Practical Machine Learning

Andrea Gilardi

4 7 2020

Downloading and cleaning data

Data can be downloaded from provided URLs

```
# Load necessary libraries

library(caret)
```

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

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

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

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Geben Sie 'rattle()' ein, um Ihre Daten mischen.
```

```
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:rattle':
##
##     importance
```

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

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
#Download dataset

training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")

#Create a partition on the training set

part <- createDataPartition(training$classe, p = 0.7, list = FALSE)
TrainSet <- training[part, ]
TestSet <- training[-part, ]
```

We will be using 70% of the data for training purposes and the remaining 30% for testing purposes.

```
dim(TrainSet)
```

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

```
dim(TestSet)
```

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

Out of the 160 variables, we can exclude those wo contains NA or that are approx. zero and also the 5 used for ID.

```
# remove NA
TrainSet <- TrainSet[ , colSums(is.na(TrainSet)) == 0]
TestSet <- TestSet[ , colSums(is.na(TestSet)) == 0]

# remove variables with approx zero variance
TrainSet<- TrainSet[, -nearZeroVar(TrainSet)]
TestSet <- TestSet[, -nearZeroVar(TestSet)]

# remove the first 5 columns -ID only

TrainSet<- TrainSet[ , -c(1:5)]
TestSet <- TestSet[ , -c(1:5)]
```

We have now reduced our variables from 160 to 54

```
dim(TrainSet)
```

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

```
dim(TestSet)
```

```
## [1] 5885   54
```

Training Model

Let´s now try to model the cleaned data; we now from the course that random forest usually perform very well in this kind of inquiries so we will fit it to our datasets. The confusion matrix provide us a better overview about accuracy of the different models.

```
set.seed(12345)
ctr <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
RF <- train(classe ~., data = TrainSet, method = "rf", trControl = ctr)
RF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.24%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B    9 2648    1    0    0 0.0037622272
## C    0    4 2391    1    0 0.0020868114
## D    0    0   12 2240    0 0.0053285968
## E    0    0    0    5 2520 0.0019801980
```

```
#prediction on test dataset
predRF <- predict(RF, TestSet)
#confusion matrix
confMatRF <- confusionMatrix(predRF, TestSet$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1131    2    0    0
##          C    0    1 1024    5    0
##          D    0    2    0  956    0
##          E    0    0    0    3 1082
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9969          
##                  95% CI : (0.9952, 0.9982)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9961          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9930   0.9981   0.9917   1.0000
## Specificity            0.9988   0.9996   0.9988   0.9996   0.9994
## Pos Pred Value         0.9970   0.9982   0.9942   0.9979   0.9972
## Neg Pred Value         1.0000   0.9983   0.9996   0.9984   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1922   0.1740   0.1624   0.1839
## Detection Prevalence   0.2853   0.1925   0.1750   0.1628   0.1844
## Balanced Accuracy      0.9994   0.9963   0.9984   0.9956   0.9997
```

With 99,75% accuracy we can say that our model is really good.

Apply the model to the test data

We can finally apply the random forest model to the 20 cases provided as test-dataset-

```
TEST <- predict(RF, testing)
TEST
```

```
##  [1] 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
```