Coursera: Practical Machine Learning Prediction Assignment

Anil Kumar

December 23, 2015

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).

Data

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).
2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Reproducibility

package required such as caret, rattle, rpart, rpart.plot, randomForest

```
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(rattle)
```

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

```
## Warning: Failed to load RGtk2 dynamic library, attempting to install it.
```

```
## Please install GTK+ from http://r.research.att.com/libs/GTK_2.24.17-X11.pkg
## If the package still does not load, please ensure that GTK+ is installed and that it is on your PATH
## IN ANY CASE, RESTART R BEFORE TRYING TO LOAD THE PACKAGE AGAIN
## Rattle: A free graphical interface for data mining with R.
## Version 4.0.5 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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

```
## Warning: package 'rpart.plot' 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.
```

set seed

```
set.seed(12345)
```

**get data from URL and clean data, and also getrid off irrelevant variables

```r
# get training data from url
train_data <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
# get testing data from url
test_data <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Lets store the data in variables
training <- read.csv(url(train_data), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(test_data), na.strings=c("NA","#DIV/0!",""))

# check dimension of these variables
dim(training); dim(testing)
# Delete columns with missing values
training1 <- training[,colSums(is.na(training)) == 0]
testing1 <- testing[,colSums(is.na(testing)) == 0]
dim(training1); dim(testing1)
# Deleting irrelevant variables (user_name, raw_timestamp_part_1, raw_timestamp_part_, 2 cvtd_timestamp
training1 <- training1[,-c(1:7)]
testing1 <- testing1[,-c(1:7)]

dim(training1); dim(testing1)
```

Partitioning Data into Training and Testing Part

There are 19622 observations so we need 75% training and 25% of data for testing

```r
inTrain <- createDataPartition(y=training1$classe, p=0.75, list=FALSE)
myTraining <- training1[inTrain, ]
myTesting <- training1[-inTrain, ]
dim(myTraining)
dim(myTesting)
```

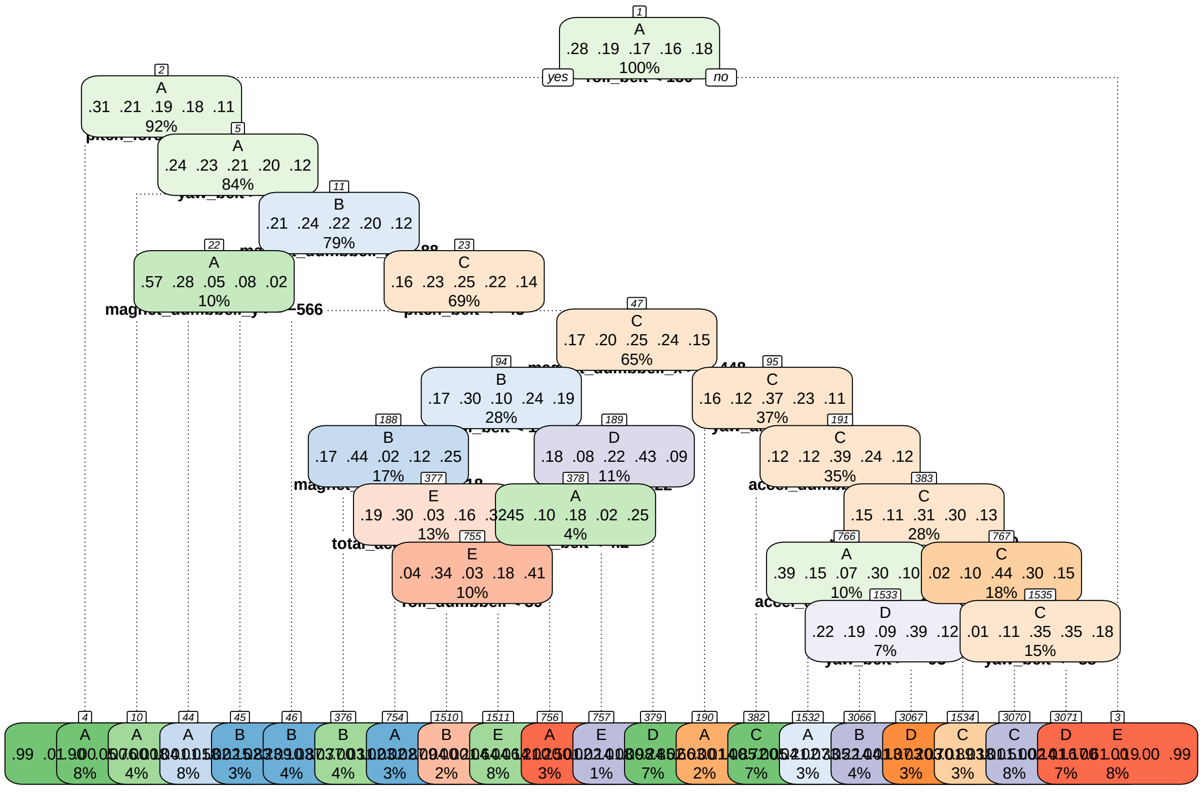
1st Prediction Model - Decision Tree

```r
modFit1 <- rpart(classe ~ ., data=myTraining, method="class")

# Predicting
prediction1 <- predict(modFit1, myTesting, type = "class")

# Plotting
fancyRpartPlot(modFit1, cex=.5,under.cex=1,shadow.offset=0)
```

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


Rattle 2015−Dec−24 13:04:41 anilkumar

```r
# Testing results on myTesting data set
confusionMatrix(prediction1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1260  156   33   40   23
##          B   52  555   73   52   52
##          C   24  136  575   83   95
##          D   40   33  150  513   89
##          E   19   69   24  116  642
##
## Overall Statistics
##
##                Accuracy : 0.7229
##                  95% CI : (0.7101, 0.7354)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6486
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9032   0.5848   0.6725   0.6381   0.7125
## Specificity            0.9282   0.9421   0.9165   0.9239   0.9430
## Pos Pred Value         0.8333   0.7079   0.6298   0.6218   0.7379
## Neg Pred Value         0.9602   0.9044   0.9298   0.9287   0.9358
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2569   0.1132   0.1173   0.1046   0.1309
## Detection Prevalence   0.3083   0.1599   0.1862   0.1682   0.1774
## Balanced Accuracy      0.9157   0.7635   0.7945   0.7810   0.8278
```

2nd Prediction Model - Random Forest

```r
modFit2 <- randomForest(classe ~ ., data=myTraining, method="class")

prediction2 <- predict(modFit2, myTesting, type="class")

confusionMatrix(prediction2, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##            A 1395    6    0    0    0
##            B    0  939    2    0    0
##            C    0    4  852    7    1
##            D    0    0    1  797    4
##            E    0    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9965   0.9913   0.9945
## Specificity            0.9983   0.9995   0.9970   0.9988   1.0000
## Pos Pred Value         0.9957   0.9979   0.9861   0.9938   1.0000
## Neg Pred Value         1.0000   0.9975   0.9993   0.9983   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1737   0.1625   0.1827
## Detection Prevalence   0.2857   0.1919   0.1762   0.1635   0.1827
## Balanced Accuracy      0.9991   0.9945   0.9968   0.9950   0.9972
```

CONCLUSION

Its very clear that Random Forest method shows better prediction with very high accuracy of 0.99 where as Decision Tree model has an accuracy about 0.72.

Programming assignment for automated grading

Random forest machine algorithm to 20 test cases

```r
prediction <- predict(modFit2, testing1, type="class")
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(prediction)
```