R Notebook

Code

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Hide

```
library(caret)
```

```
Loading required package: lattice
Loading required package: ggplot2
Want to understand how all the pieces fit together? See the R for Data Science book:
http://r4ds.had.co.nz/
```

Hide

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

```
Rattle: A free graphical interface for data science with R.
Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
Type 'rattle()' to shake, rattle, and roll your data.
```

Hide

```
library(randomForest)
```

```
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.

Attaching package: ‘randomForest’

The following object is masked from ‘package:rattle’:

    importance

The following object is masked from ‘package:ggplot2’:

    margin
```

Hide

```
library(knitr)
#Data Preparation
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!",""))
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
```

After partitioning the data, we remove the NearZeroVariance Variables and remove the variable which contain mostly NAs. Finally, we coerce our different dataset into the same type.

Hide

```
#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)]
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])  # remove the classe column
myTesting <- myTesting[clean1]         # allow only variables in myTesting that are a
lso in myTraining
testing <- testing[clean2]             # allow only variables in testing that are als
o in myTraining
dim(myTesting)
```

```
[1] 7846   58
```

Hide

```
dim(testing)
```

```
[1] 20 57
```

Hide

```
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,]
```

The first prediction we make is with decision tree

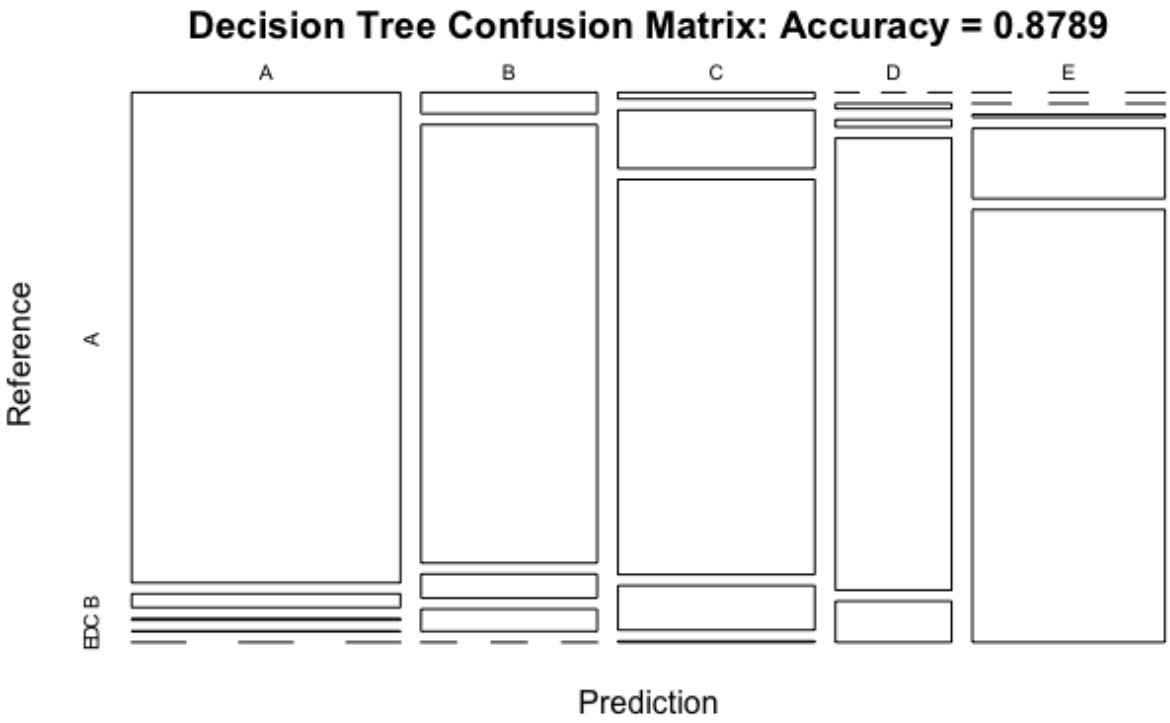
Hide

```
#Predictions with Decision Trees
set.seed(12345)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
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
```

Hide

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

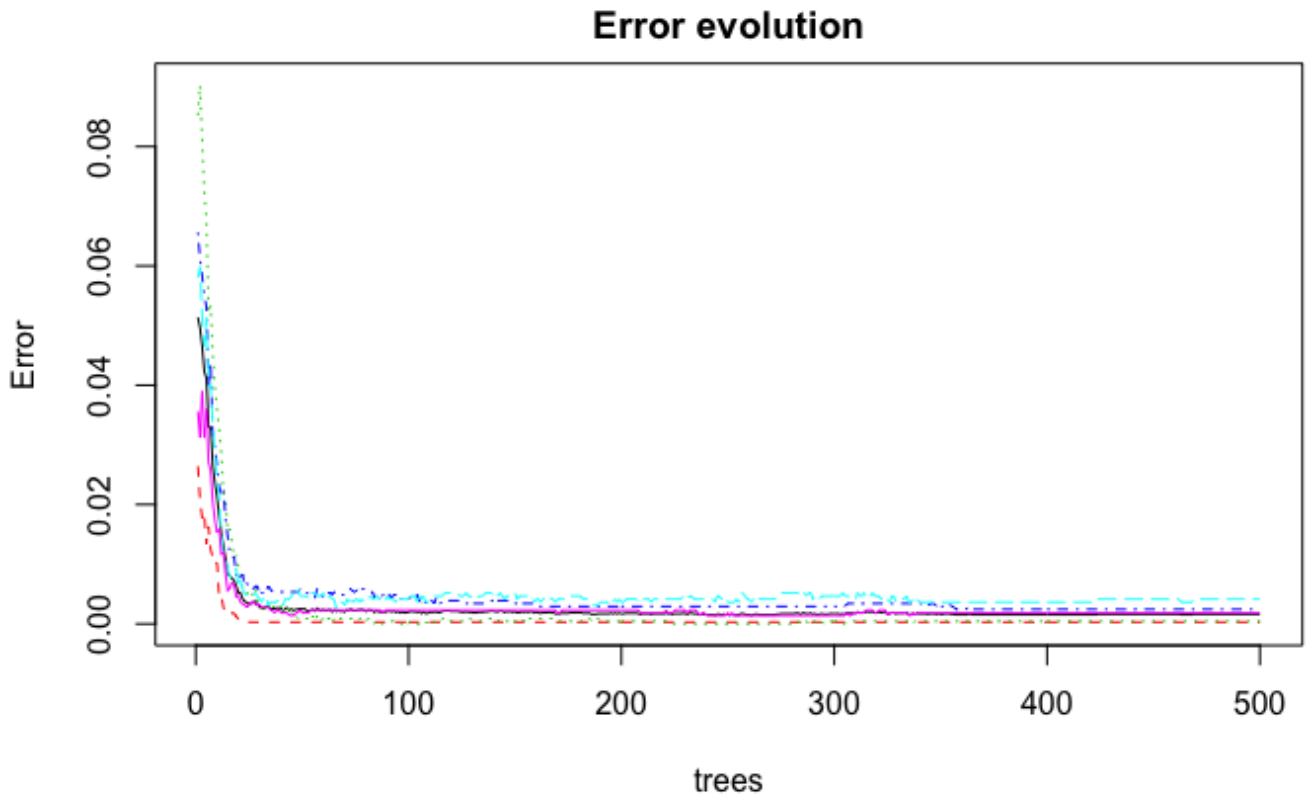


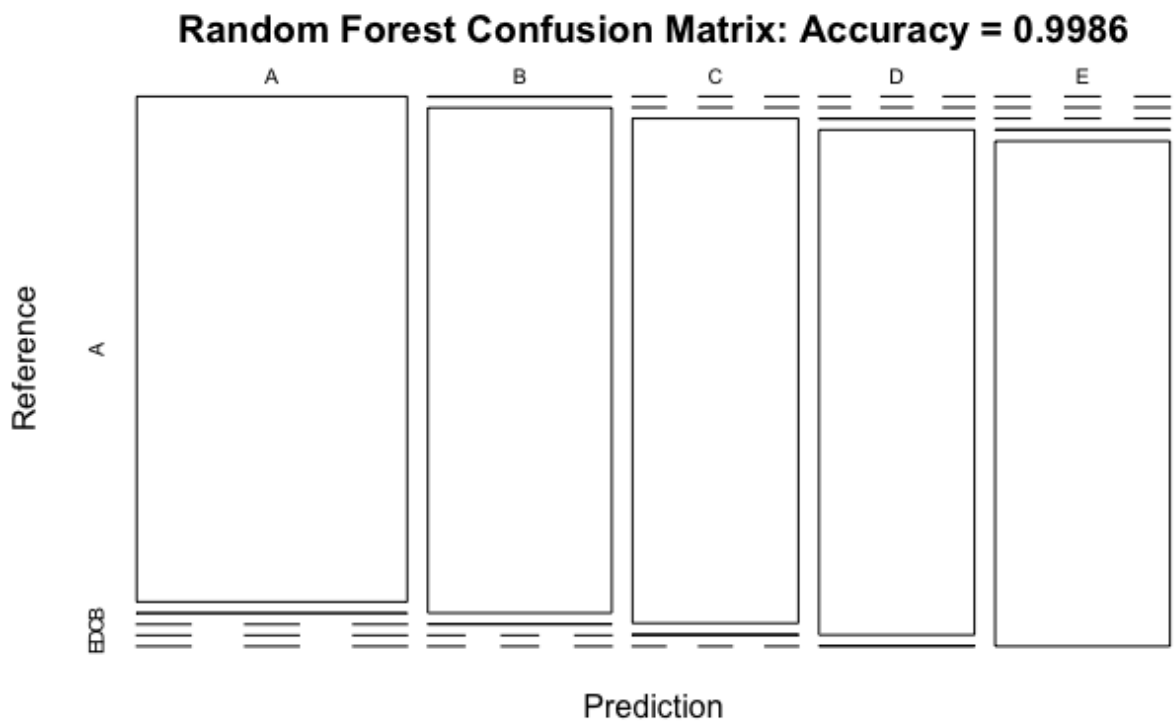
A second method for prediction is using random forests.

Hide

```
#Prediction with random forest
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 1281    1
         E    0    0    0    2 1441

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   0.9961   0.9993
Specificity            0.9996   0.9997   0.9995   0.9997   0.9997
Pos Pred Value         0.9991   0.9987   0.9978   0.9984   0.9986
Neg Pred Value         0.9998   0.9997   0.9997   0.9992   0.9998
Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
Detection Rate         0.2843   0.1932   0.1741   0.1633   0.1837
Detection Prevalence   0.2846   0.1935   0.1745   0.1635   0.1839
Balanced Accuracy      0.9996   0.9992   0.9990   0.9979   0.9995
```





The last method is using Generalized Boosted Regression using repeated cross validation.

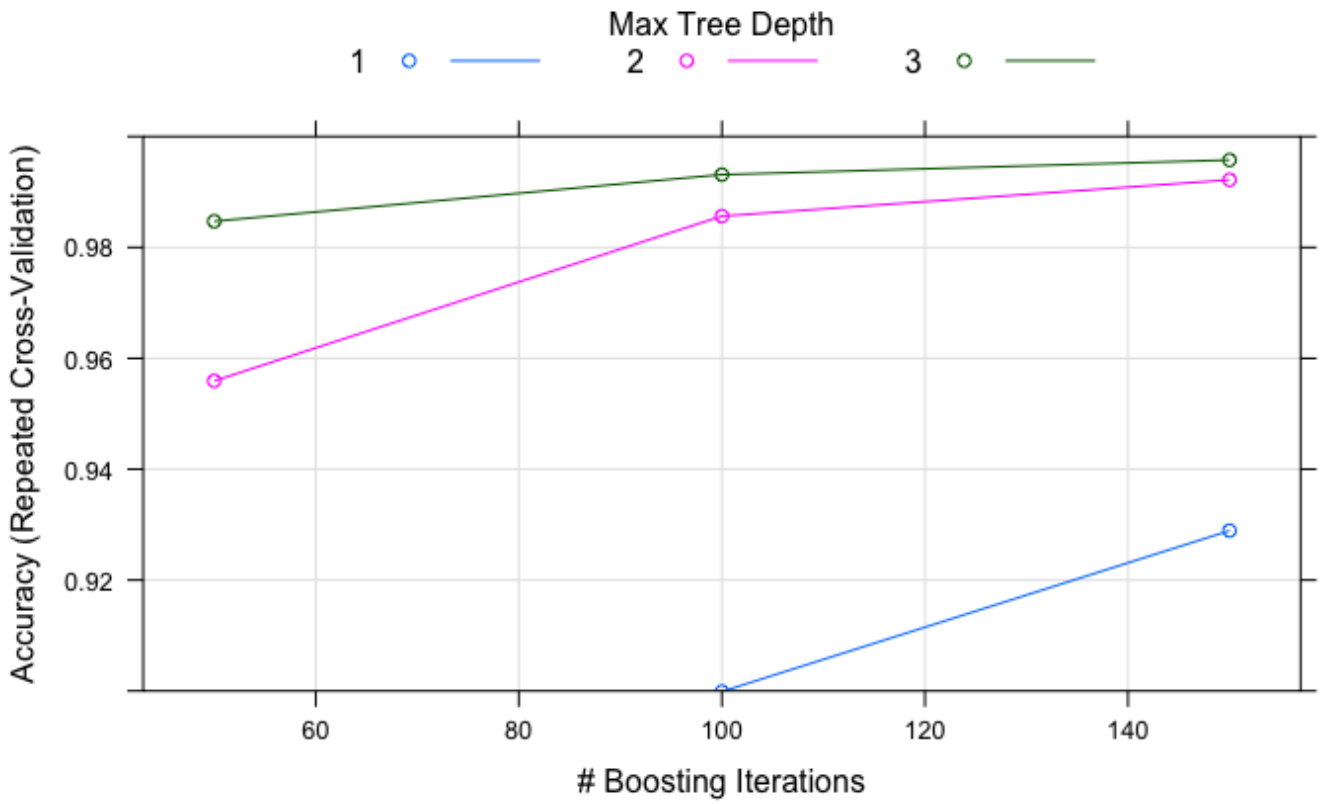
Hide

```
#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)
```

Hide

```
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 2230    4    0    0    0
         B    2 1512    1    0    0
         C    0    2 1361    5    0
         D    0    0    6 1272    0
         E    0    0    0    9 1442

Overall Statistics

               Accuracy : 0.9963
                 95% CI : (0.9947, 0.9975)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9953

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9991   0.9960   0.9949   0.9891   1.0000
Specificity            0.9993   0.9995   0.9989   0.9991   0.9986
Pos Pred Value         0.9982   0.9980   0.9949   0.9953   0.9938
Neg Pred Value         0.9996   0.9991   0.9989   0.9979   1.0000
Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
Detection Rate         0.2842   0.1927   0.1735   0.1621   0.1838
Detection Prevalence   0.2847   0.1931   0.1744   0.1629   0.1849
Balanced Accuracy      0.9992   0.9978   0.9969   0.9941   0.9993
```



And finally, we are using the model 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%.

Hide

```
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
```

Hide

```
# Write the results to a text file for submission
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

 pml_write_files(predictionB2)
```