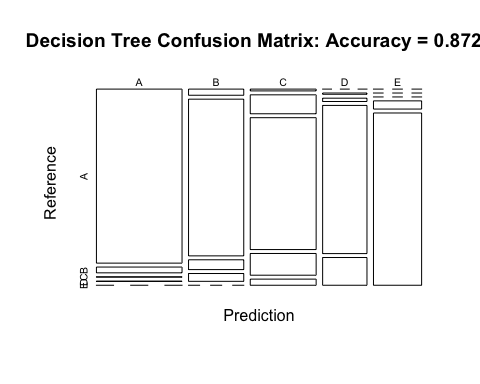
set.seed(12345)  
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")  
rpart.plot(modFitA1)



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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1624 54 4 2 0  
## B 37 943 59 47 0  
## C 13 136 947 156 44  
## D 0 6 16 716 134  
## E 0 0 0 43 904  
##   
## Overall Statistics  
##   
## Accuracy : 0.8724   
## 95% CI : (0.8636, 0.8808)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8386   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9701 0.8279 0.9230 0.7427 0.8355  
## Specificity 0.9858 0.9699 0.9282 0.9683 0.9910  
## Pos Pred Value 0.9644 0.8683 0.7307 0.8211 0.9546  
## Neg Pred Value 0.9881 0.9592 0.9828 0.9505 0.9640  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2760 0.1602 0.1609 0.1217 0.1536  
## Detection Prevalence 0.2862 0.1845 0.2202 0.1482 0.1609  
## Balanced Accuracy 0.9779 0.8989 0.9256 0.8555 0.9133

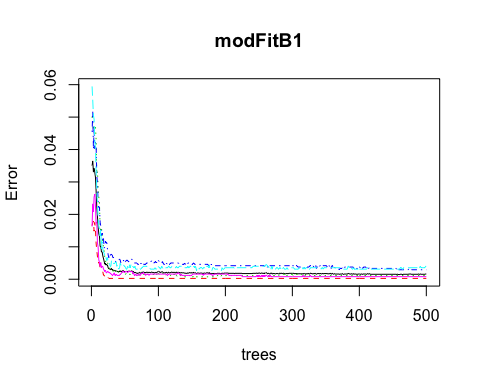
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy =", round(cmtree$overall['Accuracy'], 4)))

 ### Prediction with Random Forests

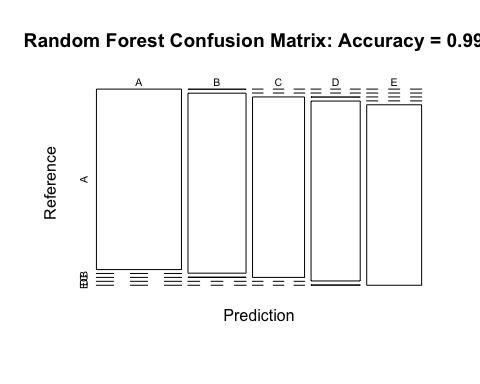
set.seed(12345)  
modFitB1 <- randomForest(classe ~ ., data=myTraining)  
predictionB1 <- predict(modFitB1, myTesting, type = "class")  
cmrf <- confusionMatrix(predictionB1, myTesting$classe)  
cmrf

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1673 0 0 0 0  
## B 1 1139 2 0 0  
## C 0 0 1023 0 0  
## D 0 0 1 964 2  
## E 0 0 0 0 1080  
##   
## Overall Statistics  
##   
## Accuracy : 0.999   
## 95% CI : (0.9978, 0.9996)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9987   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9994 1.0000 0.9971 1.0000 0.9982  
## Specificity 1.0000 0.9994 1.0000 0.9994 1.0000  
## Pos Pred Value 1.0000 0.9974 1.0000 0.9969 1.0000  
## Neg Pred Value 0.9998 1.0000 0.9994 1.0000 0.9996  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2843 0.1935 0.1738 0.1638 0.1835  
## Detection Prevalence 0.2843 0.1941 0.1738 0.1643 0.1835  
## Balanced Accuracy 0.9997 0.9997 0.9985 0.9997 0.9991

plot(modFitB1)



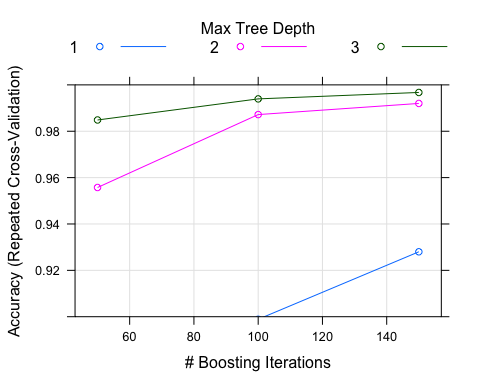
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", round(cmrf$overall['Accuracy'], 4)))

 ### 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 1673 2 0 0 0  
## B 1 1135 1 0 0  
## C 0 1 1016 3 0  
## D 0 1 9 956 3  
## E 0 0 0 5 1079  
##   
## Overall Statistics  
##   
## Accuracy : 0.9956   
## 95% CI : (0.9935, 0.9971)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9944   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9994 0.9965 0.9903 0.9917 0.9972  
## Specificity 0.9995 0.9996 0.9992 0.9974 0.9990  
## Pos Pred Value 0.9988 0.9982 0.9961 0.9866 0.9954  
## Neg Pred Value 0.9998 0.9992 0.9979 0.9984 0.9994  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2843 0.1929 0.1726 0.1624 0.1833  
## Detection Prevalence 0.2846 0.1932 0.1733 0.1647 0.1842  
## Balanced Accuracy 0.9995 0.9980 0.9947 0.9945 0.9981

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 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