Final\_Project-Practical Machine Learning

#LOAD THE DATA SET  
library(caret); library(rattle); library(rpart); library(rpart.plot)

## Loading required package: lattice

## Loading required package: ggplot2

## 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(randomForest); library(repmis)

## 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

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!",""))  
  
#PARTITIONING THE TRAINING DATA SET INTO TWO  
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)  
inTrain <- createDataPartition(training$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]  
myTraining <- myTraining[c(-1)] #will remove the first column of the training set  
  
#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]}}}}  
  
## Set back to the original variable name  
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)

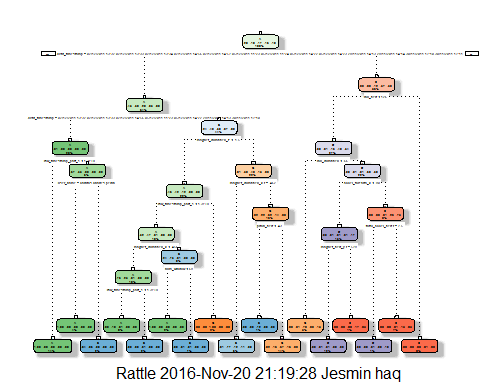
## [1] 7846 58

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,]

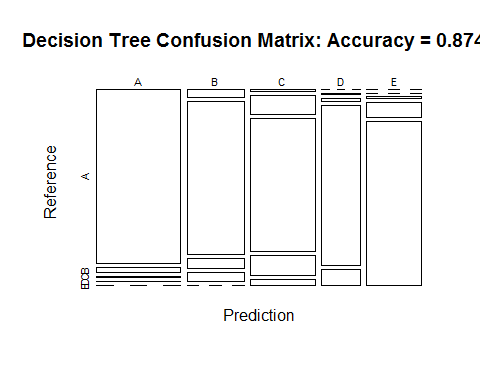
#PREDICTION WITH DECISION TREES  
set.seed(12345)  
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")  
fancyRpartPlot(modFitA1)



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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2138 56 10 3 0  
## B 68 1277 82 72 0  
## C 26 178 1246 181 57  
## D 0 7 21 903 87  
## E 0 0 9 127 1298  
##   
## Overall Statistics  
##   
## Accuracy : 0.8746   
## 95% CI : (0.8671, 0.8818)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8414   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9579 0.8412 0.9108 0.7022 0.9001  
## Specificity 0.9877 0.9649 0.9318 0.9825 0.9788  
## Pos Pred Value 0.9687 0.8519 0.7382 0.8870 0.9052  
## Neg Pred Value 0.9833 0.9620 0.9802 0.9439 0.9775  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2725 0.1628 0.1588 0.1151 0.1654  
## Detection Prevalence 0.2813 0.1911 0.2151 0.1297 0.1828  
## Balanced Accuracy 0.9728 0.9031 0.9213 0.8423 0.9395

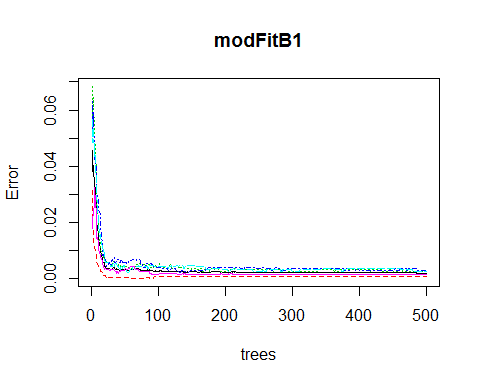
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy =", round(cmtree$overall['Accuracy'], 4)))



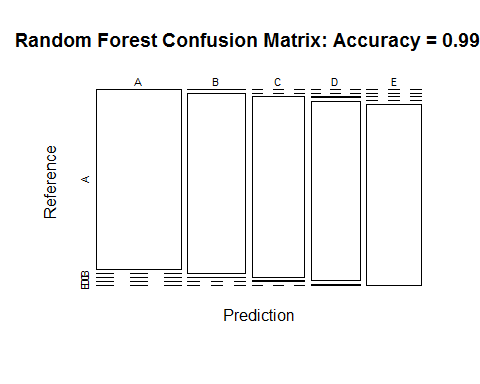
# PREDICTION WITH RANDOM FORESTS  
set.seed(12345)  
modFitB1 <- randomForest(classe ~ ., data=myTraining)  
predictionB1 <- predict(modFitB1, myTesting, type = "class")  
cmrf <- confusionMatrix(predictionB1, myTesting$classe)  
cmrf

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 2230 0 0 0 0  
## B 2 1518 2 0 0  
## C 0 0 1361 3 0  
## D 0 0 5 1283 3  
## E 0 0 0 0 1439  
##   
## Overall Statistics  
##   
## Accuracy : 0.9981   
## 95% CI : (0.9968, 0.9989)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9976   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9991 1.0000 0.9949 0.9977 0.9979  
## Specificity 1.0000 0.9994 0.9995 0.9988 1.0000  
## Pos Pred Value 1.0000 0.9974 0.9978 0.9938 1.0000  
## Neg Pred Value 0.9996 1.0000 0.9989 0.9995 0.9995  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2842 0.1935 0.1735 0.1635 0.1834  
## Detection Prevalence 0.2842 0.1940 0.1738 0.1645 0.1834  
## Balanced Accuracy 0.9996 0.9997 0.9972 0.9982 0.9990

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)

## 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

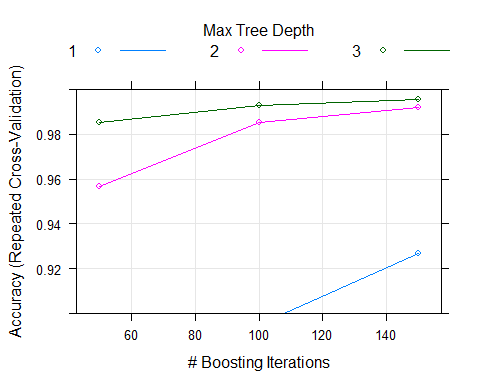
## Loaded gbm 2.1.1

## Loading required package: plyr

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 2225 3 0 0 0  
## B 7 1512 2 0 0  
## C 0 3 1353 2 0  
## D 0 0 13 1284 1  
## E 0 0 0 0 1441  
##   
## Overall Statistics  
##   
## Accuracy : 0.996   
## 95% CI : (0.9944, 0.9973)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.995   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9969 0.9960 0.9890 0.9984 0.9993  
## Specificity 0.9995 0.9986 0.9992 0.9979 1.0000  
## Pos Pred Value 0.9987 0.9941 0.9963 0.9892 1.0000  
## Neg Pred Value 0.9988 0.9991 0.9977 0.9997 0.9998  
## Prevalence 0.2845 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2836 0.1927 0.1724 0.1637 0.1837  
## Detection Prevalence 0.2840 0.1939 0.1731 0.1654 0.1837  
## Balanced Accuracy 0.9982 0.9973 0.9941 0.9982 0.9997

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



#PREDICTION RESULT ON THE TEST DATA  
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

comment: The Random Forrest gave the most accuracy with 99.89% accuracy than the Decision Trees and the Generalized Boosted Regression. The out of sample rate is .11%