Practical_ML
DF
11/15/2022

Introduction

Using devices such as Jawbone Up, Nike Fuel Band, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These types 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.Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. It was made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg). My goal here is to predict the *“class”* with the help of other predictors. This project is a part of Coursera Practical Machine Learning Week 4 - Peer-graded Assignment: Prediction Assignment Writeup.

Data

Load the data

Let’s load the data. I have downloaded the data already on my local system. Please download the data from here : Training and Testing. And run this code on the same directory as the data.

```
dfTrain <- read.csv("pml-training.csv", stringsAsFactors = F,na.strings = c("","NA","#DIV/0!"))
dfTest <- read.csv("pml-testing.csv", stringsAsFactors = F,na.strings = c("","NA","#DIV/0!"))
dim(dfTrain); dim(dfTest)
```

[1] 19622 160
[1] 20 160

Let’s create a validation for model tuning:

```r
#for reproducability
set.seed(101)
inTrain <- createDataPartition(dfTrain$classe, p = 0.8, list = F)
dfVal <- dfTrain[-inTrain,]
dfTrain <- dfTrain[inTrain,]
dim(dfTrain); dim(dfVal)
```

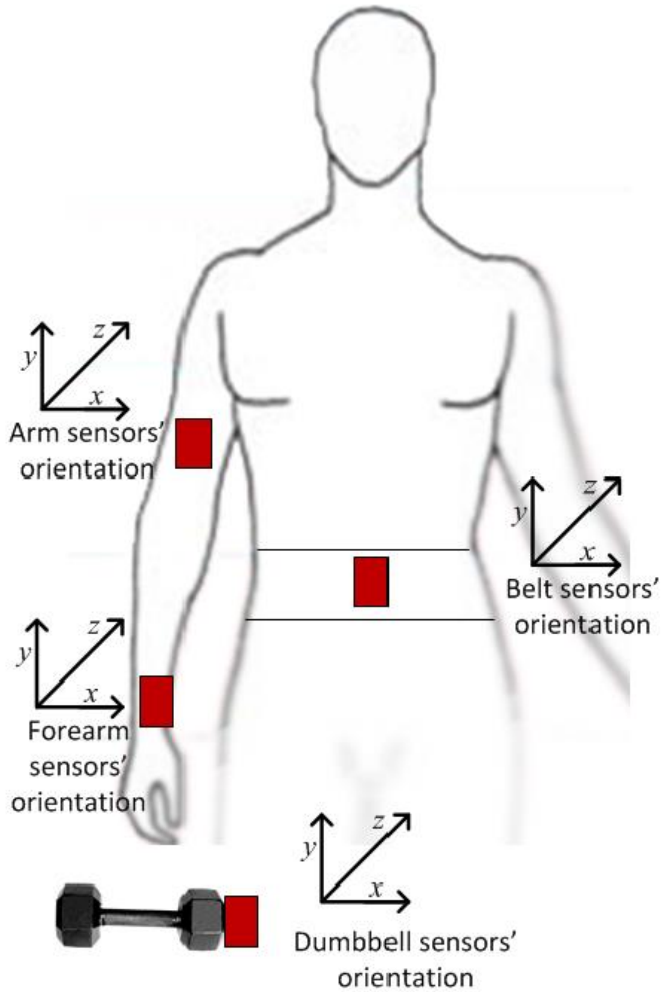
[1] 15699 160
[1] 3923 160
Now 3 partition of our data is ready, lets dive into analysis but 1st lets look at the proportion of different
“classe”:

```r
table(dfTrain$classe)/nrow(dfTrain)
```

A B C D E
0.2843493 0.1935155 0.1744060 0.1638958 0.1838334
From the above it is clear that there are not that much bias in the data in term of different “classe”.

Column overview

The data has 160 columns and for training data 15699 rows. Data was collected with the help of 4 sensors,
shown in the below diagram (diagram source).



The following symbol " " designates one of the set of sensors described in the text

Few Key points about the columns:

• *"X"* is primary key for the data.

• *"user_name"* is the id of the users. This may help us see interesting patterns for each activity for different users.

• *"classe"* is the target for prediction.

• Column - *3 to 7* is not necessary for this project. (5 features)

• As mentioned above there are 4 different sensors used for data collection. For each sensor there are 38 different features.

• Each sensor("belt","arm","forearm","dumbbell") has raw accelerometer, gyroscope and magnetometer readings for x, y and z axis. (4 sensor * 3 feature * 3 axis = 36 features)

• Each sensor(“belt”,“arm”,“forearm”,“dumbbell”) has Euler angles (roll, pitch and yaw) feature.(4 sensor
* 3 euler angles = 12 features)
• For the Euler angles of each of the four sensors eight features were calculated: mean, variance, standard
deviation, max, min, amplitude, kurtosis and skewness. (4 sensor * 3 feature * 8 measures = 96 features)
• For accelerometer we also have “total” and “variance of total” feature for the 4 sensors. But for
“belt”, “variance of total” is given as “var_total_accel_belt”, for the other sensors it is given as
(“var_accel_arm”,“var_accel_dumbbell”,“var_accel_forearm”). So I am considering the “belt” one
as a typo. (4 sensor * 2 feature = 8 features)
• There is another thing to note here. For “belt” Euler angles feature skewness is given as “skewness_
roll_belt”, “skewness_roll_belt.1” and “skewness_yaw_belt”. I am also considering “skewness_
roll_belt.1” as a typo and considering it as “skewness_pitch_belt”.

Missingness in the data

Let’s take a quick look at the missingness of the data. As the no of feature is large, its better to see them
by the 4 sensors:

Belt

For Belt sensor:

```
belt_miss <- sapply(select(dfTrain,names(dfTrain)[grepl("_belt",names(dfTrain))]),
                    function(x) sum(is.na(x)))
belt_miss
```

roll_belt pitch_belt yaw_belt
0 0 0
total_accel_belt kurtosis_roll_belt kurtosis_picth_belt
0 15396 15413
kurtosis_yaw_belt skewness_roll_belt skewness_roll_belt.1
15699 15395 15413
skewness_yaw_belt max_roll_belt max_picth_belt
15699 15388 15388
max_yaw_belt min_roll_belt min_pitch_belt
15396 15388 15388
min_yaw_belt amplitude_roll_belt amplitude_pitch_belt
15396 15388 15388
amplitude_yaw_belt var_total_accel_belt avg_roll_belt

15396 15388 15388
stddev_roll_belt var_roll_belt avg_pitch_belt
15388 15388 15388
stddev_pitch_belt var_pitch_belt avg_yaw_belt
15388 15388 15388
stddev_yaw_belt var_yaw_belt gyros_belt_x
15388 15388 0
gyros_belt_y gyros_belt_z accel_belt_x
0 0 0
accel_belt_y accel_belt_z magnet_belt_x
0 0 0
magnet_belt_y magnet_belt_z
0 0

Arm

For Arm sensor:

```
arm_miss <- sapply(select(dfTrain,names(dfTrain)[grepl("_arm",names(dfTrain))]),
                    function(x) sum(is.na(x)))
arm_miss
```

0 0 0 0
var_accel_arm avg_roll_arm stddev_roll_arm var_roll_arm
15388 15388 15388 15388
avg_pitch_arm stddev_pitch_arm var_pitch_arm avg_yaw_arm
15388 15388 15388 15388
stddev_yaw_arm var_yaw_arm gyros_arm_x gyros_arm_y
15388 15388 0 0
gyros_arm_z accel_arm_x accel_arm_y accel_arm_z
0 0 0 0
magnet_arm_x magnet_arm_y magnet_arm_z kurtosis_roll_arm
0 0 0 15446
kurtosis_picth_arm kurtosis_yaw_arm skewness_roll_arm skewness_pitch_arm
15448 15398 15445 15448
skewness_yaw_arm max_roll_arm max_picth_arm max_yaw_arm
15398 15388 15388 15388
min_roll_arm min_pitch_arm min_yaw_arm amplitude_roll_arm
15388 15388 15388 15388
amplitude_pitch_arm amplitude_yaw_arm
15388 15388

Forearm

For Forearm sensor:

```r
forearm_miss <- sapply(select(dfTrain,
                              names(dfTrain)[grepl("_forearm",names(dfTrain))]),
                       function(x) sum(is.na(x)))
forearm_miss
```

```
##          roll_forearm          pitch_forearm            yaw_forearm
##                     0                      0                      0
##   kurtosis_roll_forearm kurtosis_picth_forearm    kurtosis_yaw_forearm
##                 15448                  15449                  15699
##   skewness_roll_forearm skewness_pitch_forearm    skewness_yaw_forearm
##                 15447                  15449                  15699
##      max_roll_forearm       max_picth_forearm         max_yaw_forearm
##                 15388                  15388                  15448
##      min_roll_forearm       min_pitch_forearm         min_yaw_forearm
##                 15388                  15388                  15448
## amplitude_roll_forearm amplitude_pitch_forearm  amplitude_yaw_forearm
##                 15388                  15388                  15448
##     total_accel_forearm       var_accel_forearm        avg_roll_forearm
##                     0                  15388                  15388
##     stddev_roll_forearm        var_roll_forearm       avg_pitch_forearm
##                 15388                  15388                  15388
##    stddev_pitch_forearm       var_pitch_forearm         avg_yaw_forearm
##                 15388                  15388                  15388
##      stddev_yaw_forearm         var_yaw_forearm         gyros_forearm_x
##                 15388                  15388                      0
##         gyros_forearm_y         gyros_forearm_z         accel_forearm_x
##                     0                      0                      0
##         accel_forearm_y         accel_forearm_z        magnet_forearm_x
```

```
##                     0                      0                      0
##        magnet_forearm_y        magnet_forearm_z
##                     0                      0
```

Dumbbell

For Dumbbell sensor:

```r
dumbbell_miss <- sapply(select(dfTrain,
                              names(dfTrain)[grepl("_dumbbell",names(dfTrain))]),
                       function(x) sum(is.na(x)))
dumbbell_miss
```

```
##            roll_dumbbell           pitch_dumbbell             yaw_dumbbell 
##                        0                        0                        0 
##   kurtosis_roll_dumbbell  kurtosis_picth_dumbbell    kurtosis_yaw_dumbbell 
##                    15392                    15390                    15699 
##   skewness_roll_dumbbell  skewness_pitch_dumbbell    skewness_yaw_dumbbell 
##                    15391                    15389                    15699 
##        max_roll_dumbbell       max_picth_dumbbell         max_yaw_dumbbell 
##                    15388                    15388                    15392 
##        min_roll_dumbbell       min_pitch_dumbbell         min_yaw_dumbbell 
##                    15388                    15388                    15392 
##  amplitude_roll_dumbbell amplitude_pitch_dumbbell   amplitude_yaw_dumbbell 
##                    15388                    15388                    15392 
##     total_accel_dumbbell       var_accel_dumbbell        avg_roll_dumbbell 
##                        0                    15388                    15388 
##     stddev_roll_dumbbell        var_roll_dumbbell       avg_pitch_dumbbell 
##                    15388                    15388                    15388 
##    stddev_pitch_dumbbell       var_pitch_dumbbell         avg_yaw_dumbbell 
##                    15388                    15388                    15388 
##      stddev_yaw_dumbbell         var_yaw_dumbbell         gyros_dumbbell_x 
##                    15388                    15388                        0 
##         gyros_dumbbell_y         gyros_dumbbell_z         accel_dumbbell_x 
##                        0                        0                        0 
##         accel_dumbbell_y         accel_dumbbell_z        magnet_dumbbell_x 
##                        0                        0                        0 
##        magnet_dumbbell_y        magnet_dumbbell_z 
##                        0                        0
```

So it is very interesting to see that few of the features are over 90% missing, I would drop those columns for
further analysis. But the interesting thing is that all of those columns have same no of NA values.

```
column_2drop <- c(names(belt_miss[belt_miss != 0]),
                  names(arm_miss[arm_miss != 0]),
                  names(forearm_miss[forearm_miss != 0]),
                  names(dumbbell_miss[dumbbell_miss != 0]))
length(column_2drop)
```

[1] 100
So we can drop 100 column as they are mostly missing. After we drop these column there will be 52 predictors
left.

Analysis
Now lets get into analysis, first let's look at the correlation among the predictors.

```r
#dropping the cols
dfAnalize <- tbl_df(dfTrain %>%
                      select(-column_2drop,
                             -c(X,user_name, raw_timestamp_part_1,
                                raw_timestamp_part_2, cvtd_timestamp,
                                new_window,num_window)))
```


```r
#dropping the cols
dfAnalize <- tbl_df(dfTrain %>%
                      select(-column_2drop,
                             -c(X,user_name, raw_timestamp_part_1,
                                raw_timestamp_part_2, cvtd_timestamp,
                                new_window,num_window)))
```


```
## Warning: 'tbl_df()' was deprecated in dplyr 1.0.0.
## i Please use 'tibble::as_tibble()' instead.
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use 'all_of()' or 'any_of()' instead.
##   # Was:
##   data %>% select(column_2drop)
##
##   # Now:
##   data %>% select(all_of(column_2drop))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

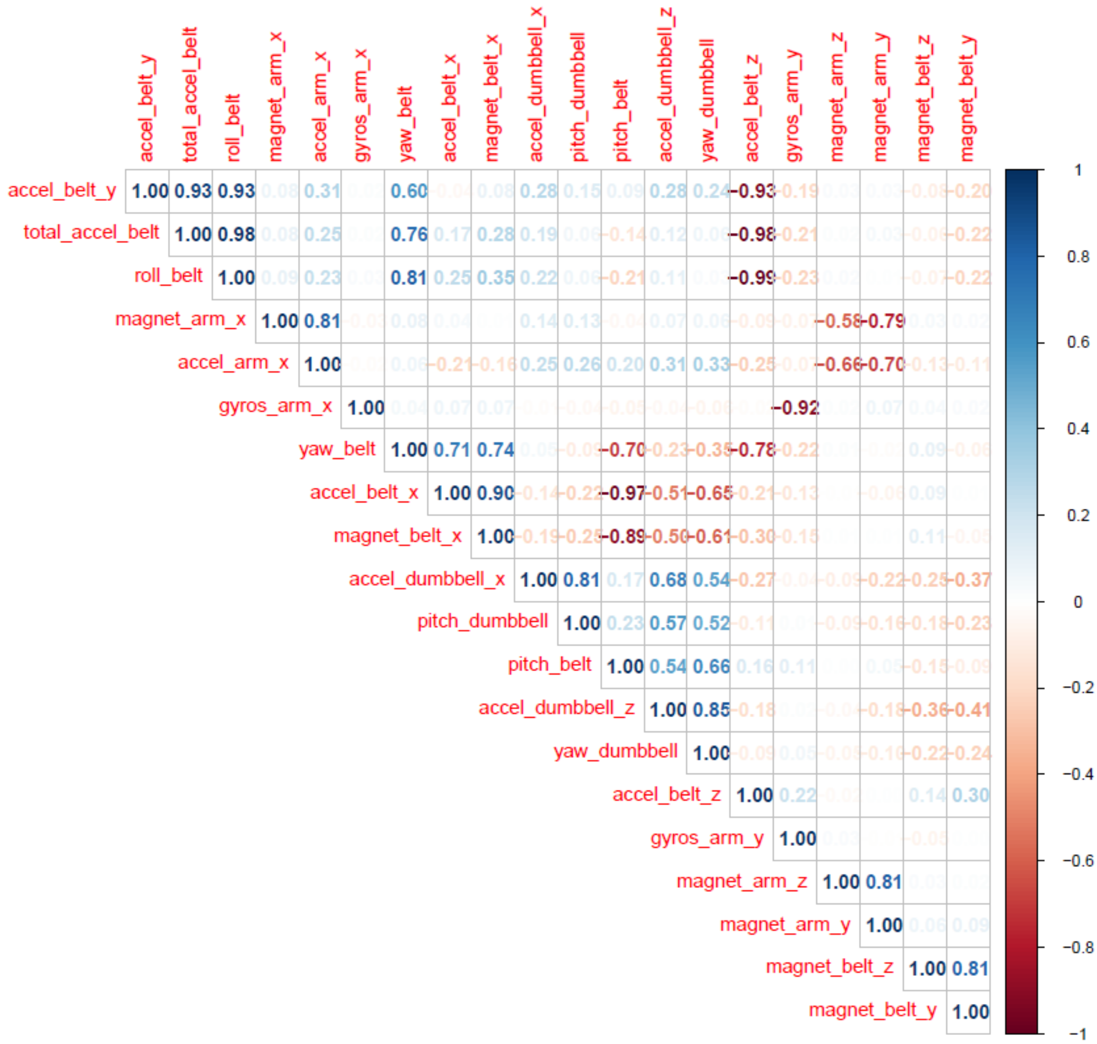
```
dfAnalize$classe <- as.factor(dfAnalize$classe)
dfAnalize[,1:52] <- lapply(dfAnalize[,1:52],as.numeric)
dim(dfAnalize)
```

```
## [1] 15699    53
```

Correlation among predictors

```
corr_col <- cor(select(dfAnalize, -classe))
diag(corr_col) <- 0
corr_col <- which(abs(corr_col)>0.8,arr.ind = T)
corr_col <- unique(row.names(corr_col))
corrplot(cor(select(dfAnalize,corr_col)),
         type="upper", order="hclust",method = "number")
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(corr_col)
##
##   # Now:
##   data %>% select(all_of(corr_col))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```



Here I have subsetted the data to show only the columns for which absolute correlation is higher than 0.8
with at least one other column. From Correlation plot it is clear that there is lot of columns that are highly
correlated. That might be an issue when we will be in modeling phase. Either we can drop those columns
or we can perform PCA(Principal Components Analysis). One important thing to note from this graph is
that high correlation is only seen between the same sensor i.e. “belt”,“arm”,“forearm” and “dumbbell”.

Correlation with the target

As the target is a categorical variable, we cannot check correlation with the other variables directly. But we

can use **correlationfunnel::correlate** to see the correlation with each level of"classe" and other features.
Lets go by them one by one.

```
# binarizing data
#correlationfunnel website: https://business-science.github.io/correlationfunnel/
corr_funl_df <- dfAnalize %>% binarize(n_bins = 4, thresh_infreq = 0.01)
```

classe___A

```
corr_a <- corr_funl_df %>% correlate(target = classe__A)
corr_a %>% plot_correlation_funnel(interactive = T,limits = c(-0.5,0.5))
```

For *classe__A* it seems that the "Arm and Forearm" sensors are more important.

- "accel_arm_x" is correlated with "magnet_arm_x", so wont consider.
- "gyros_arm_y" is correlated with "gyros_arm_x", so wont consider.
- So top 5 significant features for "classe__A" are - (magnet_arm_x, pitch_forearm , magnet_dumbbell_y, roll_forearm, gyros_dumbbell_y)

classe___B

```
corr_b <- corr_funl_df %>% correlate(target = classe__B)
corr_b %>% plot_correlation_funnel(interactive = T,limits = c(-0.5,0.5))
```

For *classe__B* it seems that the "Dumbbell and Belt" sensors are more important.

- So top 5 significant features for "classe__A" are - (magnet_dumbbell_y, magnet_dumbbell_x , roll_dumbbell , magnet_belt_y , accel_dumbbell_x)

classe___C

```
corr_c <- corr_funl_df %>% correlate(target = classe__C)
corr_c %>% plot_correlation_funnel(interactive = T,limits = c(-0.5,0.5))
```

For *classe__C* it seems that the "Dumbbell" sensors are more important.

- So top 5 significant features for "classe__A" are - (magnet_dumbbell_y, roll_dumbbell , accel_dumbbell_y , magnet_dumbbell_x, magnet_dumbbell_z) classe___D

```
corr_d <- corr_funl_df %>% correlate(target = classe__D)
corr_d %>% plot_correlation_funnel(interactive = T,limits = c(-0.5,0.5))
```

For *classe___D* it seems that the "Forearm, Arm and Dumbbell" sensors are more important.

- So top 5 significant features for "classe___A" are - (pitch_forearm , magnet_arm_y , magnet_forearm_x, accel_dumbbell_y, accel_forearm_x)

classe___E

```
corr_e <- corr_funl_df %>% correlate(target = classe__E)
corr_e %>% plot_correlation_funnel(interactive = T,limits = c(-0.5,0.5))
```

For *classe___E* it seems that the "Belt" sensors are more important.

- "total_accel_belt" is correlated with "roll_belt", so wont consider.
- "yaw_belt" is correlated with "roll_belt", so wont consider.
- "accel_belt_z" is correlated with "roll_belt", so wont consider.
- So top 5 significant features for "classe___A" are - (magnet_belt_y , magnet_belt_z , roll_belt, gyros_belt_z , magnet_dumbbell_y)

Let's make some plots

This document is already too long coursera assignment, so for this section I'll work on top 5 features for each class selected in the last section. So lets select only those columns.

```
#subseting dfAnalize
col_a <- c("magnet_arm_x", "pitch_forearm" , "magnet_dumbbell_y",
           "roll_forearm", "gyros_dumbbell_y")
col_b <- c("magnet_dumbbell_y", "magnet_dumbbell_x" , "roll_dumbbell" ,
           "magnet_belt_y" , "accel_dumbbell_x" )
col_c <- c("magnet_dumbbell_y", "roll_dumbbell" , "accel_dumbbell_y" ,
           "magnet_dumbbell_x", "magnet_dumbbell_z")
col_d <- c("pitch_forearm" , "magnet_arm_y" , "magnet_forearm_x",
           "accel_dumbbell_y", "accel_forearm_x")
col_e <- c("magnet_belt_y" , "magnet_belt_z" , "roll_belt",
           "gyros_belt_z" , "magnet_dumbbell_y")
final_cols <- character()
for(c in c(col_a,col_b,col_c,col_d,col_e)){
  final_cols <- union(final_cols, c)
}
dfAnalize2 <- dfAnalize %>% select(final_cols, classe)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(final_cols)
##
##   # Now:
##   data %>% select(all_of(final_cols))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
```

```
data.frame("arm" = sum(grepl("_arm",final_cols)),
           "forearm" = sum(grepl("_forearm",final_cols)),
           "belt" = sum(grepl("_belt",final_cols)),
           "dumbbell" = sum(grepl("_dumbbell",final_cols)))
```

```
##   arm forearm belt dumbbell
## 1   2       4    4        7
```

One interesting thing to note here is that the dumbbell sensor turned out to be the most important sensor among the 4. I would like to explore that in future works.

Pairs plot

```
my_dens <- function(data, mapping, ...) {
  ggplot(data = data, mapping=mapping) +
    geom_density(..., alpha = 0.3)+scale_fill_brewer(palette="Set2")
}
my_point <- function(data, mapping, ...) {
  ggplot(data = data, mapping=mapping) +
    geom_point(..., alpha = 0.1)+ scale_fill_brewer(palette="Set2")
}
ggpairs(dfAnalize2, columns = 1:5,aes(color = classe),
        lower = list(continuous = my_point),diag = list(continuous = my_dens))
```