Coursera PML Assignment: Predicting how well an exercise is performed

Najma Rajah

30/03/2019

Summary

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. People regularly quantify how much of a particular activity they do, but they rarely quantify how well they do it. This project examined data from accelerometers on the belt, forearm, arm, and dumbell of six young men who conducted a weight-lifting exercise to predict whether they performed the exercise correctly. The analysis found that the random forest model was the most accurate of the four models considered.

Exploratory data analysis

The first step involved loading the data.

```
fileUrl1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv?accessType=DOWNLOAD"
download.file(fileUrl1, destfile = "pml-training.csv", method="curl")
dateDownloaded<-date()

fileUrl2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv?accessType=DOWNLOAD"
download.file(fileUrl2, destfile = "pml-testing.csv", method="curl")
dateDownloaded<-date()

traindata<-read.csv("pml-training.csv", header=TRUE,sep=",", stringsAsFactors
= FALSE)
testdata<-read.csv("pml-testing.csv", header=TRUE,sep=",", stringsAsFactors =
FALSE)
```

The training data set comprises 19,622 observations and 160 variables. As the goal of this project is to predict whether each particant performed the exercise correctly, it is useful to tabulate the variable classe. The classe variable had five levels - A to E. A indicated that the exercise was performed correctly and categories B to E indicated that there was some form of mistake.

As well as the classe variable, the training data set also included data from accelerometers on the belt, forearm, and dumbell of the six participants. However variables had many missing values. So it was necessary to remove variables which have many NAs or missing values, both in the training and in the test data sets. We also repeated this process for the test data set.

```
table(traindata$classe)

##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

traindata<-select(traindata,
(8:11),(37:49),(60:68),(84:86),102,(113:124),140,(151:159),160)
testdata<-select(testdata,
(8:11),(37:49),(60:68),(84:86),102,(113:124),140,(151:159))
```

As a result of this data cleaning, the number of variables was reduced to 53.

```
dim(traindata)

## [1] 19622    53
```

Creating of a testing data set

In order to assess the performance of alternative models, it is necessary to split the original training set (comprising 16922 observations) into two - a training data set and a testing data set which provides an opportunity to understand how each model performs when it is required to predict out of sample.

```
r  library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

r  inTrain<-createDataPartition(y=traindata$classe, p=0.75, list=FALSE)
training<-traindata[inTrain,]  testing<-traindata[-inTrain,]
table(training$classe)

##    ##    A    B    C    D    E   ## 4185 2848 2567 2412 2706
```

Development of four models based on the training data set

The analysis looked at four approaches which are suitable for predicting categorical data - Linear Discriminant Analysis (LDA), tree based methods, Boosting and Random Forests. For each model, we (a) estimated the parameters for each model based on the training data set; (b) estimated predictions based on each model using the testing data set; and (c) assessed the performance of each model based on the output of the confusion Matrix.

Of particular interest was the accuracy rate for each model as this shows the proportion of correct predictions when the model predicts out of sample. Note that a seed was set for the tree based, boosting and random forests to enable reproducibility.

Linear Discriminant Analysis

```
library(caret)
set.seed(109)
model1<-train(classe~., data=training, method="lda")
pmodel1<-predict(model1, newdata=testing)
testing$classe<-as.factor(testing$classe)
confusionMatrix(pmodel1, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1179  122   99   51   33
##          B   28  634   87   37  148
##          C   85  104  552   95   71
##          D   97   30   91  598   99
##          E    6   59   26   23  550
##
## Overall Statistics
##
##                Accuracy : 0.7164
##                  95% CI : (0.7035, 0.7289)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6407
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8452   0.6681   0.6456   0.7438   0.6104
## Specificity            0.9131   0.9241   0.9123   0.9227   0.9715
## Pos Pred Value         0.7945   0.6788   0.6086   0.6536   0.8283
## Neg Pred Value         0.9368   0.9207   0.9242   0.9484   0.9172
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2404   0.1293   0.1126   0.1219   0.1122
## Detection Prevalence   0.3026   0.1905   0.1850   0.1866   0.1354
## Balanced Accuracy      0.8791   0.7961   0.7790   0.8332   0.7910
```

The LDA model had an accuracy of around 0.70.

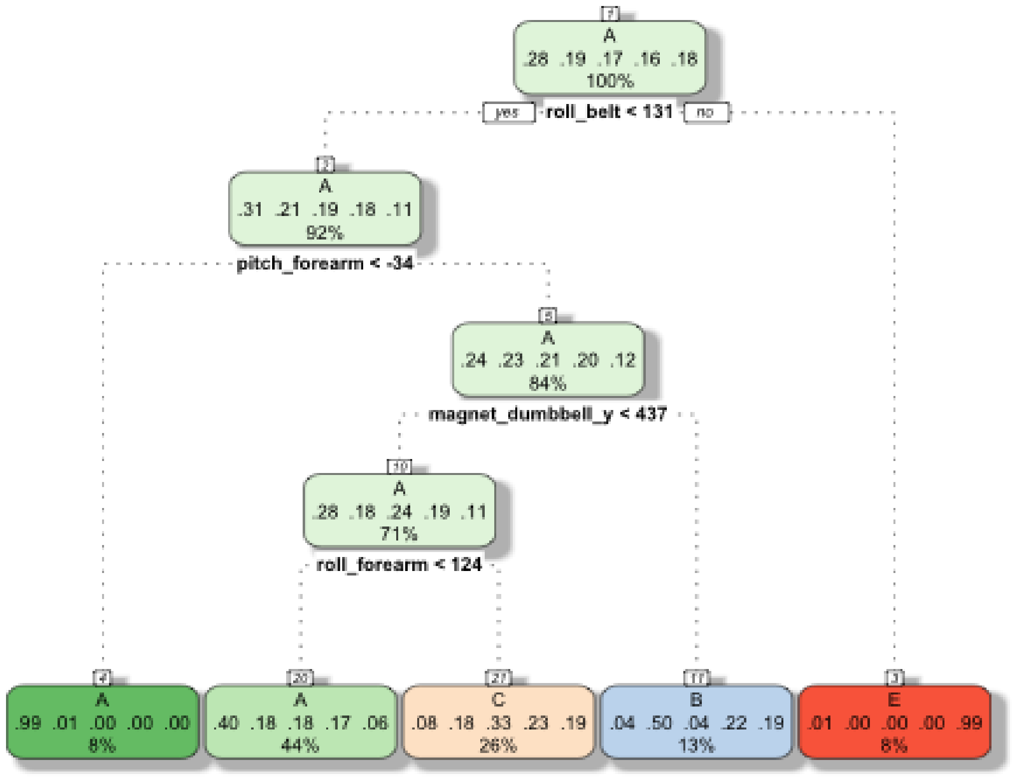
Tree-based model

```r
library(caret)
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(123)
model2<-train(classe~., method="rpart", data=training)
```

The chart shows that the model performed relatively poorly within sample, predicting that 52% of observations in the training data set were in category A when the actual percentage in class A was 28%. The model therefore also performed badly out of sample (in the testing set), with an accuracy of around 0.49.



Rattle 2019-Mar-31 22:45:39 geoffreyknowles

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1274  369  392  340  123
##          B   23  337   26  151  131
##          C   92  243  437  313  225
##          D    0    0    0    0    0
##          E    6    0    0    0  422
##
## Overall Statistics
##
##                Accuracy : 0.5037
##                  95% CI : (0.4896, 0.5178)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3522
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9133  0.35511  0.51111   0.0000  0.46837
## Specificity            0.6512  0.91631  0.78439   1.0000  0.99850
## Pos Pred Value         0.5100  0.50449  0.33359      NaN  0.98598
## Neg Pred Value         0.9497  0.85552  0.88370   0.8361  0.89298
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2598  0.06872  0.08911   0.0000  0.08605
## Detection Prevalence   0.5094  0.13622  0.26713   0.0000  0.08728
## Balanced Accuracy      0.7822  0.63571  0.64775   0.5000  0.73343
```

Boosting

Boosting is an approach which improves the predictions from a decision tree. It works by fitting trees on a sequential basis to the errors and by placing greater weight on large errors, it generates more accurate predictions than simple decision tree-based models.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1370   26    0    0    2
##          B   17  890   26    2   12
##          C    3   31  820   30    3
##          D    3    0    8  768   15
##          E    2    2    1    4  869
##
## Overall Statistics
##
##                Accuracy : 0.9619
##                  95% CI : (0.9561, 0.9671)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9518
##  Mcnemar's Test P-Value : 4.507e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9821   0.9378   0.9591   0.9552   0.9645
## Specificity            0.9920   0.9856   0.9835   0.9937   0.9978
## Pos Pred Value         0.9800   0.9398   0.9245   0.9673   0.9897
## Neg Pred Value         0.9929   0.9851   0.9913   0.9912   0.9921
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2794   0.1815   0.1672   0.1566   0.1772
## Detection Prevalence   0.2851   0.1931   0.1809   0.1619   0.1790
## Balanced Accuracy      0.9870   0.9617   0.9713   0.9744   0.9811
```

The accuracy for this model was around 0.96.

Random Forest

Random Forest models build on the basic principles of decision trees but use an approach which leads to improved accuracy. It only considers a subset of the predictors at each split, and places less weight on strong predictors, to avoid the problem of highly correlated trees. This makes the resulting trees less variable and more reliable.

A disadvantage of random forests is that they can be slow to estimate. For this reason, the parallel package was used in conjuntion with the caret package. For further discussion see https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md for further discussion.

```
## Loading required package: foreach

## Loading required package: iterators

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    4    0    0    0
##          B    0  941    4    0    0
##          C    0    4  850   17    0
##          D    0    0    1  786    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9933
##                  95% CI : (0.9906, 0.9954)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9916   0.9942   0.9776   0.9978
## Specificity            0.9989   0.9990   0.9948   0.9993   0.9998
## Pos Pred Value         0.9971   0.9958   0.9759   0.9962   0.9989
## Neg Pred Value         1.0000   0.9980   0.9988   0.9956   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1919   0.1733   0.1603   0.1833
## Detection Prevalence   0.2853   0.1927   0.1776   0.1609   0.1835
## Balanced Accuracy      0.9994   0.9953   0.9945   0.9884   0.9988
```

The random forest model had the highest accuracy at around 0.99.

Prediction using the test data

For this reason, the random forest model was used to predict the variable classe in the test data. It predicted correctly for 100% of the observations in the test data set.