Practical Machine Learning Week4 Prediction Assignment

Ajay Aggarwal

May 20, 2019

Overview

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.

The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The data will be used to predict the manner in which they did the exercise. We will apply machine learning algorithm to the 20 test cases available in the test data and generate predictions.

Data

The training data for this project are available here:

pml-training (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

pml-testing (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

Load Data

```
##Load libraries
### install.packages("caret")

library("caret")
```

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

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

```
#Download the data

trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile <- "pml-training.csv"
testFile <- "pml-testing.csv"

getwd()
```

```
## [1] "D:/Work/mystuff/Education/DataScience/MachineLearning"
```

```
if(!file.exists(trainFile)){
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile
= trainFile)
  }

if(!file.exists("pml-testing.csv")){
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile
 = testFile)
  }

#Read the training data and replace empty values by NA
trainData <- read.csv(trainFile, sep=",", header=TRUE, na.strings = c("NA","",'#DIV/0!'))
testData  <- read.csv(testFile, sep=",", header=TRUE, na.strings = c("NA","",'#DIV/0!'))

dim(trainData)
```

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

```
dim(testData)
```

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

Data Cleaning

The training data set contains many columns with NA values or blank values.So we will remove them, because they will not produce any information.

The first seven columns give information about the people who did the test, and also timestamps. We will not take them in our model.

```
trainData <- trainData[,(colSums(is.na(trainData) ) == 0)]
trainData <- trainData[, -c(1:7)]
dim(trainData)
```

```
## [1] 19622    53
```

```
testData <- testData[,(colSums(is.na(testData) ) == 0)]
testData <- testData[, -c(1:7)]
dim(testData)
```

```
## [1] 20 53
```

Partioning the training set for prediction

Partioning Training data set into two data sets, 70% for Training, 30% for Testing. This splitting will help us to compute the out-of-sample errors. The test data will stay as is and will be used later to validate the prodction algorithm on the 20 cases.

```
set.seed(1234)
x <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
trainData70 <- trainData[x, ]
testData30 <- trainData[-x, ]
dim(trainData70)
```

```
## [1] 13737    53
```

```
dim(testData30)
```

```
## [1] 5885   53
```

Identify highly correlated attributes

we use the findCorrelation function to search for highly correlated attributes with a cut off equal to 0.75.

```
### install.packages("corrplot")
### The downloaded binary packages are in C:\Users\aaggarwa.ORADEV\AppData\Local\Temp\RtmpgfDRGk
\downloaded_packages

library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
hc <- findCorrelation(   cor(trainData70[, -53]) , cutoff=0.75)
names(trainData70)[hc]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "accel_dumbbell_z"  "accel_belt_x"
##  [7] "pitch_belt"        "magnet_dumbbell_x" "accel_dumbbell_y"
## [10] "magnet_dumbbell_y" "accel_arm_x"       "accel_dumbbell_x"
## [13] "accel_arm_z"       "magnet_arm_y"      "magnet_belt_z"
## [16] "accel_forearm_y"   "gyros_forearm_y"   "gyros_dumbbell_x"
## [19] "gyros_dumbbell_z"  "gyros_arm_x"
```

Train Model

We will use following algorithms, classification trees,random forests, and generalized boosted model to identify the best model predict the outcome.

1. Classification trees
2. Random forests
3. Generalized Boosted Model

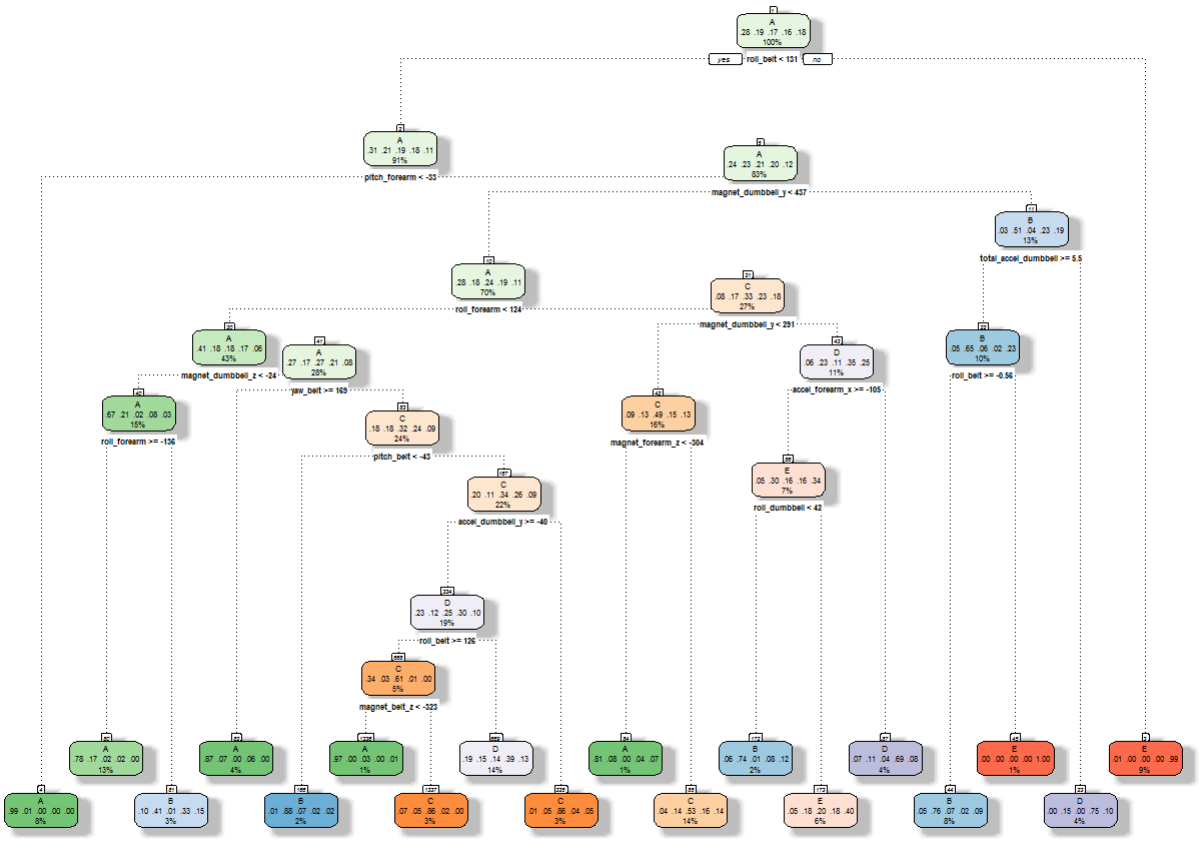
1. Prediction with classification trees

We first apply the Classification tree model, and then we use the fancyRpartPlot() function to plot the classification tree.

```
## install.packages("rpart")

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.
```

```
set.seed(12345)

##dTreeModel <- train(classe~., data=trainData70, method="rpart", trControl=trainControl(method
="cv", number=5))

dTreeModel <- rpart(classe ~ ., data=trainData70, method="class")
fancyRpartPlot(dTreeModel)
```



Rattle 2019-May-21 19:19:23 aaggarwa

Applying classification trees model on test data to determine how well it performed based on the accuracy variable

```
##install.packages('e1071', dependencies=TRUE)
p1 <- predict(dTreeModel, testData30, type = "class")

cmdTree <- confusionMatrix(p1, testData30$"classe")
cmdTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1364  169   24   48   16
##          B   60  581   46   79   74
##          C   52  137  765  129  145
##          D  183  194  125  650  159
##          E   15   58   66   58  688
##
## Overall Statistics
##
##                Accuracy : 0.6879
##                  95% CI : (0.6758, 0.6997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6066
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8148  0.51010   0.7456   0.6743   0.6359
## Specificity            0.9390  0.94543   0.9047   0.8657   0.9590
## Pos Pred Value         0.8415  0.69167   0.6230   0.4958   0.7774
## Neg Pred Value         0.9273  0.88940   0.9440   0.9314   0.9212
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2318  0.09873   0.1300   0.1105   0.1169
## Detection Prevalence   0.2754  0.14274   0.2087   0.2228   0.1504
## Balanced Accuracy      0.8769  0.72776   0.8252   0.7700   0.7974
```

```
#plot(dTreeModel,main="classification trees model error by number of trees")

acrate <- round(cmdTree$overall['Accuracy'], 4)
outofSampleError <- 1-acrate
```

Accuracy rate of the model is 0.6879 out-of-sample-error is about 0.3121 which is considerable.

2. Prediction using random forest

Here we will predict the sample error using random forest ML algorithms and using confusion Matrix to test results.

```
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
```

```
##Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same ##training set, with the goal of reducing the variance.This comes at the expense of  a small increase in the bias and some loss of interpretability, but generally greatly boosts th e performance in the final model.

##RFs train each tree independently, using a random sample of the data. This randomness helps to make the model more robust than a single decision tree, and less likely to overfit on the traini ng data

##rfModel <- train(classe~., data=trainData70, method="rf", trControl=trControl, verbose=FALSE)
rfModel <- randomForest(classe ~. , data=trainData70)
predictRFModel <- predict(rfModel, testData30, type = "class")
rfCMD <- confusionMatrix(predictRFModel, testData30$classe)
rfCMD
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1132    5    0    0
##          C    0    1 1020    4    0
##          D    0    0    1  959    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9939   0.9942   0.9948   0.9991
## Specificity            0.9986   0.9989   0.9990   0.9996   0.9998
## Pos Pred Value         0.9964   0.9956   0.9951   0.9979   0.9991
## Neg Pred Value         1.0000   0.9985   0.9988   0.9990   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1733   0.1630   0.1837
## Detection Prevalence   0.2855   0.1932   0.1742   0.1633   0.1839
## Balanced Accuracy      0.9993   0.9964   0.9966   0.9972   0.9994
```

```
rfCMD$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1132    5    0    0
##          C    0    1 1020    4    0
##          D    0    0    1  959    1
##          E    0    0    0    1 1081
```

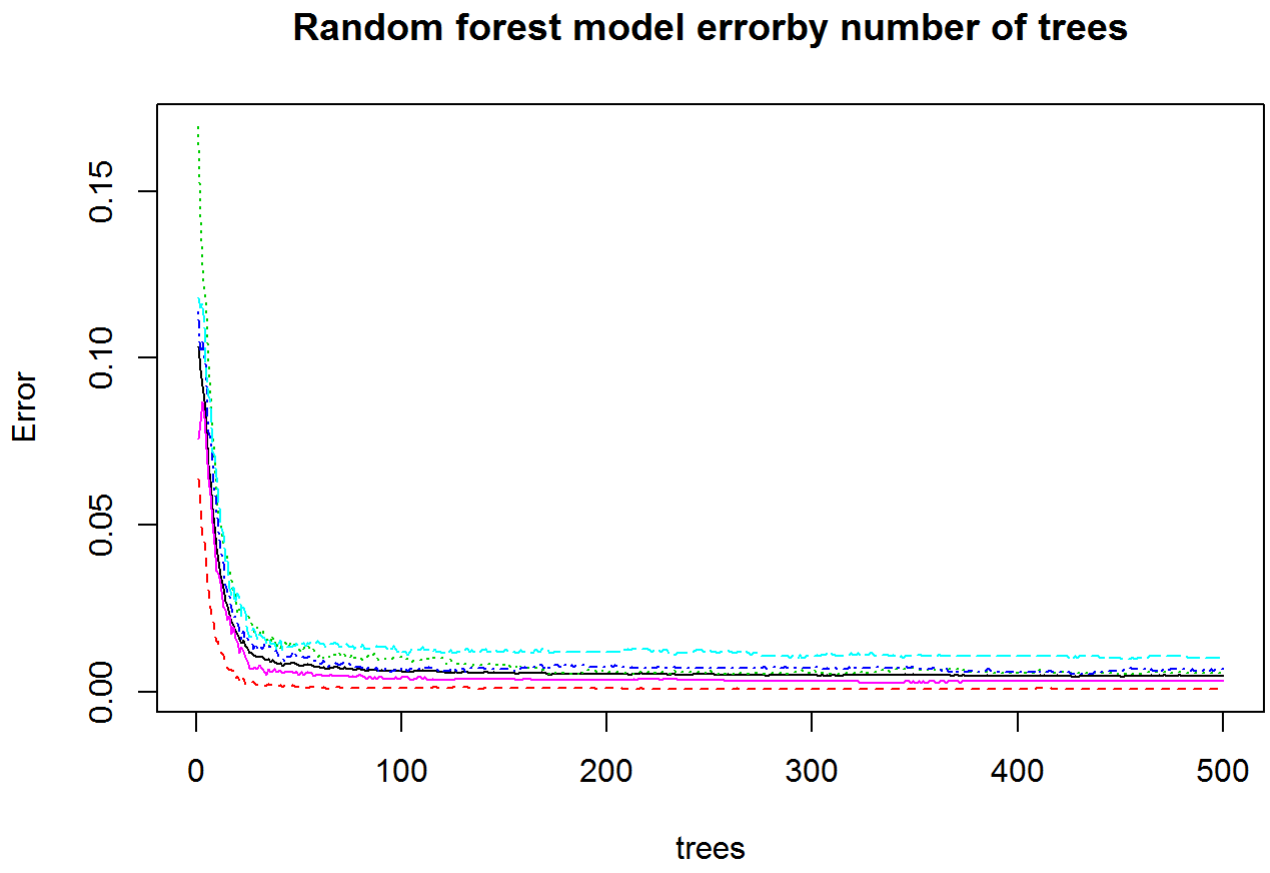
```
rfAcrate <- rfCMD$overall[1]
rfoutofSampleError2 <- 1-rfAcrate
```

Random Forests yielded better Results, as expected! Accuracy rate of the model is 0.6879 out-of-sample-error is about 0.3121 which is considerable.

The accuracy rate using the random forest is very high: Accuracy : 0.9967715 and therefore the out-of-sample-error is equal to 0.0032285. But it might be due to overfitting.

Let’s plot the model

```
## This plot shows each of the principal components in order from most important to least import
ant.
##varImpPlot(rfModel$finalModel, sort = TRUE, type = 1, pch = 19, col = 1, cex = 0.6, main = "Im
portance of the ##Individual Principal Components")
##modFitrf <- train(classe ~., method="rf", data=training, trControl=trainControl(method='cv'),
 number=5, ##allowParallel=TRUE, importance=TRUE )
plot(rfModel,main="Random forest model errorby number of trees")
```




```
##names(rfModel)
```

3. Prediction with Generalized Boosted Regression Models

```
###install.packages("gbm")

library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
##  GBT build trees one at a time, where each new tree helps to correct errors made by previousl
y trained tree.

set.seed(12345)
GBRMControl <- trainControl(method = "repeatedcv", number = 5, repeats = 1)

GBM_Model  <- train(classe ~ ., data=trainData70, method = "gbm", trControl = GBRMControl, verbo
se = FALSE)
GBM_Model$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

```
###Validate the GBM model and
pGBM <- predict(GBM_Model, newdata=testData30)
cmGBM <- confusionMatrix(pGBM, testData30$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1655   45    0    1    1
##          B   10 1069   27    4   20
##          C    4   24  985   24    9
##          D    4    1   11  924   16
##          E    1    0    3   11 1036
##
## Overall Statistics
##
##                Accuracy : 0.9633
##                  95% CI : (0.9582, 0.968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9535
##
##  Mcnemar's Test P-Value : 6.099e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9886   0.9385   0.9600   0.9585   0.9575
## Specificity            0.9888   0.9871   0.9874   0.9935   0.9969
## Pos Pred Value         0.9724   0.9460   0.9417   0.9665   0.9857
## Neg Pred Value         0.9955   0.9853   0.9915   0.9919   0.9905
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2812   0.1816   0.1674   0.1570   0.1760
## Detection Prevalence   0.2892   0.1920   0.1777   0.1624   0.1786
## Balanced Accuracy      0.9887   0.9628   0.9737   0.9760   0.9772
```

```
gbmAcRate <- cmGBM$overall[1]
```

Test cases:Applying the best model to the 20 test cases validation data

Comparing the accuracy rate values of the three models: 1. Generalized Boosted Regression Models 0.9632965 2. Random Forests Models 0.9967715 3. Classification Tree Models 0.6879

By comparing the accuracy rate values of the three models, it is clear the the ‘Random Forest’ model is the winner. So will use it on the validation data

```
Results <- predict(rfModel, newdata=testData)
Results
```

```
##  1  2  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
```