Coursera_Regression_Project

Zaid Muhsin

December 13, 2018

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

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Project Requirement

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.

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 :-). 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.

Analysis Steps

1. Load the data and needed packages.
2. Clean data (get rid of un necessary variables to the prediction models, those with mostly NA values and near zero variance).
3. Apply random forest and Decision tree models to predict the classes.
4. Choose the model with highest accuracy.
5. Apply the model on the test data (20 observations) to answer the quiz.

Packages, Libraries and Seed

```r
library(caret)
```

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

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

```r
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:ggplot2':
##
##     margin
```

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

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```r
library(randomForest)
#setting seed
set.seed(12345)
```

Getting and cleaning data

```
#Downloaded the training and test data sets from links belwod, and then load them into dat
a frames.
setwd("C://data_science/regression/project")
training<-read.csv("pml-training.csv")
testing<-read.csv("pml-testing.csv")

# create a partition with the training dataset (70%training and 30 testing)
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
dim(TrainSet)
```

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

```
dim(TestSet)
```

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

Cleaning Data

In this step, we will clean the dataset and get rid of observations with missing values as well as some meaningless variables.

```
#removing those with near zero variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]

# remove variables that are mostly NA
AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
table(AllNA)
```

```
## AllNA
## FALSE  TRUE
##    59    47
```

```
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]

# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

```
## [1] 13737    54
```

```
#Now we are eliminated un-neccessary variables, down from 160 to 54 variables
```

Modelling

Method 01: Random forest

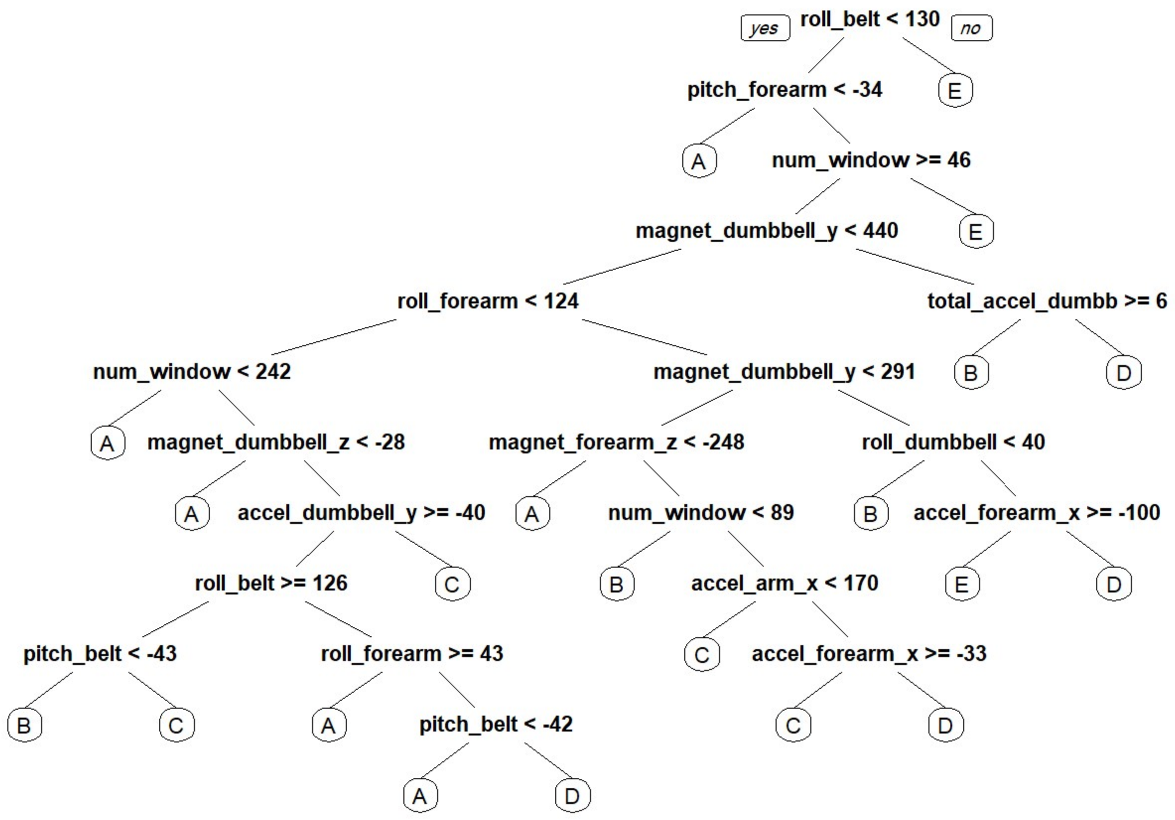
```
modelRF <- randomForest(classe ~ .,   TrainSet, do.trace=F)
predictRF <- predict(modelRF, TestSet)
confusionMatrix(TestSet$classe, predictRF)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    6 1132    1    0    0
##          C    0    5 1021    0    0
##          D    0    0   12  952    0
##          E    0    0    0    4 1078
## 
## Overall Statistics
## 
##                Accuracy : 0.9952
##                  95% CI : (0.9931, 0.9968)
##     No Information Rate : 0.2855
##     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            0.9964   0.9956   0.9874   0.9958   1.0000
## Specificity            1.0000   0.9985   0.9990   0.9976   0.9992
## Pos Pred Value         1.0000   0.9939   0.9951   0.9876   0.9963
## Neg Pred Value         0.9986   0.9989   0.9973   0.9992   1.0000
## Prevalence             0.2855   0.1932   0.1757   0.1624   0.1832
## Detection Rate         0.2845   0.1924   0.1735   0.1618   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9982   0.9971   0.9932   0.9967   0.9996
```

This method has a prediction accuracy of 99.5 %.

Method 02: Decision tree

```
modelTree <- rpart(classe ~ ., data = TrainSet, method = "class")
prp(modelTree)
```



```
#Now, we estimate the performance of the model on the TestSet data set.
predictTree <- predict(modelTree, TestSet, type = "class")
confusionMatrix(TestSet$classe, predictTree)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1530   35   17   39   53
##          B  269  575   73  146   76
##          C   51   31  743  130   71
##          D   79   25   68  702   90
##          E   16   68   84  128  786
## 
## Overall Statistics
##                                           
##                Accuracy : 0.7368          
##                  95% CI : (0.7253, 0.748) 
##     No Information Rate : 0.3305          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.6656          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7866  0.78338   0.7543   0.6131   0.7305
## Specificity            0.9635  0.89051   0.9422   0.9447   0.9384
## Pos Pred Value         0.9140  0.50483   0.7242   0.7282   0.7264
## Neg Pred Value         0.9014  0.96650   0.9502   0.9100   0.9396
## Prevalence             0.3305  0.12472   0.1674   0.1946   0.1828
## Detection Rate         0.2600  0.09771   0.1263   0.1193   0.1336
## Detection Prevalence   0.2845  0.19354   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8750  0.83694   0.8483   0.7789   0.8345
```

```
#As can be seen this model has accuracy of 74%, less than the accuracy of the random fores
t.
```

Conclusion

Random forest model is the most accurate model and will be applied on the final testing dataset to predict the classes for the (20 observations).

```
predictRF <- predict(modelRF, testing)
predictRF
```

```
##  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
```