Predict how well weight lifting exercises is done

Carlos Martinez Reyes

30/11/2020

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 do on a regular basis 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 belt, forearm, arm, and dumbbell accelerometer data from 6 participants and predict how they performed the exercise from the class variable in the training set.

Data Preprocessing

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------ tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
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:dplyr':
##
##     combine
```

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

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(RGtk2)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Versión 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

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

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

```
set.seed(9999)
```

Download the Data

```
trainfile   <- './data/pml-training.csv'
testfile <- './data/pml-testing.csv'
url_train    <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
url_test  <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
if (!file.exists("data")){
  dir.create("data")
}
download.file(url_train, trainfile)
download.file(url_test,testfile )
training   <-read.csv(trainfile)
```

Exploratory data analysis

Before training a predictive model, or even before performing any computation with a new data set, it is very important to perform a descriptive exploration of the data. This process allows us to better understand

what information each variable contains, as well as to detect possible errors. Furthermore, it can give clues as to which variables are not suitable as predictors in a model (accelerometer measurements).

Multiple columns do not have measures for each observation, rather they are summary statistics, we can omit many of the summary variables, we remove all rows that contain summary statistics and not observation data. We filter all columns that are only used to present summary data. It is also not convenient to include predictors that have a variance close to zero, that is, predictors that take only a few values, of which some appear very infrequently. The problem with the latter is that they can become predictors with zero variance when the observations are split by cross-validation or bootstrap.

Data description and cleaning

The pml-training: data / observations with which the model is trained. The pml-test data: data / observations of the same type as those that make up the training set but that have not been used in the creation of the model. They are data that the model has not "seen". One of the first checks to do after loading the data is to verify that each variable has been stored with the corresponding type of value, that is, that the numerical variables are numbers and the qualitative variables are factor, character or Boolean.

```
training   <-read.csv(trainfile, na.strings=c("NA","#DIV/0!", ""))
testing <-read.csv(testfile , na.strings=c("NA", "#DIV/0!", ""))
training<-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]
training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
dim(training)
```

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

```
dim(testing)
```

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

Now, the cleaned training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables. The "classe" variable is still in the cleaned training set.

Cross-validation

The simplest method of validation consists of dividing the available observations into two groups, one used to train the model and the other to evaluate it. The proper size of the partitions depends largely on the amount of data available and the security that is needed in estimating the error, 75% -25% usually gives good results. We will use the validation data set to conduct cross.

```
subSamples <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
subTraining <- training[subSamples, ]
subTesting <- training[-subSamples, ]
dim(subTraining)
```

```
## [1] 14718    53
```

```
dim(subTesting)
```

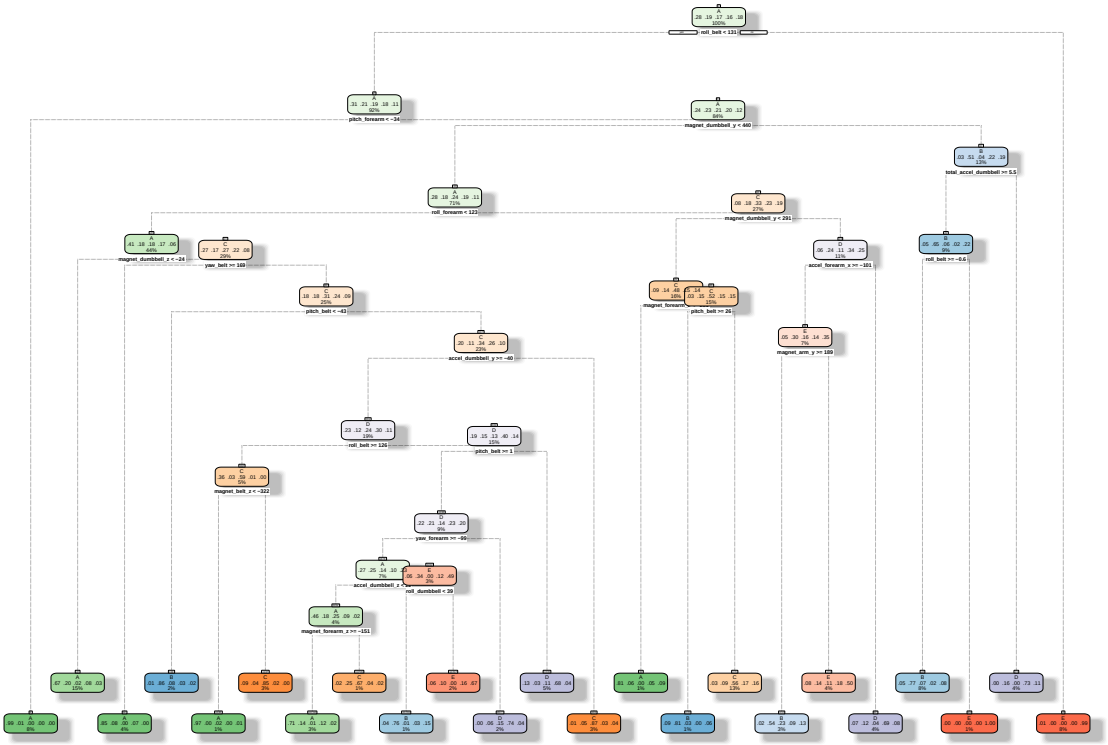
```
## [1] 4904   53
```

Prediction models

Once the data have been preprocessed and the predictors selected, the next step is to use a machine learning algorithm to create a model capable of representing the patterns present in the training data and generalizing them to new observations. An decision tree and random forest will be applied to the data.

```
modFitDT <- rpart(classe ~ ., data=subTraining, method="class")
predictDT <- predict(modFitDT, subTesting, type = "class")
fancyRpartPlot(modFitDT)
```

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



Predicting with the Decision Tree Model

```
set.seed(12345)
confusionMatrix(predictDT, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1266  208   25   91   29
##          B   33  535   71   30   67
##          C   28   90  676  130   94
##          D   45   72   59  501   43
##          E   23   44   24   52  668
##
## Overall Statistics
##
##                Accuracy : 0.7435
##                  95% CI : (0.731, 0.7557)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6738
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9075   0.5638   0.7906   0.6231   0.7414
## Specificity            0.8994   0.9492   0.9155   0.9466   0.9643
## Pos Pred Value         0.7820   0.7269   0.6640   0.6958   0.8237
## Neg Pred Value         0.9607   0.9007   0.9539   0.9276   0.9431
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2582   0.1091   0.1378   0.1022   0.1362
## Detection Prevalence   0.3301   0.1501   0.2076   0.1468   0.1654
## Balanced Accuracy      0.9035   0.7565   0.8531   0.7849   0.8528
```

```
accuracy <- postResample(predictDT, subTesting$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.7434747 0.6737973
```

So, the estimated accuracy of the model is 74.35%

Building the Random Forest Model

```
set.seed(12345)
modFitRF <- randomForest(classe ~ ., data=subTraining, method="class")
```

Predicting with the Random Forest Model

```
predictRF <- predict(modFitRF, subTesting, type = "class")
confusionMatrix(predictRF, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    2    0    0    0
##          B    1  946    8    0    0
##          C    0    1  847    7    0
##          D    0    0    0  795    2
##          E    0    0    0    2  899
##
## Overall Statistics
##
##                Accuracy : 0.9953
##                  95% CI : (0.993, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9941
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9968   0.9906   0.9888   0.9978
## Specificity            0.9994   0.9977   0.9980   0.9995   0.9995
## Pos Pred Value         0.9986   0.9906   0.9906   0.9975   0.9978
## Neg Pred Value         0.9997   0.9992   0.9980   0.9978   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1929   0.1727   0.1621   0.1833
## Detection Prevalence   0.2847   0.1947   0.1743   0.1625   0.1837
## Balanced Accuracy      0.9994   0.9973   0.9943   0.9942   0.9986
```

```
accuracy <- postResample(predictRF, subTesting$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9953100 0.9940673
```

So, the estimated accuracy of the model is 99.53%

Conclusion

The confusion matrices show, that the Random Forest algorithm performens better than decision trees. The accuracy for the Random Forest model was 0.9953 compared to 0.7435 for Decision Tree model. The random Forest model is choosen.

Predicting on the Testing Data (pml-testing.csv) with the Best Predictive Model

Random Forest Prediction

```
predict_quiz <- predict(modFitRF, testing, type = "class")
predict_quiz
```

```
##  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
```