Rujuta_Coursera_Machine_Learning_Project_Jan16

Rujuta Joshi

Thursday, January 28, 2016

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).

Methodology

Step 1A : Loading up all the relevant packages for r.

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.1.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.1.3
```

```r
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.1.3
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.1.3
```

Step 1B : downloading the test data and the train data.

```
#traindataurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#download.file(traindataurl, destfile="./TrainData.csv")

TrainData <- read.csv("F:/Rujuta/Coursera/MachineLearning/TrainData.csv", header=TRUE)
dim(TrainData)
```

```
## [1] 19622   160
```

```
# names(TrainData) - keeping it in comment form so as to not get the detailed list in the report.
# for cross validation, i need to make a part of my data, for training and testing, and then use
# the test data only for running the final model.

#testdataurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#download.file(testdataurl, destfile="./TestData.csv")
TestData <- read.csv("F:/Rujuta/Coursera/MachineLearning/TestData.csv", header=TRUE)

dim(TestData)
```

```
## [1]  20 160
```

Step 2 : Exploratory Analysis of the data, to understand it. , here we also make sure all relevant packages are installed and opened.

```
# Exploratory Analysis

#str(TrainData) keeping it in comment form to avoid the lengthy list
# understand the missing values
na_count2 <- sapply(TrainData, function(y) sum(length(which(is.na(y)))))
#str(na_count2)

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

library(randomForest)
library(knitr)
```

Steps in Data Preparation

The following steps are taken. 1. we need to predict values of the 20 test data points. However in order to train the model that we build, we need another test data, where we can check the accuracy. so we create a partition in the data, and create another validation data set, called Test set, where we can check how well is the model doing. 2. We make sure that the NA columns are removed, the data is in the same format in these 3 sets, and remove the first column.

1. making the partition in Training Data - training and validation data sets

```
Trainpart <- createDataPartition (TrainData$classe, p=0.7, list=FALSE)
TrainingSubset <- TrainData[Trainpart,]
TestSubset <- TrainData[-Trainpart,]
dim(TrainingSubset)
```

```
## [1] 13737   160
```

```
dim(TestSubset)
```

```
## [1] 5885  160
```

2. Clean the data by a] Clearing near zero variance variables, b] Clearing empty columns

```
nz <- nearZeroVar(TrainingSubset, saveMetrics=TRUE)
dim(nz)
```

```
## [1] 160   4
```

```
myTraining <- TrainingSubset[,nz$nzv==FALSE]
dim(myTraining)
```

```
## [1] 13737   107
```

```
nz2 <- nearZeroVar(TestSubset, saveMetrics=TRUE)
dim(nz2)
```

```
## [1] 160   4
```

```
myTest <- TestSubset[, nz2$nzv==FALSE]
dim(myTest)
```

```
## [1] 5885  108
```

```r
#2b. Removing the first column
myTraining <- myTraining [c(-1)]
#2c  Remove the columns that have mostly NA- Removing for 60%

trainingroughwork <- myTraining

for (i in 1: length(myTraining))
{if (sum(is.na(myTraining[,i]))/nrow(myTraining) >= 0.6)
{for(j in 1: length(trainingroughwork))
{if (length(grep(names(myTraining[i]),
names(trainingroughwork)[j]))==1)
{trainingroughwork <- trainingroughwork[,-j]}
}

}
}

myTraining <- trainingroughwork

#2d now do this procedure for the myTest data(validation set ) and the testing data
smallsetcolumnnames <- colnames(myTraining)
smallsetcolname2 <- colnames(myTraining[,-58])

myTest <- myTest[smallsetcolumnnames]
TestData <- TestData[smallsetcolname2]

# 2e - make sure that the data in Training set, validation set and the test set is in the same format




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

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

Step 3 : Prediction using Decision Trees and Random Forest and Generalised boosting regression.

we check with all 3 methods, check which one has the best accuracy and go ahead with the model for that method. for this data, it turns out that random forest works best for us.

```r
# Prediction using Decision Tree Analysis
```

```r
set.seed(11111)
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.1.3
```

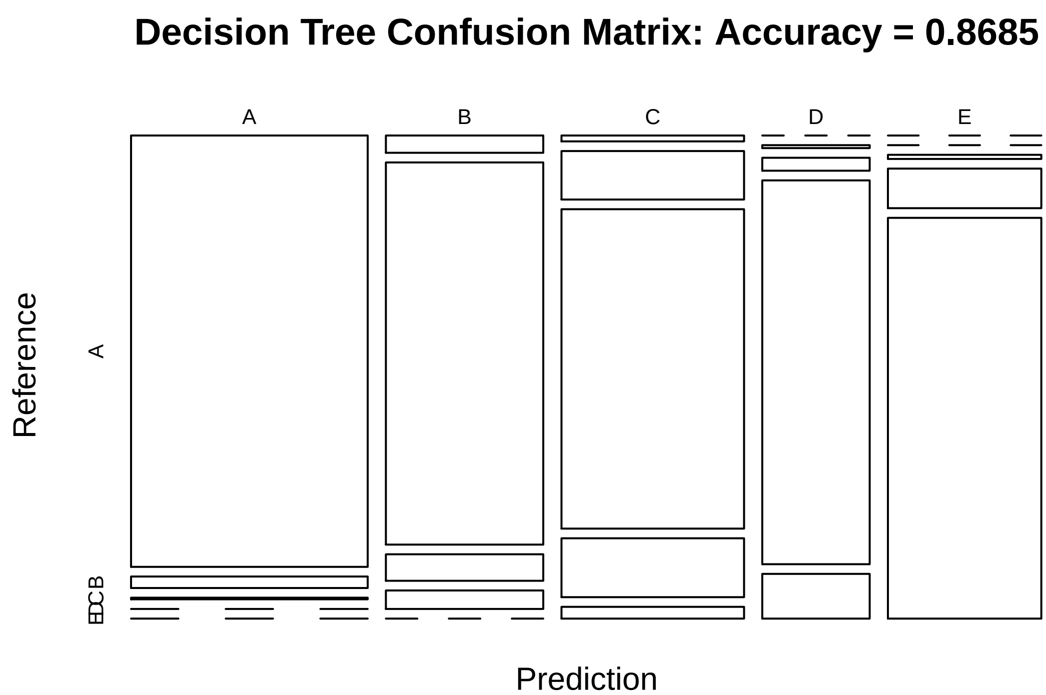
```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
ModelFitA1 <- rpart(classe~.,data=myTraining,method="class")

predictionsA1 <- predict(ModelFitA1, myTest, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTest$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1614   43    6    0    0
##          B   43  951   66   46    0
##          C   17  140  922  170   34
##          D    0    5   22  652   76
##          E    0    0   10   96  972
##
## Overall Statistics
##
##                Accuracy : 0.8685
##                  95% CI : (0.8596, 0.877)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8336
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9642   0.8349   0.8986   0.6763   0.8983
## Specificity            0.9884   0.9673   0.9257   0.9791   0.9779
## Pos Pred Value         0.9705   0.8599   0.7186   0.8636   0.9017
## Neg Pred Value         0.9858   0.9607   0.9774   0.9392   0.9771
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2743   0.1616   0.1567   0.1108   0.1652
## Detection Prevalence   0.2826   0.1879   0.2180   0.1283   0.1832
## Balanced Accuracy      0.9763   0.9011   0.9122   0.8277   0.9381
```

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



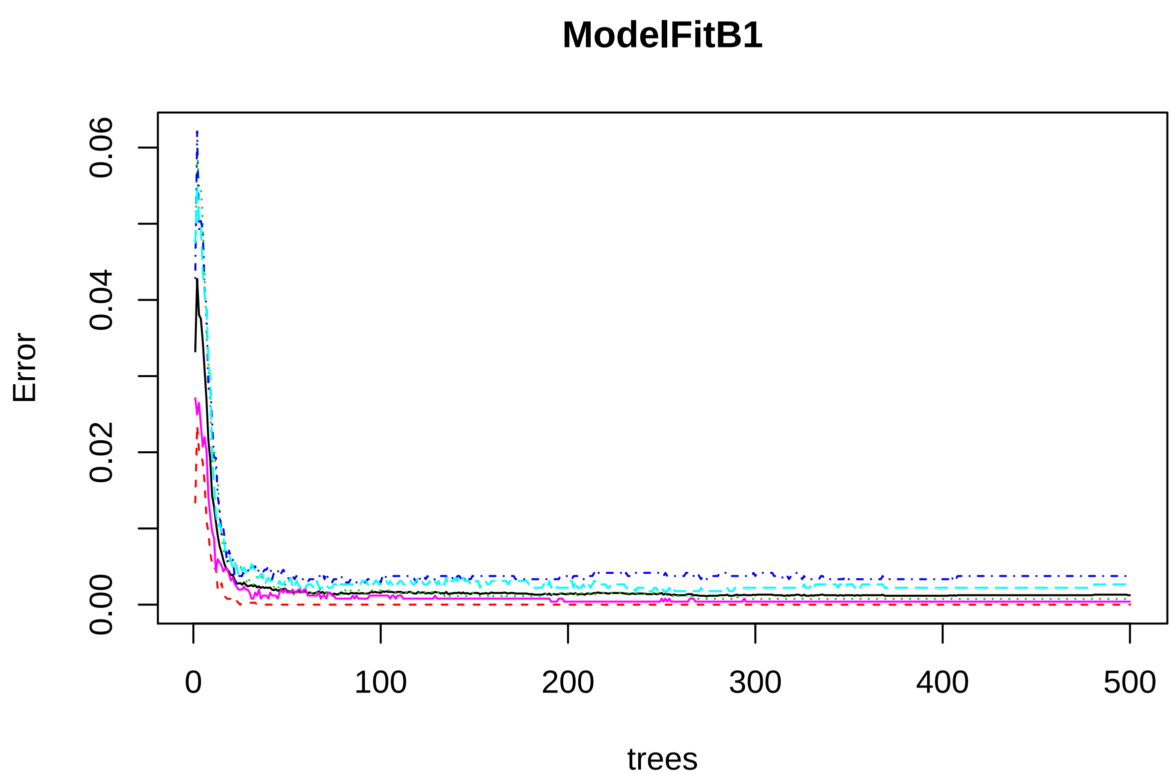

```r
# Prediction with Random Forests

set.seed(22222)
library(randomForest)
ModelFitB1 <- randomForest(classe~., data=myTraining)
predictionB1 <- predict(ModelFitB1, myTest, type="class")
cmrf <- confusionMatrix(predictionB1, myTest$classe)
cmrf
```

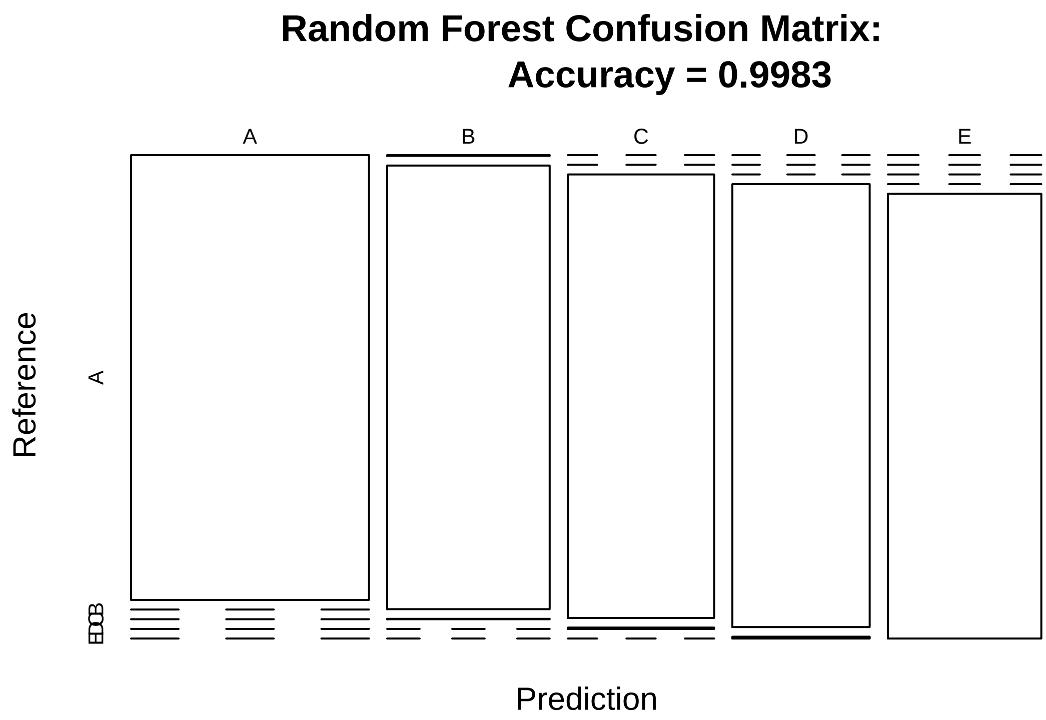
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    0    0    0    0
##          B    2 1139    1    0    0
##          C    0    0 1025    3    0
##          D    0    0    0  961    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   1.0000   0.9990   0.9969   0.9963
## Specificity            1.0000   0.9994   0.9994   0.9992   1.0000
## Pos Pred Value         1.0000   0.9974   0.9971   0.9959   1.0000
## Neg Pred Value         0.9995   1.0000   0.9998   0.9994   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1935   0.1742   0.1633   0.1832
## Detection Prevalence   0.2841   0.1941   0.1747   0.1640   0.1832
## Balanced Accuracy      0.9994   0.9997   0.9992   0.9980   0.9982
```

```r
plot(ModelFitB1)
```




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




```
# Method 3 : Prediction with Generalised Boosted Regression

set.seed(33333)
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

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

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.1.3
```

```
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
```

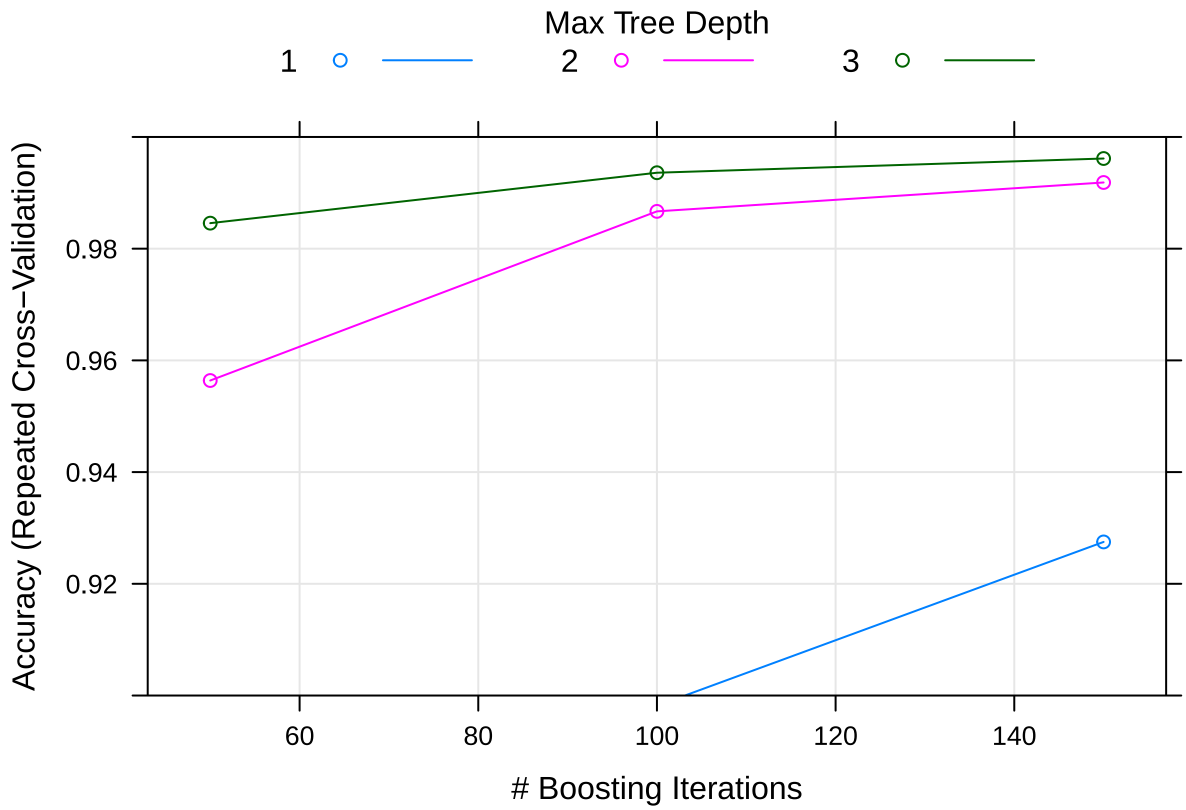
```
## Loaded gbm 2.1.1
## Loading required package: plyr
```

```
gbmFinMod1 <- gbmFit1$finalModel

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    0    0    0    0
##          B    2 1136    1    0    0
##          C    0    1 1015    2    0
##          D    0    2   10  960    8
##          E    0    0    0    2 1074
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9931, 0.9968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9974   0.9893   0.9959   0.9926
## Specificity            1.0000   0.9994   0.9994   0.9959   0.9996
## Pos Pred Value         1.0000   0.9974   0.9971   0.9796   0.9981
## Neg Pred Value         0.9995   0.9994   0.9977   0.9992   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1930   0.1725   0.1631   0.1825
## Detection Prevalence   0.2841   0.1935   0.1730   0.1665   0.1828
## Balanced Accuracy      0.9994   0.9984   0.9943   0.9959   0.9961
```

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



Step 4 : Now that we know that we have high accuracy on the validation data set, lets predict for the test data- Thus here are the results.

```
PredictedResults <- predict(ModelFitB1, TestData, type="class")
PredictedResults
```

```
##  2  3  4 51  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```