Practical Machine Learning Project

Antonio Rubio Calzado

29 de mayo de 2017

Introduction.

For this project we are given data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The training data is labeld with the manner in which they did the exercise. We are asked to developed an algorithm able to predict this label and perform it on a testing set (data without this labels).

Cleaning and processing data.

Firstly, we download the training and testing data on our computer respectively from:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

If we open our raw data, there are multiple fields with "NA" , "#DIV/0!" or "", so we will think they are NA values when reading it.

```
setwd("C:/Users/arubioca/Desktop/MLPROJECT")
train <- read.csv("pml-training.csv",dec=".",na.strings = c("NA","",
"#DIV/0!"))
test <-  read.csv("pml-testing.csv",dec=".",na.strings = c("NA","",
"#DIV/0!"))
```

Now, lets load the libraries neccessary for this project

```
suppressMessages(library(caret))
suppressMessages(library(randomForest))
suppressMessages(library(rpart))
```

The next step is using the **caret** package to create a partition of our training data in two subgroups: The first one will have the 60% of the size of this data and the other one the remaining 40%. This will be used to do cross-validations of our future predictive algorithms.

```
set.seed(100)
inTrain <- createDataPartition(train$classe, p = 0.6, list = FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]
```

Let's remove near zero variance predictors from our data. With the following commands, we are detecting the variables that are near zero variance predictors in the train dataset.

```r
set.seed(100)
nzv <- nearZeroVar(train, saveMetrics = TRUE)
rm_nzv_index <- which(nzv$zeroVar == TRUE)
```

Also notice that many fields in our test.csv are constantly NA. We this procedure, we save in a vector the colums of the train dataset with this behaviour:

```r
rm_NA_test_index <- c()
for (i in 1:160){
  if (all(is.na(test[,i]))){
     rm_NA_test_index <- c(rm_NA_test_index,i)
    }
  }
```

Now we are going to create a vector with the variables that can be dropped from our train dataset, since they are constantly NA or near zero variance variables.

```r
index <- unique(c(rm_nzv_index, rm_NA_test_index))
training_new <- training[,-index]
testing_new <- testing[,-index]
```

The variables `X`, `user_name`, `raw_timestamp_part_1`, `raw_timestamp_par_2`,`cvtd_timestamp`, `new_window` and `num_window` can be ommited in our study, so we also dropped them:

```r
training_new <- training_new[,-c(1,2,3,4,5,6,7)]
testing_new <- testing_new[,-c(1,2,3,4,5,6,7)]
```

Predictive Models

We start fitting a CART-tree model on our training set, using the package `rpart`:

```r
set.seed(100)
model_tree <- rpart(classe ~ ., data = training_new, method = "class")
```

After training phase, we are studying the out-of-sample error of this model, by predicting a classe label on our testing set and comparing those predictions with the real classe labels via the confussion Matrix:

```r
predictions_tree <- predict(model_tree, testing_new, type = "class")
confusionMatrix(predictions_tree, testing_new$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1968  353   20  121   41
##          B   48  845   83   40   92
##          C   58  143 1170  192  163
##          D   85  111   81  837   74
##          E   73   66   14   96 1072
```

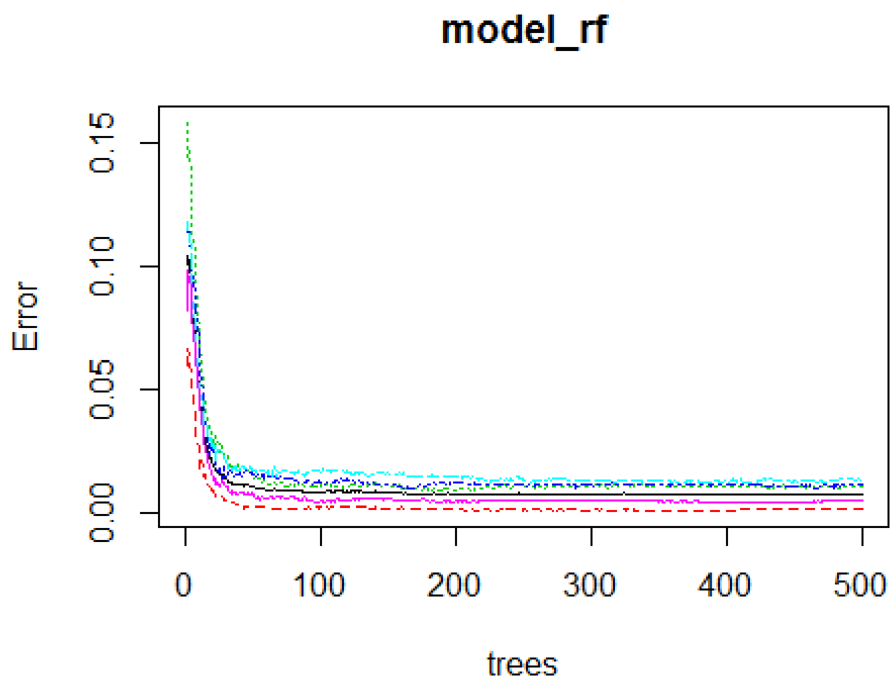
```
##
## Overall Statistics
##
##                Accuracy : 0.751
##                  95% CI : (0.7412, 0.7605)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6839
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8817   0.5567   0.8553   0.6509   0.7434
## Specificity            0.9047   0.9584   0.9142   0.9465   0.9611
## Pos Pred Value         0.7863   0.7626   0.6779   0.7045   0.8115
## Neg Pred Value         0.9506   0.9001   0.9676   0.9326   0.9433
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2508   0.1077   0.1491   0.1067   0.1366
## Detection Prevalence   0.3190   0.1412   0.2200   0.1514   0.1684
## Balanced Accuracy      0.8932   0.7575   0.8847   0.7987   0.8523
```

The accuracy of the model is only 75.1%, so it's a good idea try to fit another different model.

Let's repeat this proccess with a random forest algorithm. We start training the model using the `randomForest` package:

```r
set.seed(100)
model_rf <- randomForest(classe ~ ., data = training_new)
```

The following plot shows a graph of Error vs Trees in the previous random forest model:



Now we are checking the accuracy of this model:

```
set.seed(100)
predictions_rf <- predict(model_rf, testing_new, type = "class")
confusionMatrix(predictions_rf, testing_new$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    6    1    0    0
##          B    2 1509    8    0    0
##          C    0    3 1356   15    2
##          D    0    0    3 1270    6
##          E    0    0    0    1 1434
##
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9924
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9941   0.9912   0.9876   0.9945
## Specificity            0.9988   0.9984   0.9969   0.9986   0.9998
## Pos Pred Value         0.9969   0.9934   0.9855   0.9930   0.9993
## Neg Pred Value         0.9996   0.9986   0.9981   0.9976   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1923   0.1728   0.1619   0.1828
## Detection Prevalence   0.2851   0.1936   0.1754   0.1630   0.1829
## Balanced Accuracy      0.9989   0.9962   0.9941   0.9931   0.9971
```

The accuracy of this model is 99.4%, so the out-of-sample is only 0.6% and hence, this random forest model highly improves the before CART-tree model.

Predictions

We are going to predict the classe-label for test data with the random forest model that we have previously trained.

The first thing is to drop all the variables that aren't used:

```
test_new <- test[,-index]
test_new <- test_new[,-c(1,2,3,4,5,6,7)]
```

And lastly, we make the prediction we have been asked for:

```
pred <- predict(model_rf, test_new, type = "class")
print(as.character(pred))

##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```