Predicting the Quantified self

Patrick Osoro

Saturday, September 17, 2016

Contents

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 â 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.

Project Goal

In this project, the goal was to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which the participants did the exercise. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.It is the classe varaible in the dataset, and one can use any of the other variables to predict classe. 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).

Synopsis

The results obtained by using random forest modeling technique were highly accurate on the testing set achieving accuracy of 100% and kappa of 1, while those resulting from devcision tree model had accuracy of 76% and kappa of 0.7. Based on these results the Random forest classification technique works better than desicion tree in this case in predicting the manner in which the participants did the exercise.

Set environment variables

```
    knitr::opts_chunk$set(fig.width=12, fig.height=8, fig.path='Figs/', Sys.setlocale(category = "LC_ALL
    invisible(dev.off())
```

```
    library(knitr)
```

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

```
    opts_chunk$set(tidy.opts=list(width.cutoff=80))
    devtools::install_github("rstudio/rmarkdown")
```

```
## Skipping install for github remote, the SHA1 (81c20927) has not changed since last install.
##   Use `force = TRUE` to force installation
```

Load necessary R libraries

```
library(caret)
```

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

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

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

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
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.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

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

```
library(ggplot2)
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.
```

```
library(Hmisc)
```

```
## Warning: package 'Hmisc' 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: Formula
## Warning: package 'Formula' was built under R version 3.1.3
##
## Attaching package: 'Hmisc'
## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.1.3
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize
```

```
library(AppliedPredictiveModeling)
```

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

Data Processing

Data sources

The training data for this project are available here:

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

The test data are available here:

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

Load the training and testing data sets

Import the data treating empty values as NA.

```
set.seed(10000)
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainingSet <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testingSet <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

#Explore features and data values

dim(trainingSet)
```

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

```
dim(testingSet)
```

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

```
#validate that training set and test set are from same sets i.e if similar in terms of column names
#colnames(trainingSet)
#colnames(testingSet)
colnames_train <- colnames(trainingSet)
colnames_test <- colnames(testingSet)

all.equal(colnames_train[1:length(colnames_train)-1], colnames_test[1:length(colnames_train)-1])
```

```
## [1] TRUE
```

```
#The datasets have same columns
#head(trainingSet)
#head(testingSet)
```

The training set has 19622 observations and 160 variables while the testing sets has 20 observations and 160 variables.

The training sets and testing set have same column names.

Data Cleaning

There were 9 variables that consist of only 0 or NA, namely, kurtosis_yaw_belt, kurtosis_yaw_dumbbell, kurtosis_yaw_forearm, skewness_yaw_belt, skewness_yaw_dumbbell, skewness_yaw_forearm, amplitude_yaw_belt, amplitude_yaw_dumbbell, and amplitude_yaw_forearm. We know those variables will not help in terms of classification, so they were removed.

```
#Delete columns with missing values
trainingSet <- trainingSet[, colSums(is.na(trainingSet)) == 0]
testingSet <- testingSet[, colSums(is.na(testingSet)) == 0]


#Remove the first seven predictors since these variables have little predicting power for the outcome v

NewtrainingSet <- trainingSet[, -c(1:7)]
NewtestingSet <- testingSet[, -c(1:7)]
dim(NewtrainingSet); dim(NewtestingSet)
```

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

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

The cleaned Newtrainingset has 53 variables.

The cleaned NewtestingSet has 53 variables.

Split training set to create a validation set of 40% of the training data set

Split the cleaned training set into a pure training data set (60%) and a validation data set (40%). The validation data set will be used to conduct cross validation.

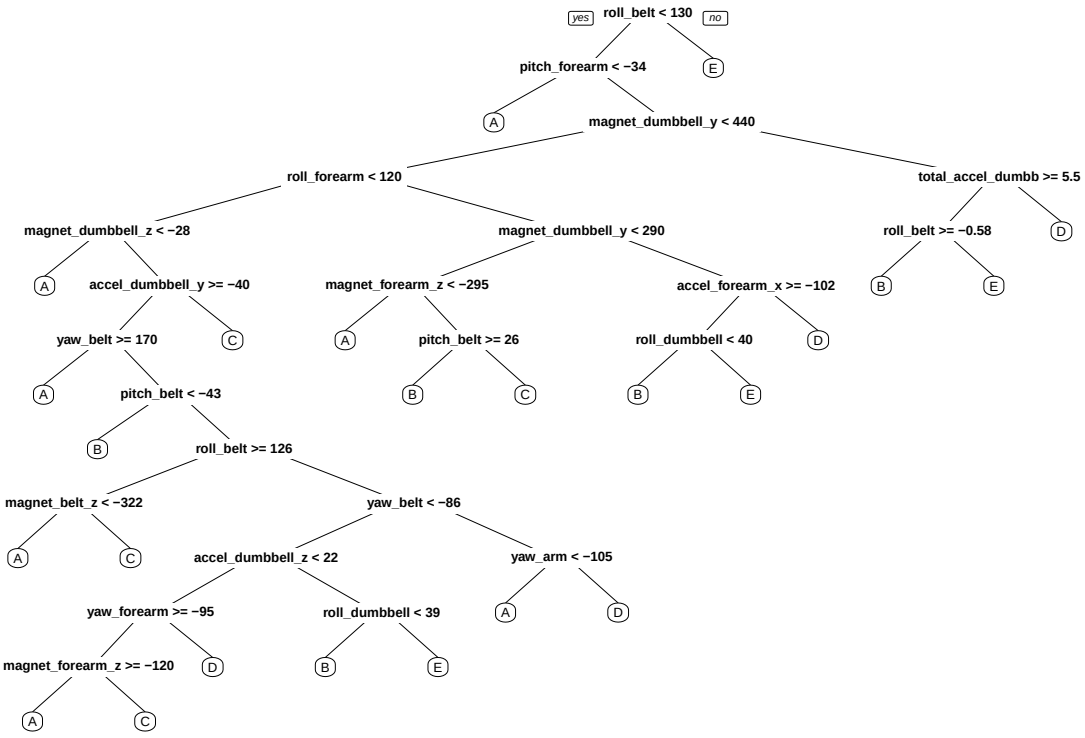
```
set.seed(10000)
inTrain <- createDataPartition(NewtrainingSet$classe, p=0.6, list=FALSE)

NewtrainingSet <- NewtrainingSet[inTrain, ]
validationSet <- NewtrainingSet[-inTrain, ]

dim(NewtrainingSet); dim(validationSet)
```

```
## [1] 11776    53
```

```
## [1] 4704   53
```

Tree Vizualization

```
treeModel <- rpart(classe ~ ., data=NewtrainingSet, method="class")
prp(treeModel) # plot of Tree Model
```



Prediction Algorithms

Predicting with Decision Trees

One can use classification trees and random forests to predict the outcome. Requires library(e1071)

```
set.seed(10000)
library(e1071)
```

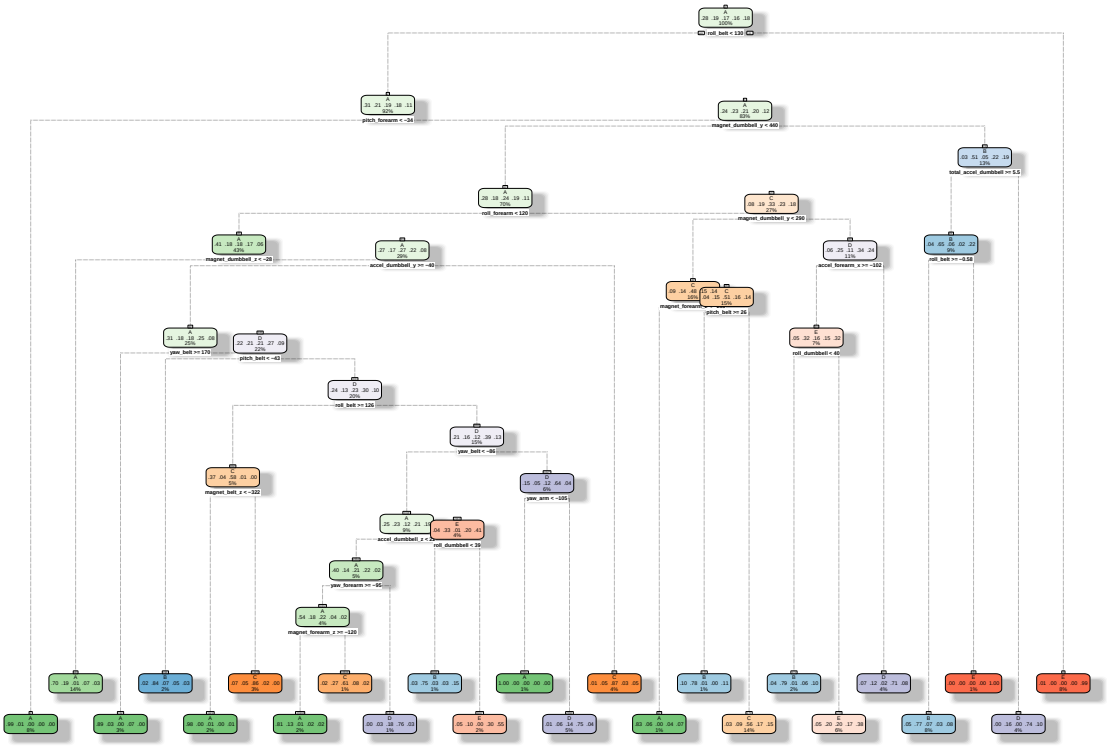
```
## Warning: package 'e1071' was built under R version 3.1.3
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:Hmisc':
##
##     impute
```

```
modelFit1 <- rpart(classe ~ ., data=NewtrainingSet, method="class")
fancyRpartPlot(modelFit1)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2016−Sep−21 00:16:12 Administrator

```
#---------------------------------------------------------------------
prediction1 <- predict(modelFit1, validationSet, type = "class")
```

Using confusion Matrix to test results:

```
    confusionMatrix(prediction1, validationSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1263  165   10   56   27
##          B   33  517   34   25   58
##          C   31   74  662  129  104
##          D   18   61   49  502   40
##          E   27   60   39   86  634
##
## Overall Statistics
##
##                Accuracy : 0.7606
##                  95% CI : (0.7482, 0.7728)
##     No Information Rate : 0.2917
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6955
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9206   0.5895   0.8338   0.6291   0.7346
## Specificity            0.9226   0.9608   0.9136   0.9570   0.9448
## Pos Pred Value         0.8304   0.7751   0.6620   0.7493   0.7494
## Neg Pred Value         0.9658   0.9108   0.9644   0.9266   0.9406
## Prevalence             0.2917   0.1864   0.1688   0.1696   0.1835
## Detection Rate         0.2685   0.1099   0.1407   0.1067   0.1348
## Detection Prevalence   0.3233   0.1418   0.2126   0.1424   0.1798
## Balanced Accuracy      0.9216   0.7752   0.8737   0.7930   0.8397
```

Predicting with Random forests

```
modelFit2 <- randomForest(classe ~. , data=NewtrainingSet)
```

Predicting in-sample error:

```
prediction2 <- predict(modelFit2, validationSet, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(prediction2, validationSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1372    0    0    0    0
##          B    0  877    0    0    0
##          C    0    0  794    0    0
##          D    0    0    0  798    0
##          E    0    0    0    0  863
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9992, 1)
##     No Information Rate : 0.2917
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2917   0.1864   0.1688   0.1696   0.1835
## Detection Rate         0.2917   0.1864   0.1688   0.1696   0.1835
## Detection Prevalence   0.2917   0.1864   0.1688   0.1696   0.1835
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
#Summary results

#accuracy
accuracy <- postResample(prediction2, validationSet$classe)
accuracy
```

```
## Accuracy    Kappa
##        1        1
```

```
#sample error
error <- 1 - as.numeric(confusionMatrix(validationSet$classe, prediction2)$overall[1])
error
```

```
## [1] 0
```

Accuracy of this prediction model is 100% using the validation set

Out of Sample Error is 0%.

The out of sample error is just the error rate that we get when we apply the classification model on a new data set.

Predicting using original Test Data

Apply the model to the original testing data set downloaded from the data source. Generate "problem_id_x.txt" Files for the assignments. These generated individual files are stored in working directory.

```
Testprediction <- predict(modelFit2, NewtestingSet)
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(Testprediction)
```

Loaded these files as per Problem_id number into Project Assignment and obtained the correct results for all 20 cases.

Conclusion

The results obtained by using random forest technique were highly accurate on the testing set achieving accuracy of 100% and kappa of 1, while those resulting from devcision tree model had accuracy of 76% and kappa of 0.7. Based on these results the Random forest classification technique works better than desicion tree in this case.