Practical Machine Learning

Germán Hernández

30/7/2020

Summary

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).Below are the links to training data and testing data that we are using to build the model.

```
Url1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Model Construction

The basic aim of this prediction assignment is to build two classifier models and compare them with each other to derive conclusions.I am using decision trees and random forest classifier methods in this scenario.Finally i combine the two predictor models by model stacking and use the new predictor model to make the preictions.Out of the three models the model wil be choosen to predict the test data. The models are built to predict the classe variable, a factor variable with 5 levels.

Cross-validation

When considering the size of the training dataset it would be better if we split the training dataset into training subset,testing subset and the validation subset.First we would train the training sub dataset with the both trees and random forest models and use the validation dataset on the combined predictor model.

```
train_set <- read.csv(url(Url1),na.strings=c("NA","#DIV/0!",""))
test_set <- read.csv(url(Url2),na.strings=c("NA","#DIV/0!",""))
```

```
dim(train_set)
```

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

Expected Out of Sample Error

Expected out of sample error is the most crucial factor which determine how succesful our model is.Here the expected out of sample error will be calculated from the test dataset and it will be calculated based on the observations that our model misclassified.So model is more useful when the out of sample error is minimized.

1.Data pre-processing

```r
str(train_set)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/201
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
```

```
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Removing variables with too many missing values

```
train_set_slash <- train_set
for (i in 1:ncol(train_set)) {
  if( sum( is.na( train_set[, i] ) ) /nrow(train_set) >= .4 ){
    for(j in 1:ncol(train_set_slash)) {
            if( length( grep(names(train_set[i]), names(train_set_slash)[j]) ) ==1)  {
                train_set_slash <- train_set_slash[ , -j]
            }
    }
  }
}
## The above function is used to remove the variables from dataset with missing values more than 40%, t
```

```
test_set_slash <- test_set
for (i in 1:ncol(test_set)) {
  if( sum( is.na( test_set[, i] ) ) /nrow(test_set) >= .4 ){
    for(j in 1:ncol(test_set_slash)) {
            if( length( grep(names(test_set[i]), names(test_set_slash)[j]) ) ==1)  {
                test_set_slash <- test_set_slash[ , -j]
            }
    }
  }
}
```

```
train_set <- train_set_slash
test_set <- test_set_slash
dim(train_set);dim(test_set)
```

```
## [1] 19622    60
```

```
## [1] 20 60
```

We can see that the number of variables have reduced to 60.

Removing data with near zero variance

```
library(caret);library(rpart);library(rpart.plot);library(rattle);library(randomForest);library(RColorB
```

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

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

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

```
## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Versión 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.

## 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
```

```
NZV <- nearZeroVar(train_set, saveMetrics=TRUE)
```

```
train_set <- train_set[,NZV$nzv == "FALSE"]
test_set <- test_set[,NZV$nzv == "FALSE"]
```

```
dim(train_set);dim(test_set)
```

```
## [1] 19622    59

## [1] 20 59
```

We can see that only one dimension is reduced.Let's check if there are any missing values that needed to be imputed

```
sum(is.na(train_set)==TRUE)
```

```
## [1] 0
```

```
test_set <- test_set[,-60]
```

Dropping the id column in the both datasets

```
train_set <- train_set[,-1]
test_set <- test_set[,-1]
```

2.Data slicing

```r
inBuild_data <- createDataPartition(y = train_set$classe,p = .7,list = FALSE)
validation_subset <- train_set[-inBuild_data,] ;Build_data <- train_set[inBuild_data,]
in_train<- createDataPartition(y= Build_data$classe,p = .7,list = FALSE)
train_subset <- Build_data[in_train,]
test_subset <- Build_data[-in_train,]
```

In here the data has been sliced into 3 parts train__subset,test__subtest and validation__subset repectively.We are using a validation set here in order reduce the out of sample error because we are going to trian the first two models and combine them to form a better model.

```r
dim(train_subset)
```

```
## [1] 9619   58
```

```r
dim(test_subset)
```

```
## [1] 4118   58
```

```r
dim(validation_subset)
```

```
## [1] 5885   58
```

3.Model Building

Fitting the randomForest model

```r
train_subset$classe <- as.factor(train_subset$classe)
Modelfit_1 <- randomForest(classe ~ ., data = train_subset)
```

```r
test_subset$classe <- as.factor(test_subset$classe)
predictions_1 <- predict(Modelfit_1, test_subset)
```

```r
confusionMatrix(predictions_1, test_subset$classe)
```

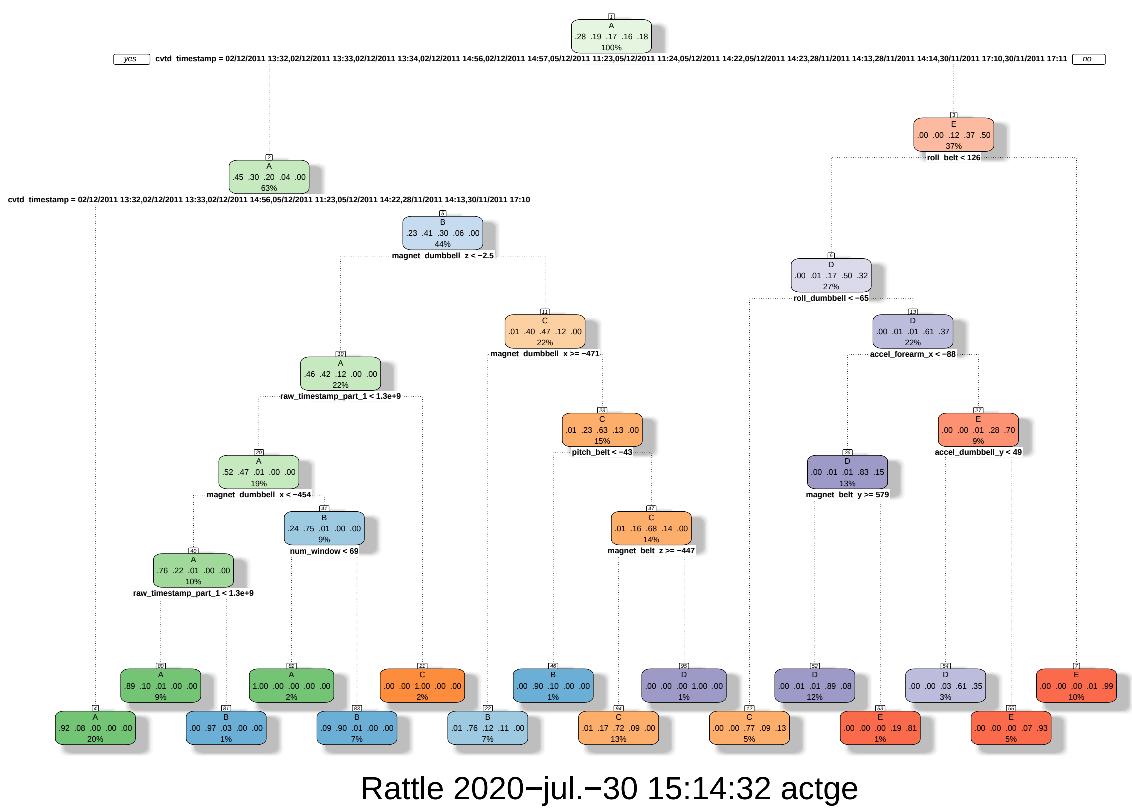
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1169    3    0    0    0
##          B    2  794    2    0    0
##          C    0    0  714    2    0
##          D    0    0    2  673    1
##          E    0    0    0    0  756
##
## Overall Statistics
##
##                Accuracy : 0.9971
##                  95% CI : (0.9949, 0.9985)
```

```
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9963
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9983   0.9962   0.9944   0.9970   0.9987
## Specificity            0.9990   0.9988   0.9994   0.9991   1.0000
## Pos Pred Value         0.9974   0.9950   0.9972   0.9956   1.0000
## Neg Pred Value         0.9993   0.9991   0.9988   0.9994   0.9997
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2839   0.1928   0.1734   0.1634   0.1836
## Detection Prevalence   0.2846   0.1938   0.1739   0.1642   0.1836
## Balanced Accuracy      0.9986   0.9975   0.9969   0.9981   0.9993
```

Fitting the Decision Trees Model

```r
Modelfit_2 <- rpart(train_subset$classe ~ ., data = train_subset,method = "class")
```

```r
fancyRpartPlot(Modelfit_2)
```



```r
predictions_2 <- predict(Modelfit_2,test_subset,type = "class")
```

```r
confusionMatrix(predictions_2, test_subset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1136  119    4    1    0
##          B   20  590   48   34    0
##          C   15   84  656   79   28
##          D    0    4   10  529   84
##          E    0    0    0   32  645
##
## Overall Statistics
##
##                Accuracy : 0.8635
##                  95% CI : (0.8527, 0.8739)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.827
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9701   0.7403   0.9136   0.7837   0.8520
## Specificity            0.9579   0.9693   0.9394   0.9715   0.9905
## Pos Pred Value         0.9016   0.8526   0.7610   0.8437   0.9527
## Neg Pred Value         0.9878   0.9396   0.9810   0.9582   0.9675
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2759   0.1433   0.1593   0.1285   0.1566
## Detection Prevalence   0.3060   0.1680   0.2093   0.1523   0.1644
## Balanced Accuracy      0.9640   0.8548   0.9265   0.8776   0.9213
```

Although at the begining of the assignment my intention was to combine the two predictor models ,by looking at the accuracy levels of the two models its not necessary to combine the two preictor model. RandomForest model is best model amomg the two models according to the accuracy level so let's choose that model.Lets test the choosen model on the validation test.

```r
validation_subset$classe <- as.factor(validation_subset$classe)
predictions_3 <- predict(Modelfit_1,validation_subset)
```

```r
confusionMatrix(predictions_3, validation_subset$classe)
```

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

```
##          A 1674    3    0    0    0
##          B    0 1136    5    0    0
##          C    0    0 1017    0    0
##          D    0    0    4  964    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9956, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9912   1.0000   0.9963
## Specificity            0.9993   0.9989   1.0000   0.9984   1.0000
## Pos Pred Value         0.9982   0.9956   1.0000   0.9918   1.0000
## Neg Pred Value         1.0000   0.9994   0.9982   1.0000   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1930   0.1728   0.1638   0.1832
## Detection Prevalence   0.2850   0.1939   0.1728   0.1652   0.1832
## Balanced Accuracy      0.9996   0.9982   0.9956   0.9992   0.9982
```

4.Tesiting the model on the Test set

```r
train_set$classe <- as.factor(train_set$classe)
Final_model <- randomForest(classe ~. ,data = train_subset )
```

```r
Final_predictions <- predict(Final_model,test_set)
Final_predictions
```

```
##  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
```