Practical Machine Learning Assignment

Tommaso Bicego 25/07/2014 Milano, Italia (IT)

Synopsis

The task of the project is to correctly identify how well the test subjects executed lifts, by analysing the data obtained thanks to accelerometers located on the belt, forearm, arm, and dumbbell of the participants. How well means to identify into which of the five different *classes* the particular execution of the lift falls.

Data processing

To process the data I begin by importing the following libraries

```
library(caret)
library(foreach)
library(plyr)
library(gbm)
library(randomForest)
```

Then I read the downloaded datasets

```
train_csv <- read.csv("pml-training.csv",stringsAsFactors = TRUE)
test_csv <- read.csv("pml-testing.csv")
```

I'm going to clean our datasets from the following columns/data:

- columns that contain missing values (!NAs);
- columns that contain blank spaces;
- the columns 'X', 'user_name', 'new_window', 'num_window' because they are identifiable;
- the timestamps ('raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp')

```
train_csv_not_nas <- train_csv[ , colSums(is.na(train_csv)) == 0]
train_csv_not_blanks <- train_csv_not_nas[ , !colSums(train_csv_not_nas=="")]
final_training <- train_csv_not_blanks[8:60]
```

```
chosen_columns <- names(final_training)
final_testing <- test_csv[names(test_csv) %in% chosen_columns]

test_csv_not_nas <- test_csv[ , colSums(is.na(test_csv)) == 0]
test_csv_not_blanks <- test_csv_not_nas[ , !colSums(test_csv_not_nas=="")]
final_testing <- test_csv_not_blanks[8:60]
```

Now I create two sets by splitting the final_training set into a new train (70% of the whole training set) and test set(the other 30%). I will use them as a sort of preliminary test executed on the training data in order to decide which is the best predictive model among some of the ones taught during the course and to use it later on the final_testing set. I also set the seed to make the data reproducible.

Here the two sets:

```r
set.seed(12345)

indexTraining <- createDataPartition(y = final_training$classe, p = 0.7, list = FALSE)
petit_train <- final_training[indexTraining,]
petit_test <- final_training[-indexTraining,]
```

Comparisons

Now I compare some of the predictive models explained in the lessons and see which is the "best" at fitting the "petit" sets (from this point onward the code is all commented because otherwise it takes too long to execute all the methods at once).

- rpart (Recursive Partitioning and Regression Trees)

```r
#fit_rpart <- train(classe~., method = "rpart", data =petit_train)
#p_fit_rpart <- predict(fit_rpart, petit_test)
#confusionMatrix(p_fit_rpart, petit_test$classe)
```

```
Confusion Matrix and Statistics

        Reference
    Prediction      A    B    C    D    E
                A 1494  470  467  438  141
                B   21  380   29  184  147
                C  128  289  530  342  277
                D    0    0    0    0    0
                E   31    0    0    0  517

    Overall Statistics
    Accuracy : 0.4963
    95% CI : (0.4835, 0.5092)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16
    Kappa : 0.3425
    Mcnemar's Test P-Value : NA
```

- lda (Linear Discriminant Analysis Classification)

```r
#fit_lda <- train(classe~., method = "lda", data=petit_train)
#p_fit_lda <- predict(fit_lda, petit_test)
#confusionMatrix(p_fit_lda, petit_test$classe)
```

```
  Confusion Matrix and Statistics

        Reference
    Prediction      A    B    C    D    E
```

```
            A 1380  191   91   48   38
            B   36  709  104   49  193
            C  133  142  658  125   99
            D  116   43  144  698  100
            E    9   54   29   44  652

    Overall Statistics
    Accuracy : 0.6962
    95% CI : (0.6842, 0.7079)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16
    Kappa : 0.6155
    Mcnemar's Test P-Value : < 2.2e-16
```

- nb (Naive Bayes Classification)

```
#fit_nb <- train(classe~., method = "nb", data =petit_train)
#p_fit_nb <- predict(fit_nb, petit_test)
#confusionMatrix(p_fit_nb, petit_test$classe)
```

```
Confusion Matrix and Statistics

        Reference
    Prediction      A    B    C    D    E
            A 1504  246  250  198   64
            B   32  741   73    2   86
            C   43   82  655  111   42
            D   86   57   47  606   43
            E    9   13    1   47  847

    Overall Statistics
    Accuracy : 0.7397
    95% CI : (0.7283, 0.7509)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16
    Kappa : 0.6664
    Mcnemar's Test P-Value : < 2.2e-16
```

- gbm (Boosting with Trees Classification)

```
#fit_gbm <- train(classe~., method = "gbm", data =petit_train)
#p_fit_gbm <- predict(fit_gbm, petit_test)
#confusionMatrix(p_fit_gbm, petit_test$classe)
```

```
  Confusion Matrix and Statistics

        Reference
```

```
Prediction      A    B    C    D    E
            A 1648   36    0    2    2
            B   16 1067   46    4   15
            C    4   30  963   33    5
            D    6    6   15  919   19
            E    0    0    2    6 1041

Overall Statistics
Accuracy : 0.958
95% CI : (0.9526, 0.963)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.9469
Mcnemar's Test P-Value : 3.655e-07
```

- parRF (Random Forest)

```r
#fit_parRF <- train(classe~., method = "parRF", data =petit_train)
#p_fit_parRF <- predict(fit_parRF, petit_test)
#confusionMatrix(p_fit_parRF, petit_test$classe)
```

```
Confusion Matrix and Statistics

    Reference
Prediction      A    B    C    D    E
            A 1673   13    0    0    0
            B    1 1119   14    0    0
            C    0    7 1008   25    0
            D    0    0    4  939    3
            E    0    0    0    0 1079

Overall Statistics
Accuracy : 0.9886
95% CI : (0.9856, 0.9912)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.9856
Mcnemar's Test P-Value : NA
```

Results

The results are the followings:

```
Method  Accuracy    K

rpart     0.4963    0.3425
lda       0.6962    0.6155
nb        0.7397    0.6664
gbm       0.958     0.9469
parRF     0.9886    0.9856
```

Random forest is the most accurate model with a 98.9% of accuracy, a K of 0.98, where K measures the agreement of the prediction with the real class, **and an error rate of 1-Accuracy = 1.1%**. Therefore we execute the final prediction on the final_testing set with this method.

```
#prediction <- predict(p_fit_parRF, final_testing)
#prediction
```