Practical ML : Course Project

Marcos Gestal

Friday, May 08, 2015

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

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases

Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Load data and pre-process steps

```
train <- read.csv(trainData, na.strings=c("NA", "#DIV/0!"))
test <- read.csv(testData, na.strings=c("NA", "#DIV/0!"))

# Pre-processing of datasets: clean features (columms) with all values of NAs in
# the train dataset

notNullFeatures <- colSums(is.na(train))==0

train <- train[, notNullFeatures]
test <- test[, notNullFeatures]

irrelevantFeatures <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
                        "cvtd_timestamp", "new_window", "num_window")
```

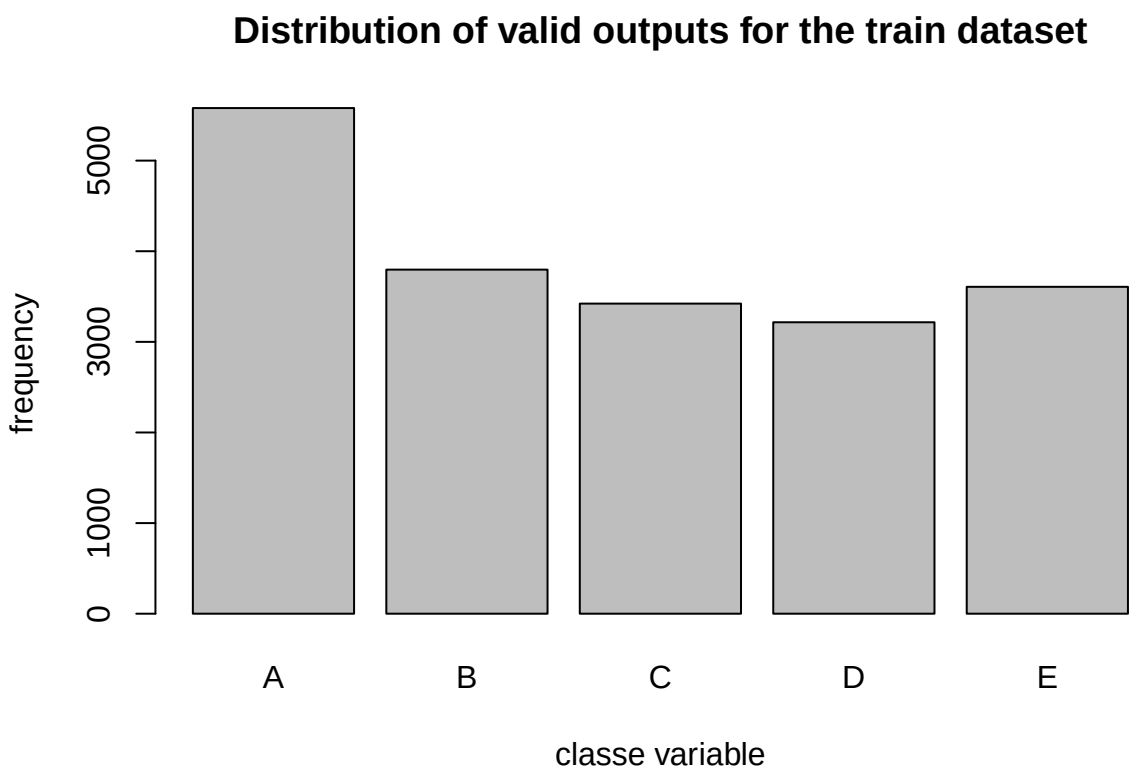
```
train <- train[, !names(train) %in% irrelevantFeatures]
test <- test[, !names(test) %in% irrelevantFeatures]
```

The final dataset used for training phase cointains 19622 patterns with 53 features each one. On the other hand the test dataset contains a total of 20 patterns with the *same* features.

Data overview

The outcome variable for our work is labelled as `classe`. It contains 5 different levels with the following distribution

```
plot(train$classe,
     main="Distribution of valid outputs for the train dataset",
     xlab="classe variable", ylab="frequency")
```



As the graphics shows, class A is a little more present in the dataset, but there is not any class either over- or under- represented. Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Read more: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises

Partitioning the training set

We split the original data set, with the 60% of that samples used for train phase and the 40% for the validation (random subsampling without replacement)

```
library(caret)
library(kernlab)
library(rpart.plot)

set.seed(12345)
inTrain <- createDataPartition(train$classe, p = 0.6, list = FALSE)

train <- train[inTrain, ]
validation <- train[-inTrain, ]
```

Artificial Neural Networks

```
library(nnet)
system.time(modelNet <- nnet(classe ~ . , data = train, size=17, maxit=2000,
                             abstol=1e-6, preProc=c("center", "scale"), trace=FALSE))
```

```
##    user  system elapsed
##  474.72    0.14  486.75
```

```
predictionsNet <- predict(modelNet, newdata = validation, type="class")
confusionMatrix(predictionsNet, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1054  108   54  112   47
##          B   19  444   67   38  207
##          C   99  129  655  133  133
##          D  135   68   19  444   74
##          E   18  174   31   50  418
##
## Overall Statistics
##
##                Accuracy : 0.6374
##                  95% CI : (0.6235, 0.6511)
##     No Information Rate : 0.2801
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5418
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7955  0.48104   0.7930  0.57143  0.47554
## Specificity            0.9057  0.91305   0.8735  0.92512  0.92911
## Pos Pred Value         0.7665  0.57290   0.5701  0.60000  0.60492
## Neg Pred Value         0.9192  0.87889   0.9522  0.91654  0.88586
## Prevalence             0.2801  0.19514   0.1746  0.16427  0.18584
```

```
## Detection Rate         0.2228  0.09387   0.1385  0.09387  0.08837
## Detection Prevalence   0.2907  0.16385   0.2429  0.15645  0.14609
## Balanced Accuracy      0.8506  0.69705   0.8332  0.74827  0.70232
```

Regression Trees

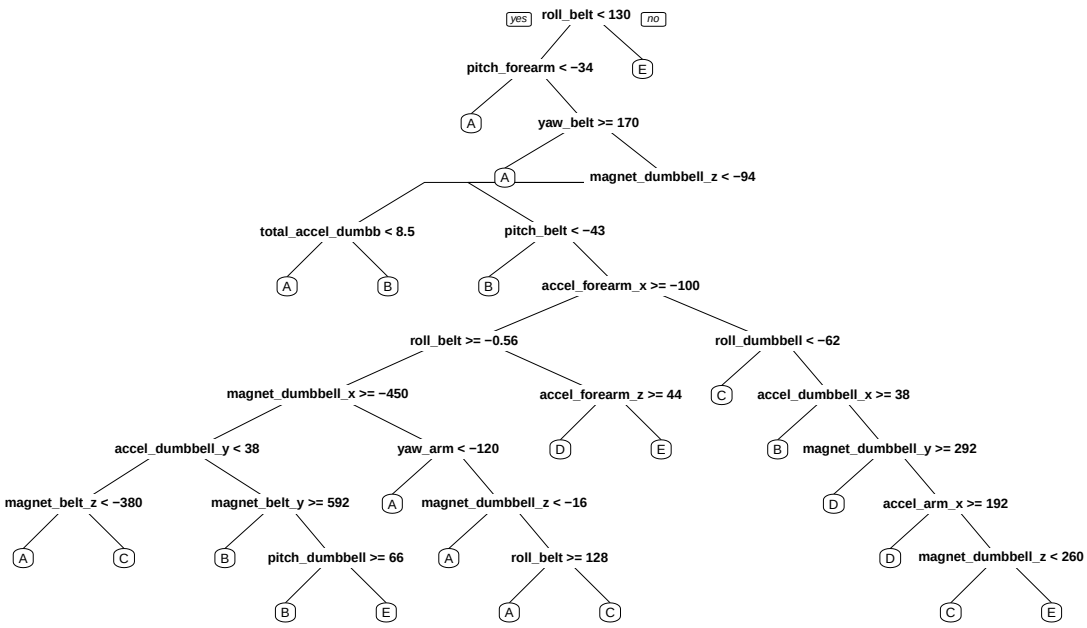
```
system.time(modelTree <- rpart (classe ~ ., data=train, method="class"))
```

```
##    user  system elapsed
##    3.37    0.00    3.40
```

```
predictionsTree <- predict(modelTree, newdata = validation, type="class")
confusionMatrix(predictionsTree, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1108  163   18   40   37
##          B   24  491   62   22   28
##          C   67  177  739  190  138
##          D  103   82    7  514   38
##          E   23   10    0   11  638
##
## Overall Statistics
##
##                Accuracy : 0.7378
##                  95% CI : (0.7251, 0.7503)
##     No Information Rate : 0.2801
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6691
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8362   0.5320   0.8947   0.6615   0.7258
## Specificity            0.9242   0.9643   0.8535   0.9418   0.9886
## Pos Pred Value         0.8111   0.7831   0.5637   0.6909   0.9355
## Neg Pred Value         0.9355   0.8947   0.9746   0.9340   0.9405
## Prevalence             0.2801   0.1951   0.1746   0.1643   0.1858
## Detection Rate         0.2342   0.1038   0.1562   0.1087   0.1349
## Detection Prevalence   0.2888   0.1326   0.2772   0.1573   0.1442
## Balanced Accuracy      0.8802   0.7481   0.8741   0.8017   0.8572
```

```
rpart.plot(modelTree, main="Classification Tree for pml data")
```



Random Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
system.time(modelRF <- randomForest(classe ~ ., data=train, method="class"))
```

```
##    user  system elapsed
##   50.21    0.36   50.65
```

```
predictionsRF <- predict(modelRF, newdata = validation, type="class")
confusionMatrix(predictionsRF, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1325    0    0    0    0
```

```
##          B    0  923    0    0    0
##          C    0    0  826    0    0
##          D    0    0    0  777    0
##          E    0    0    0    0  879
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9992, 1)
##     No Information Rate : 0.2801
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2801   0.1951   0.1746   0.1643   0.1858
## Detection Rate         0.2801   0.1951   0.1746   0.1643   0.1858
## Detection Prevalence   0.2801   0.1951   0.1746   0.1643   0.1858
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Conclussions

What you should submit

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5.
2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above.

Submission

```
predictionsFinal <- predict(modelRF, newdata = test, type="class")
predictionsFinal
```

```
##  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
```

```
# Create individual files for the submissions of the predictions

pml_write_files = function(x){
  n = length(x)
```

```
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predictionsFinal)
```