Practical Machine Learning Project

Mike

17 JÃ¤nner 2020

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.

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.

Loading necessary packages

```r
rm(list = ls(all.names = TRUE))
#install.packages("data.table")
#install.packages("rpart.plot")
#install.packages("rpart")
#install.packages("rattle")
#install.packages("e1071")
#install.packages("randomForest")
library(data.table)
library(tidyr)
library(caret)
library(rpart)
library(rattle)
library(e1071)
library(rpart.plot)
library(randomForest)
library(dplyr)
```

Getting and cleaning the Data

The first step is to download the data, load it into R and prepare it for the modeling process.

```r
train.data <- data.frame(fread("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
test.data <- data.frame(fread("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))

# Drop the first 7 columns as they're unnecessary for predicting.
train.data.clean <- train.data[,8:length(colnames(train.data))]
test.data.clean <- test.data[,8:length(colnames(test.data))]
train.data.clean <- train.data.clean[, colSums(is.na(train.data.clean)) == 0]
test.data.clean <- test.data.clean[, colSums(is.na(test.data.clean)) == 0]

# Splitting up the whole training data set into 75 % subTraining and 25 % subTesting
sub.samples <- createDataPartition(y=train.data.clean$classe, p=0.75, list=FALSE)
sub.train.data.clean <- train.data.clean[createDataPartition(y=train.data.clean$classe, p=0.75, list=FA
sub.test.data.clean <- train.data.clean[-(createDataPartition(y=train.data.clean$classe, p=0.75, list=F
```

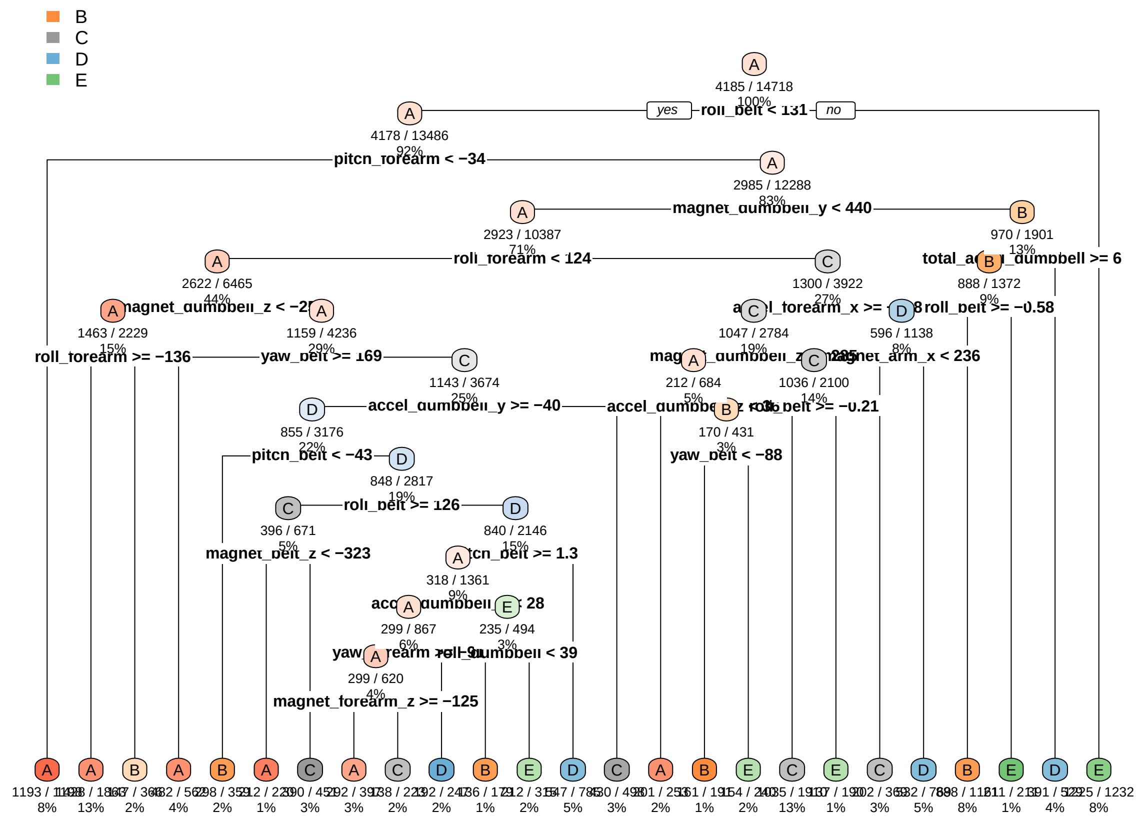
First prediction model

A decision tree is developed as the first prediction model. The model is trained using the training data set. A prediction model is derived from the data set generated in this way and the actual model is displayed as a graph. A confusion matrix and various statistics are used to evaluate the application of the model to the test data in order to obtain a statement about the suitability of the model.

```r
        first.model <- rpart(classe ~ ., data=sub.train.data.clean, method="class")

        # Predicting:
        first.prediction<- predict(first.model, sub.test.data.clean, type = "class")

        # Plot of the Decision Tree
        rpart.plot(first.model, extra=102, under=TRUE, faclen=0,cex=0.5)
```



```
    # Test results:
    confusionMatrix(first.prediction,factor(sub.test.data.clean$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1265  151   20   47   19
##          B   47  526   43   60   62
##          C   33  187  733   78  114
##          D   45   60   59  556   63
##          E    5   25    0   63  643
##
## Overall Statistics
##
##                Accuracy : 0.7592
##                  95% CI : (0.747, 0.7711)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6949
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9068   0.5543   0.8573   0.6915   0.7137
## Specificity            0.9325   0.9464   0.8982   0.9446   0.9768
## Pos Pred Value         0.8422   0.7127   0.6402   0.7101   0.8736
## Neg Pred Value         0.9618   0.8985   0.9675   0.9398   0.9381
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2580   0.1073   0.1495   0.1134   0.1311
## Detection Prevalence   0.3063   0.1505   0.2335   0.1597   0.1501
## Balanced Accuracy      0.9196   0.7503   0.8778   0.8181   0.8452
```

With regard to the results, it should be noted that the accuracy of 74% is not particularly high. For this reason, another model is being developed or tested.

Second prediction model:

A random forest model is developed using the same methodology as mentioned above. Identical trainings and test data are also used to generate the model in order to be able to make comparable statements.

```
        second.model <- randomForest(as.factor(classe) ~. , data = sub.train.data.clean, method="class")

        # Predicting:
        second.prediction<- predict(second.model, sub.test.data.clean, type = "class")

        # Test results on  subTesting data set:
        confusionMatrix(second.prediction,factor(sub.test.data.clean$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    2    0    0    0
##          B    0  946    1    0    0
##          C    0    1  854    0    0
##          D    0    0    0  804    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9976, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9988   1.0000   0.9989
## Specificity            0.9994   0.9997   0.9998   0.9998   1.0000
## Pos Pred Value         0.9986   0.9989   0.9988   0.9988   1.0000
## Neg Pred Value         1.0000   0.9992   0.9998   1.0000   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1741   0.1639   0.1835
## Detection Prevalence   0.2849   0.1931   0.1743   0.1642   0.1835
## Balanced Accuracy      0.9997   0.9983   0.9993   0.9999   0.9994
```

Decision:

As has been shown, the random forrest model has a higher accuracy than the decision tree model. Therefore, the random forest model is chosen and applied to the actual test data set.

Final modell results:

```r
 final.prediction <- predict(second.model, test.data, type="class")
final.prediction
```

```
##  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
```