Practical Machine Learning Course Project

Silva, RAFAEL

July 19, 2018

Setup

This section presents all the libraries used during the project.

```r
library(caret)
library(doParallel)
```

Introduction

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 dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website in [this link](http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The chunk below was used to download and load the training and test datasets into R.

```r
# create directory
if(!dir.exists("./data")) dir.create("./data")
# download files
if(!file.exists("./data/pml-training.csv")) {
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                "./data/pml-training.csv")
}
if(!file.exists("./data/pml-testing.csv")) {
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                "./data/pml-testing.csv")
}
# read files
training <- read.csv("data/pml-training.csv", na.strings = c("", "NA"))
testing <- read.csv("data/pml-testing.csv", na.strings = c("", "NA"))
```

Data cleaining and exploration

The first thing to look at in a new dataset is its dimensions. The code below takes care of that.

```r
dim(training)
```

```
## [1] 19622   160
```

It is easily noticeable that the amount of variables might be a problem. Unless a super computer is available, training a machine learning algorithm in so many variables might take too much time.

In a closer look it is possible to see that most of these variables are summary variables that are calculated every given period of time for some of the actual acquisitions. For this particular project, these variables have no use at all as they are not available for prediction in the testing dataset. The following code chunk identifies all those variables and prints the first 12 as an example.

```
unavailableVars <- which(colSums(is.na(training))>0)
names(training)[unavailableVars][1:12]
```

```
##  [1] "kurtosis_roll_belt"   "kurtosis_picth_belt"  "kurtosis_yaw_belt"
##  [4] "skewness_roll_belt"   "skewness_roll_belt.1" "skewness_yaw_belt"
##  [7] "max_roll_belt"        "max_picth_belt"       "max_yaw_belt"
## [10] "min_roll_belt"        "min_pitch_belt"       "min_yaw_belt"
```

Besides the variable identified above, there are some variables that, although they might improve the accuracy of some given model, they should not be relevant when identifying the quality of the exercise being executed, considering them would deliberately overfit such model. These variable are: `X`, `user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp`, `new_window`, `num_window`, and they are removed from the dataset along with the variables previously identified by next code chunk.

```
training <- training[, -c(1:7, unavailableVars)]
```

Model fitting and cross validation

In this section the machine learning algorithm is trained and cross validated. Although the cross validation might provide a good estimate of the out-of-sample (OOS) error, the training dataset is big enough to be divided and a validation set can be created.

```
set.seed(13853)

inVal <- createDataPartition(training$classe, p = 0.25, list = FALSE)

validating <- training[inVal,]
trainingSub <- training[-inVal,]
```

To create the model, the random forest method was chosen for its accuracy and compatibility with parallel processing. The cross validation was done in 10 folds. The results are printed bellow

```
set.seed(1123)

cl <- makePSOCKcluster(6)
registerDoParallel(cl)

mod <- train(classe ~ ., data = trainingSub, method = "rf", trControl = trainControl(method = "cv", numb

stopCluster(cl)

mod
```

```
## Random Forest
##
## 14715 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13245, 13243, 13245, 13244, 13243, 13243, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9921852  0.9901131
##   27    0.9930004  0.9911455
##   52    0.9877676  0.9845255
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Validation

Although the model seems to be very accurate based on the on the OOS accuracy estimate of `0.993`, in this section, it was verified how the model performs on a completely new dataset, so the OOS accuracy can be confirmed. The results are shown in the confusion matrix bellow.

```r
confusionMatrix(validating$classe, predict(mod, validating))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    0    0    0    1
##          B    5  944    1    0    0
##          C    0    0  851    5    0
##          D    0    0   11  793    0
##          E    0    0    3    2  897
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9918, 0.9962)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9928
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   1.0000   0.9827   0.9912   0.9989
## Specificity            0.9997   0.9985   0.9988   0.9973   0.9988
## Pos Pred Value         0.9993   0.9937   0.9942   0.9863   0.9945
## Neg Pred Value         0.9986   1.0000   0.9963   0.9983   0.9998
## Prevalence             0.2851   0.1924   0.1765   0.1630   0.1830
## Detection Rate         0.2841   0.1924   0.1734   0.1616   0.1828
## Detection Prevalence   0.2843   0.1936   0.1744   0.1638   0.1838
## Balanced Accuracy      0.9981   0.9992   0.9907   0.9943   0.9988
```

Conclusion and testing

The tests run above show that the final model is very accurate. This model can now be applied to the test set.

```
results <- predict(mod, testing)
```

The results where intentionally not printed in this report to comply with the student honor code as much as possible. They where presented and approved on the quiz.