Assignment 1

Assignment 1

My first assignment has three parts:

(a) a brief summary about "Veri Bilimi ve Endüstri Mühendisliği Üzerine Sohbetler - Baykal Hafızoğlu & Erdi Daşdemir":

Baykal Hafızoğlu is the guest of our lecturer Erdi Daşdemir's talk about data science and industrial engineering. In the beginning of the talk Hafızoğlu defines himself as an OR Scientist and Optimization Scientist. He indicates that his job is to write and solve OR models (mathematical models).

Hafızoğlu starts with OR and indicates that the classic definition of OR has changed a lot in the past few years. It was about optimization and statistics. However, it's more about artificial intelligence and machine learning nowadays. That's why Hafızoğlu prefers to call OR "analytics".

It is possible to divide analytics into four:

1-Descriptive analytics: Deals with defining the problem.

2-Diagnostic analytics: Deals with what the problem is and what the reasons are.

3-Predictive analytics: Deals with the future.

4-Prescriptive analytics: Deals with suggesting a solution.

He highlights that what students learn in undergrad is very precious and they should be aware of it. In the end of undergrad, students gain a good knowledge about analytics.

Later in his speech, he talks about the lessons he learned from his own experiences. Firstly, he says that "All projects start with a clear problem definition.". It means that even about the simplest thing he needs to deal with, the problem definition must be clear and concise. A good, compatible KPI (Key Performance Indicator) needs to be chosen in order to present the problem, interpret and explain it to the user. After that, a success criteria needs to be defined in order to understand whether we have achieved success or not.

Another lesson that he learned from his own experiences is the importance of deployment. He thinks delivering the model to the user is always better than keeping it in your computer. In this case, "How to deploy" and "where to deploy" are important questions.

Hafızoğlu highlights the importance of the user interface and early prototype. User satisfaction is the most important thing that's why the user needs to be understood well. Without keeping the user waiting, early prototype must be delivered for an early feedback. The user should not struggle to understand the user interface. If the user has to put effort, it shows that the user interface is not good enough.

It's very crucial for users to understand and own the model they will use. Because they wouldn't want to use models they don't understand. Therefore, we should be able to explain the analytical model we have established.

In the end, we need to show the solution's effect mathematically and refer to the problem and KPIs. Conflicting KPIs need to be considered because there can be a trade-off between KPIs. When one thing gets better, another thing can get worse.

(b) exploring statistical summaries with custom functions and loops:

```
data(mtcars)#mtcars dataset

custom_summary<-function(cars){
  mean_cars<-mean(cars)
  median_cars<-median(cars)
  standard_deviation_cars<-sd(cars)
  minimum_cars<-min(cars)
  maximum_cars<-max(cars)

  result<-c("mean"=mean_cars,
            "median"=median_cars,
            "standard deviation"=standard_deviation_cars,
            "minimum"=minimum_cars,
            "maximum"=maximum_cars)

  return(result)
}

cars<-c(1, 2, 3)#numeric vector

#writing a custom summary function
print("writing a custom summary function")
```

```
[1] "writing a custom summary function"
```

```
print("mpg")
```

```
[1] "mpg"
```

```
custom_summary(mtcars$mpg[cars])
```

```
            mean           median standard deviation            minimum
        21.60000         21.00000          1.03923         21.00000
         maximum
        22.80000
```

```
print("cyl")
```

```
[1] "cyl"
```

```
custom_summary(mtcars$cyl[cars])
```

```
            mean           median standard deviation            minimum
        5.333333         6.000000           1.154701         4.000000
         maximum
        6.000000
```

```
print("disp")
```

```
[1] "disp"
```

```
custom_summary(mtcars$disp[cars])
```

```
            mean           median standard deviation            minimum
      142.66667        160.00000          30.02221        108.00000
         maximum
      160.00000
```

```
print("hp")
```

```
[1] "hp"
```

```
custom_summary(mtcars$hp[cars])
```

```
            mean           median standard deviation            minimum
     104.333333       110.000000           9.814955         93.000000
         maximum
     110.000000
```

```
print("drat")
```

```
[1] "drat"
```

```
custom_summary(mtcars$drat[cars])
```

```
            mean           median standard deviation            minimum
     3.88333333       3.90000000         0.02886751      3.85000000
         maximum
     3.90000000
```

```
print("wt")
```

```
[1] "wt"
```

```
custom_summary(mtcars$wt[cars])
```

```
             mean            median standard deviation           minimum
        2.6050000         2.6200000         0.2778039         2.3200000
           maximum
        2.8750000
```

```
print("qsec")
```

```
[1] "qsec"
```

```
custom_summary(mtcars$qsec[cars])
```

```
             mean            median standard deviation           minimum
       17.363333         17.020000          1.115362         16.460000
           maximum
       18.610000
```

```
print("vs")
```

```
[1] "vs"
```

```
custom_summary(mtcars$vs[cars])
```

```
             mean            median standard deviation           minimum
        0.3333333         0.0000000         0.5773503         0.0000000
           maximum
        1.0000000
```

```
print("am")
```

```
[1] "am"
```

```
custom_summary(mtcars$am[cars])
```

```
             mean            median standard deviation           minimum
                1                 1                 0                 1
          maximum
                1
```

```
print("gear")
```

```
[1] "gear"
```

```
custom_summary(mtcars$gear[cars])
```

```
            mean              median standard deviation             minimum
               4                   4                   0                   4
         maximum
               4
```

```
print("carb")
```

```
[1] "carb"
```

```
custom_summary(mtcars$carb[cars])
```

```
            mean              median standard deviation             minimum
        3.000000            4.000000            1.732051            1.000000
         maximum
        4.000000
```

```
#applying the function using a loop
print("applying the function using a loop")
```

```
[1] "applying the function using a loop"
```

```
for(column_name in colnames(mtcars)){
  column_data<-mtcars[[column_name]][cars]
  print(column_name)
  print(custom_summary(column_data))
}
```

```
[1] "mpg"
            mean              median standard deviation             minimum
        21.60000            21.00000             1.03923            21.00000
         maximum
        22.80000
[1] "cyl"
            mean              median standard deviation             minimum
        5.333333            6.000000            1.154701            4.000000
         maximum
        6.000000
[1] "disp"
            mean              median standard deviation             minimum
      142.66667           160.00000            30.02221           108.00000
         maximum
      160.00000
[1] "hp"
            mean              median standard deviation             minimum
     104.333333          110.000000            9.814955           93.000000
```

```
          maximum
        110.000000
[1] "drat"
             mean            median standard deviation           minimum
        3.88333333        3.90000000        0.02886751        3.85000000
          maximum
        3.90000000
[1] "wt"
             mean            median standard deviation           minimum
         2.6050000         2.6200000         0.2778039         2.3200000
          maximum
         2.8750000
[1] "qsec"
             mean            median standard deviation           minimum
        17.363333         17.020000          1.115362         16.460000
          maximum
        18.610000
[1] "vs"
             mean            median standard deviation           minimum
        0.3333333         0.0000000         0.5773503         0.0000000
          maximum
        1.0000000
[1] "am"
             mean            median standard deviation           minimum
                1                 1                 0                 1
          maximum
                1
[1] "gear"
             mean            median standard deviation           minimum
                4                 4                 0                 4
          maximum
                4
[1] "carb"
             mean            median standard deviation           minimum
         3.000000          4.000000          1.732051          1.000000
          maximum
         4.000000
```

```r
#an alternative approach with apply
print("an alternative approach with apply")
```

```
[1] "an alternative approach with apply"
```

```r
apply(mtcars[cars, ], 2, custom_summary)
```

```
                     mpg      cyl      disp         hp       drat        wt
mean               21.60000 5.333333 142.66667 104.333333 3.88333333 2.6050000
median             21.00000 6.000000 160.00000 110.000000 3.90000000 2.6200000
standard deviation  1.03923 1.154701  30.02221   9.814955 0.02886751 0.2778039
minimum            21.00000 4.000000 108.00000  93.000000 3.85000000 2.3200000
maximum            22.80000 6.000000 160.00000 110.000000 3.90000000 2.8750000
```

```
                       qsec        vs am gear     carb
mean                17.363333 0.3333333  1    4 3.000000
median              17.020000 0.0000000  1    4 4.000000
standard deviation  1.115362 0.5773503  0    0 1.732051
minimum             16.460000 0.0000000  1    4 1.000000
maximum             18.610000 1.0000000  1    4 4.000000
```

(c) counting NA values and substituting with the number 660:

```
#install.packages("dslabs")
library(dslabs)
data(na_example)

#total count of NA values
sum(is.na(na_example))
```

```
[1] 145
```

```
#substituting the NA values with the number 660 and saving it as a new dataframe
no_nas<-ifelse(is.na(na_example), 660, na_example)

#total count of NA values in the new dataframe
sum(is.na(no_nas))
```

```
[1] 0
```

```
count<-0
#total count of the number 660 in the new dataframe
for(i in 1:1000)
  if(no_nas[i]==660)
    count<-count+1
count
```

```
[1] 145
```

Dataset with NA values

Dataset with the number 660

```
  [1]  2  1  3  2  1  3  1  4  3  2  2 NA  2  2  1  4 NA  1  1  2  1  2  2  1  2  5 NA  2  2  3  1
 [32]  2  4  1  1  1  4  5  2  3  4  1  2  4  1  1  2  1  5 NA NA NA  1  1  5  1  3  1 NA  4  4  7
 [63]  3  2 NA NA  1 NA  4  1  2  2  3  2  1  2  2  4  3  4  2  3  1  3  2  1  1  1  3  1 NA  3  1
 [94]  2  2  1  2  2  1  1  4  1  1  2  3  3  2  2  3  3  3  4  1  1  1  2 NA  4  3  4  3  1  2  1
[125] NA NA NA NA  1  5  1  2  1  3  5  3  2  2 NA NA NA NA  3  5  3  1  1  4  2  4  3  3 NA  2  3
[156]  2  6 NA  1  1  2  2  1  3  1  1  5 NA NA  2  4 NA  2  5  1  4  3  3 NA  4  3  1  4  1  1  3
[187]  1  1 NA NA  3  5  2  2  2  3  1  2  2  3  2  1 NA  2 NA  1 NA NA  2  1  1 NA  3 NA  1  2  2
[218]  1  3  2  2  1  1  2  3  1  1  1  4  3  4  2  2  1  4  1 NA  5  1  4 NA  3 NA NA  1  1  5  2
[249]  3  3  2  4 NA  3  2  5 NA  2  3  4  6  2  2  2 NA  2 NA  2 NA  3  3  2  2  4  3  1  4  2 NA
[280]  2  4 NA  6  2  3  1 NA  2  2 NA  1  1  3  2  3  3  1 NA  1  4  2  1  1  3  2  1  2  3  1 NA
[311]  2  3  3  2  1  2  3  5  5  1  2  3  3  1 NA NA  1  2  4 NA  2  1  1  1  3  2  1  1  3  4 NA
[342]  1  2  1  1  3  3 NA  1  1  3  5  3  2  3  4  1  4  3  1 NA  2  1  2  2  1  2  2  6  1  2  4
[373]  5 NA  3  4  2  1  1  4  2  1  1  1  1  2  1  4  4  1  3 NA  3  3 NA  2 NA  1  2  1  1  4  2
[404]  1  4  4 NA  1  2 NA  3  2  2  2  1  4  3  6  1  2  3  1  3  2  2  2  1  1  3  2  1  1  1  3
[435]  2  2 NA  4  4  4  1  1 NA  4  3 NA  1  3  1  3  2  4  2  2  2  3  2  1  4  3 NA  1  4  3  1
[466]  3  2 NA  3 NA  1  3  1  4  1  1  1  2  4  3  1  2  2  2  3  2  3  1  1 NA  3  2  1  1  2 NA
[497]  2  2  2  3  3  1  1  2 NA  1  2  1  1  3  3  1  3  1  1  1  1  1  2  5  1  1  2  2  1  1 NA
[528]  1  4  1  2  4  1  3  2 NA  1  1 NA  2  1  1  4  2  3  3  1  5  3  1  1  2 NA  1  1  3  1  3
[559]  2  4 NA  2  3  2  1  2  1  1  1  2  2  3  1  5  2 NA  2 NA  3  2  2  2  1  5  3  2  3  1 NA
[590]  3  1  2  2  2  1  2  2  4 NA  6  1  2 NA  1  1  2  2  3 NA  3  2  3  3  4  2 NA  2 NA  4 NA
[621]  1  1  2  2  3  1  1  1  3 NA  2  5 NA  7  1 NA  4  3  3  1 NA  1  1  1  1  3  2  4  2  2  3
[652] NA NA  1  4  3  2  2  2  3  2  4  2  2  4 NA NA NA  6  3  3  1  4  4  2  1 NA  1  6 NA  3  3
[683]  2  1  1  6 NA  1  5  1 NA  2  6  2 NA  4  1  3  1  2 NA  1  1  3  1  2  4  2  1  3  2  4  3
[714]  2  2  1  1  5  6  4  2  2  2  2  4 NA  1  2  2  2  2  4  5 NA NA NA  4  3  3  3  2  4  2  4
[745] NA NA NA NA  2  1 NA  2  4  3  2 NA  2  3  1  3  4 NA  1  2  1  2 NA  3  1  2  1  2  1  2  1
[776]  2  2  2  2  1  1  3  3  1  3  4  3 NA NA  4  2  3  2  1  3  2  4  2  2  3  1  2  4  3  3  4
[807] NA  1  4  2  1  1  1  3  1  5  2  2  4  2 NA  1  3  1  2 NA  1  2  1  2  1 NA  1  3  2  3  2
[838] NA  2  1  4  2 NA NA NA  2  4  2 NA NA  3  1 NA  5  5  2  2  2 NA  2  1  3  1  3  2  4  2  4
[869] NA  4  1  2  3  2  3  3  2  3  2  2  2  1  3  2  4  2 NA  3  3  2  2 NA NA  3  2  1  2  4  1
[900]  1  1  1  4  3  2 NA  3  2 NA  1 NA  3  2  1  1  1  2 NA  2  2  3  3  2 NA NA  4  5  2  2  2
[931]  1  2  3  1  3  3  4  3 NA  1  1  1 NA  4  3  5  1  1  2 NA  2  2  2  2  5  2  2  3  1  2  3
[962] NA  1  2 NA NA  2 NA  3  1  1  2  5  3  5  1  1  4 NA  2  1  3  1  1  2  4  3  3  3 NA  1  1
[993]  2  2  1  1  2  2 NA  2
```

```
  [1]   2   1   3   2   1   3   1   4   3   2   2 660   2   2   1   4 660   1   1   2   1   2   2   1
 [25]   2   5 660   2   2   3   1   2   4   1   1   1   4   5   2   3   4   1   2   4   1   1   2   1
 [49]   5 660 660 660   1   1   5   1   3   1 660   4   4   7   3   2 660 660   1 660   4   1   2   2
 [73]   3   2   1   2   2   4   3   4   2   3   1   3   2   1   1   1   3   1 660   3   1   2   2   1
 [97]   2   2   1   1   4   1   1   2   3   3   2   2   3   3   3   4   1   1   1   2 660   4   3   4
[121]   3   1   2   1 660 660 660 660   1   5   1   2   1   3   5   3   2   2 660 660 660 660   3   5
[145]   3   1   1   4   2   4   3   3 660   2   3   2   6 660   1   1   2   2   1   3   1   1   5 660
[169] 660   2   4 660   2   5   1   4   3   3 660   4   3   1   4   1   1   3   1   1 660 660   3   5
[193]   2   2   2   3   1   2   2   3   2   1 660   2 660   1 660 660   2   1   1 660   3 660   1   2
[217]   2   1   3   2   2   1   1   2   3   1   1   1   4   3   4   2   2   1   4   1 660   5   1   4
[241] 660   3 660 660   1   1   5   2   3   3   2   4 660   3   2   5 660   2   3   4   6   2   2   2
[265] 660   2 660   2 660   3   3   2   2   4   3   1   4   2 660   2   4 660   6   2   3   1 660   2
[289]   2 660   1   1   3   2   3   3   1 660   1   4   2   1   1   3   2   1   2   3   1 660   2   3
[313]   3   2   1   2   3   5   5   1   2   3   3   1 660 660   1   2   4 660   2   1   1   1   3   2
[337]   1   1   3   4 660   1   2   1   1   3   3 660   1   1   3   5   3   2   3   4   1   4   3   1
[361] 660   2   1   2   2   1   2   2   6   1   2   4   5 660   3   4   2   1   1   4   2   1   1   1
[385]   1   2   1   4   4   1   3 660   3   3 660   2 660   1   2   1   1   4   2   1   4   4 660   1
[409]   2 660   3   2   2   2   1   4   3   6   1   2   3   1   3   2   2   2   1   1   3   2   1   1
[433]   1   3   2   2 660   4   4   4   1   1 660   4   3 660   1   3   1   3   2   4   2   2   2   3
[457]   2   1   4   3 660   1   4   3   1   3   2 660   3 660   1   3   1   4   1   1   1   2   4   3
[481]   1   2   2   2   3   2   3   1   1 660   3   2   1   1   2 660   2   2   2   3   3   1   1   2
[505] 660   1   2   1   1   3   3   1   3   1   1   1   1   1   2   5   1   1   2   2   1   1 660   1
[529]   4   1   2   4   1   3   2 660   1   1 660   2   1   1   4   2   3   3   1   5   3   1   1   2
[553] 660   1   1   3   1   3   2   4 660   2   3   2   1   2   1   1   1   2   2   3   1   5   2 660
[577]   2 660   3   2   2   2   1   5   3   2   3   1 660   3   1   2   2   2   1   2   2   4 660   6
[601]   1   2 660   1   1   2   2   3 660   3   2   3   3   4   2 660   2 660   4 660   1   1   2   2
[625]   3   1   1   1   3 660   2   5 660   7   1 660   4   3   3   1 660   1   1   1   1   3   2   4
[649]   2   2   3 660 660   1   4   3   2   2   2   3   2   4   2   2   4 660 660 660   6   3   3   1
[673]   4   4   2   1 660   1   6 660   3   3   2   1   1   6 660   1   5   1 660   2   6   2 660   4
[697]   1   3   1   2 660   1   1   3   1   2   4   2   1   3   2   4   3   2   2   1   1   5   6   4
[721]   2   2   2   2   4 660   1   2   2   2   2   4   5 660 660 660   4   3   3   3   2   4   2   4
[745] 660 660 660 660   2   1 660   2   4   3   2 660   2   3   1   3   4 660   1   2   1   2 660   3
[769]   1   2   1   2   1   2   1   2   2   2   2   1   1   3   3   1   3   4   3 660 660   4   2   3
```

```
[793]   2   1   3   2   4   2   2   3   1   2   4   3   3   4 660   1   4   2   1   1   1   3   1   5
[817]   2   2   4   2 660   1   3   1   2 660   1   2   1   2   1 660   1   3   2   3   2 660   2   1
[841]   4   2 660 660 660   2   4   2 660 660   3   1 660   5   5   2   2   2 660   2   1   3   1   3
[865]   2   4   2   4 660   4   1   2   3   2   3   3   2   3   2   2   2   1   3   2   4   2 660   3
[889]   3   2   2 660 660   3   2   1   2   4   1   1   1   1   4   3   2 660   3   2 660   1 660   3
[913]   2   1   1   1   2 660   2   2   3   3   2 660 660   4   5   2   2   2   1   2   3   1   3   3
[937]   4   3 660   1   1   1 660   4   3   5   1   1   2 660   2   2   2   2   5   2   2   3   1   2
[961]   3 660   1   2 660 660   2 660   3   1   1   2   5   3   5   1   1   4 660   2   1   3   1   1
[985]   2   4   3   3   3 660   1   1   2   2   1   1   2   2 660   2
```