Practical-machine-learning Prediction Assignment

Saurabh Ghadge

12/02/2022

Problem Statement

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

DATA

The training data for this project are available here:
train (The%20training%20data%20for%20this%20project%20are%20available%20here:)
The test data are available here:
test (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 (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

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.

Reading Data

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
pmlval <- read_csv("pml-testing.csv")
```

```
## New names:
## * `` -> ...1
```

```
## Rows: 20 Columns: 160
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (3): user_name, cvtd_timestamp, new_window
## dbl  (57): ...1, raw_timestamp_part_1, raw_timestamp_part_2, num_window, rol...
## lgl (100): kurtosis_roll_belt, kurtosis_picth_belt, kurtosis_yaw_belt, skewn...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pmltrain <- read_csv("pml-training.csv")
```

```
## New names:
## * `` -> ...1
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
## Rows: 19622 Columns: 160
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (34): user_name, cvtd_timestamp, new_window, kurtosis_roll_belt, kurtos...
## dbl (126): ...1, raw_timestamp_part_1, raw_timestamp_part_2, num_window, rol...
```


```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```


```
intrain <- createDataPartition(y = pmltrain$classe, p = 0.6, list = F)
pmltrain <- pmltrain[intrain,]
test <- pmltrain[-intrain,]
```

Let's take a glance at data…

```
head(pmltrain)
```

```
## # A tibble: 6 x 160
##    ...1 user_name raw_timestamp_par~ raw_timestamp_pa~ cvtd_timestamp new_window
##   <dbl> <chr>                  <dbl>             <dbl> <chr>          <chr>
## 1     3 carlitos          1323084231            820366 05/12/2011 11~ no
## 2     6 carlitos          1323084232            304277 05/12/2011 11~ no
## 3     7 carlitos          1323084232            368296 05/12/2011 11~ no
## 4     8 carlitos          1323084232            440390 05/12/2011 11~ no
## 5    13 carlitos          1323084232            560359 05/12/2011 11~ no
## 6    15 carlitos          1323084232            604281 05/12/2011 11~ no
## # ... with 154 more variables: num_window <dbl>, roll_belt <dbl>,
## #   pitch_belt <dbl>, yaw_belt <dbl>, total_accel_belt <dbl>,
## #   kurtosis_roll_belt <chr>, kurtosis_picth_belt <chr>,
## #   kurtosis_yaw_belt <chr>, skewness_roll_belt <dbl>,
## #   skewness_roll_belt.1 <chr>, skewness_yaw_belt <chr>, max_roll_belt <dbl>,
## #   max_picth_belt <dbl>, max_yaw_belt <chr>, min_roll_belt <dbl>,
## #   min_pitch_belt <dbl>, min_yaw_belt <chr>, amplitude_roll_belt <dbl>, ...
```

```
dim(pmltrain)
```

```
## [1] 11776   160
```

```
any(is.na(pmltrain))
```

```
## [1] TRUE
```

```
sum(is.na(pmltrain))
```

```
## [1] 1152609
```

Looking at above outcomes we see that their are some column which are not making that much sense so that they can take part in model building. We will drop those column, also we see that their is lot’s of null values in the data.We will drop those column who contains more than 60% of missing values.

```
pmltrain <- subset(pmltrain[,-c(1:5)])
na_col <- function(x) ! sum(is.na(x))/length(x) > 0.6
pmltrain <- pmltrain %>% select(where(na_col))
head(pmltrain)
```

```
## # A tibble: 6 x 55
##   new_window num_window roll_belt pitch_belt yaw_belt total_accel_belt
##   <chr>           <dbl>     <dbl>      <dbl>    <dbl>            <dbl>
## 1 no                 11      1.42       8.07    -94.4                3
## 2 no                 12      1.45       8.06    -94.4                3
## 3 no                 12      1.42       8.09    -94.4                3
## 4 no                 12      1.42       8.13    -94.4                3
## 5 no                 12      1.42       8.2     -94.4                3
## 6 no                 12      1.45       8.2     -94.4                3
## # ... with 49 more variables: gyros_belt_x <dbl>, gyros_belt_y <dbl>,
## #   gyros_belt_z <dbl>, accel_belt_x <dbl>, accel_belt_y <dbl>,
## #   accel_belt_z <dbl>, magnet_belt_x <dbl>, magnet_belt_y <dbl>,
## #   magnet_belt_z <dbl>, roll_arm <dbl>, pitch_arm <dbl>, yaw_arm <dbl>,
## #   total_accel_arm <dbl>, gyros_arm_x <dbl>, gyros_arm_y <dbl>,
## #   gyros_arm_z <dbl>, accel_arm_x <dbl>, accel_arm_y <dbl>, accel_arm_z <dbl>,
## #   magnet_arm_x <dbl>, magnet_arm_y <dbl>, magnet_arm_z <dbl>, ...
```

Before starting, we will drop those column which are multicollinear with other.That can arise multicollinearity problem later so dropping those column will be appropriate in preprocessing step.

```
pmltrain_num <- pmltrain %>% select(where(is.numeric))#only numeric data
correlation <- cor(pmltrain_num)#correlation
diag(correlation) <- 0 #making diag entry zero as variable itself is perfectly correlated
correlation[upper.tri(correlation)] <- 0 #making correlation matrix lower triangular
any(abs(correlation) > 0.6)
```

```
## [1] TRUE
```

```
abs_thr <- function(datafrm , thr){    #function to get columns that are multicollinear
        col = c()
        for(i in 1:length(names(datafrm))){
                for(j in 1:i){
                        if (abs(datafrm[i,j]) > thr){
                                col = c(col,names(datafrm[i,j]))
                        }
                }
        }
        return(unique(col))
}
dt <- data.frame(correlation)
dt <- tibble(dt)
abs_thr(dt,0.7)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "accel_belt_y"         "accel_belt_x"
##  [7] "magnet_belt_y"        "gyros_arm_x"          "accel_belt_z"
## [10] "accel_arm_x"          "magnet_arm_x"         "accel_arm_z"
## [13] "magnet_arm_y"         "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [16] "pitch_dumbbell"       "roll_dumbbell"        "total_accel_dumbbell"
## [19] "yaw_dumbbell"         "gyros_belt_x"         "magnet_dumbbell_x"
## [22] "gyros_dumbbell_z"     "gyros_forearm_y"      "accel_forearm_y"
```

```
col <- abs_thr(dt,0.7)
pmltrain <- pmltrain %>% select(-col)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(col)` instead of `col` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
head(pmltrain)
```

```
## # A tibble: 6 x 31
##   new_window num_window gyros_belt_y gyros_belt_z magnet_belt_x magnet_belt_z
##   <chr>           <dbl>        <dbl>        <dbl>         <dbl>         <dbl>
## 1 no                 11            0        -0.02            -2          -305
## 2 no                 12            0        -0.02             0          -312
## 3 no                 12            0        -0.02            -4          -311
## 4 no                 12            0        -0.02            -2          -313
## 5 no                 12            0         0               -3          -309
## 6 no                 12            0         0               -1          -310
## # ... with 25 more variables: roll_arm <dbl>, pitch_arm <dbl>, yaw_arm <dbl>,
## #   total_accel_arm <dbl>, gyros_arm_y <dbl>, gyros_arm_z <dbl>,
## #   accel_arm_y <dbl>, magnet_arm_z <dbl>, accel_dumbbell_x <dbl>,
## #   accel_dumbbell_y <dbl>, accel_dumbbell_z <dbl>, magnet_dumbbell_y <dbl>,
## #   magnet_dumbbell_z <dbl>, roll_forearm <dbl>, pitch_forearm <dbl>,
## #   yaw_forearm <dbl>, total_accel_forearm <dbl>, gyros_forearm_x <dbl>,
## #   gyros_forearm_z <dbl>, accel_forearm_x <dbl>, accel_forearm_z <dbl>, ...
```

Here,*classe* variable is categorical,and clearly it is problem of classification.We will try to build a different classification model.We will use model which will going to perform better on our testing set.We will continue to use this model to predict or to classify validation data.
first of all we will use *Decision tree classifier* to predict the class.

```
pmltrain$new_window <- 1*(pmltrain$new_window == "yes")#one hot encode
model <- train(classe~.,data = pmltrain,method = 'rpart')
model
```

```
## CART
##
## 11776 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.04010441  0.4887630  0.33726153
##   0.04425724  0.4693836  0.30958566
##   0.05977100  0.3421625  0.09985985
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.04010441.
```

```
test$new_window <- 1*(test$new_window == "yes")#one hot encode
confusionMatrix(factor(test$classe),predict(model,test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1215   28   86    0    1
##          B  363  318  236    0    3
##          C  373   28  425    0    0
##          D  293   69  269    0  141
##          E  186   54  266    0  364
##
## Overall Statistics
##
##                Accuracy : 0.4922
##                  95% CI : (0.4778, 0.5065)
##     No Information Rate : 0.515
##     P-Value [Acc > NIR] : 0.9992
##
##                   Kappa : 0.3377
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5000   0.6398  0.33151       NA  0.71513
## Specificity            0.9497   0.8574  0.88329   0.8364  0.87978
## Pos Pred Value         0.9135   0.3457  0.51453       NA  0.41839
## Neg Pred Value         0.6414   0.9529  0.77980       NA  0.96232
## Prevalence             0.5150   0.1053  0.27173   0.0000  0.10788
## Detection Rate         0.2575   0.0674  0.09008   0.0000  0.07715
## Detection Prevalence   0.2819   0.1950  0.17507   0.1636  0.18440
## Balanced Accuracy      0.7249   0.7486  0.60740       NA  0.79745
```

Here,Model is only generating 48% accuracy…which is not good.
So we will now going to classify objects using Linear Discriminant Analysis.

```
model2 <- train(classe~.,data = pmltrain,method = "lda")
confusionMatrix(factor(test$classe),predict(model2,test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 912 148 100  80  90
##          B 165 476 105 101  73
##          C 128  43 518 100  37
##          D  58  85 139 405  85
##          E  64 149 121  94 442
##
## Overall Statistics
##
##                Accuracy : 0.5835
##                  95% CI : (0.5693, 0.5976)
##     No Information Rate : 0.2813
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4738
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6873   0.5283   0.5270  0.51923  0.60798
## Specificity            0.8767   0.8837   0.9175  0.90681  0.89276
## Pos Pred Value         0.6857   0.5174   0.6271  0.52461  0.50805
## Neg Pred Value         0.8775   0.8881   0.8805  0.90497  0.92594
## Prevalence             0.2813   0.1910   0.2084  0.16532  0.15409
## Detection Rate         0.1933   0.1009   0.1098  0.08584  0.09368
## Detection Prevalence   0.2819   0.1950   0.1751  0.16363  0.18440
## Balanced Accuracy      0.7820   0.7060   0.7222  0.71302  0.75037
```

Accuracy is now increases as compared to the tree classifier, but not enough.So now, we are going to classify our objects using *naive bayes classifier*.

```
model3 <- train(classe~.,data = pmltrain,method = "naive_bayes")
confusionMatrix(factor(test$classe),predict(model3,test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1118   32   74   96   10
##          B  107  596  145   59   13
##          C   25   46  695   60    0
##          D   38    4  150  525   55
##          E   31   86   50   74  629
##
## Overall Statistics
##
##                Accuracy : 0.7552
##                  95% CI : (0.7427, 0.7674)
##     No Information Rate : 0.2796
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6911
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8476   0.7801   0.6239   0.6450   0.8897
## Specificity            0.9376   0.9181   0.9637   0.9367   0.9399
## Pos Pred Value         0.8406   0.6478   0.8414   0.6801   0.7230
## Neg Pred Value         0.9407   0.9558   0.8923   0.9268   0.9797
## Prevalence             0.2796   0.1619   0.2361   0.1725   0.1499
## Detection Rate         0.2370   0.1263   0.1473   0.1113   0.1333
## Detection Prevalence   0.2819   0.1950   0.1751   0.1636   0.1844
## Balanced Accuracy      0.8926   0.8491   0.7938   0.7908   0.9148
```

Which is far good than other,generating about 76% accuracy.Also sensitivity (which is being right) is also far better for each class. This will going to be final model and we will use this to predict validation set. Now we will train model using knn(k nearest neighbour) classifier.

```
model4 <- train(classe~.,data = pmltrain,method = "knn")
confusionMatrix(factor(test$classe),predict(model4,test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1312    7    6    4    1
##          B   26  851   27   11    5
##          C    3   17  783   18    5
##          D    4    5   40  717    6
##          E    4   10   10   35  811
##
## Overall Statistics
##
##                Accuracy : 0.9483
##                  95% CI : (0.9416, 0.9544)
##     No Information Rate : 0.2859
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9346
##
##  Mcnemar's Test P-Value : 2.203e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9726   0.9562   0.9042   0.9134   0.9795
## Specificity            0.9947   0.9820   0.9888   0.9860   0.9848
## Pos Pred Value         0.9865   0.9250   0.9479   0.9288   0.9322
## Neg Pred Value         0.9891   0.9897   0.9787   0.9828   0.9956
## Prevalence             0.2859   0.1886   0.1836   0.1664   0.1755
## Detection Rate         0.2781   0.1804   0.1660   0.1520   0.1719
## Detection Prevalence   0.2819   0.1950   0.1751   0.1636   0.1844
## Balanced Accuracy      0.9836   0.9691   0.9465   0.9497   0.9822
```

We got an extraordinary 94% of accuracy and all class sensitivity nearly close to 1 (probability of being right).This will going to be our final model and we will use this to predict validation set.

```
pmlval$new_window <- 1*(pmlval$new_window == "yes")
pred <- predict(model4,pmlval)
```

```
pred
```

```
##  [1] B A C A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```