Data Analysis Practical Test E15: Sample Answer

TAESEUNG HAHN
https://github.com/tshahn/DataAnalPrac

Question No. 1

주어진 데이터 파일은 특정 철강사의 제품코드, 불량코드, 그리고 공정 과정에서 발생한 데이터를 담고 있다.
해당 데이터를 이용하여 다음 문제의 답변을 작성하시오.

제공 데이터 파일: E15Q1_data_raw.csv

- 1-25번 컬럼: Analog Data
- 26번 컬럼: 제품코드 (Binary Integer)
- 27번 컬럼: 불량코드 (Integer with range 1 to 7)

Load Package

```
pkgs = c("tidyverse", "caret", "stringr", "lubridate", "Amelia", "skimr", "ROCR",
         "NbClust", "corrplot", "ggfortify")
for (pkg in pkgs) if (!pkg %in% installed.packages()[,1]) install.packages(pkg)
invisible(lapply(pkgs, library, character.only=TRUE))
```

```
── Attaching packages ─────────────────────────────── tidyverse 1.3.0 ──

✓ ggplot23.3.0     ✓ purrr  0.3.4
✓ tibble 3.0.1     ✓ dplyr  0.8.5
✓ tidyr  1.0.2     ✓ stringr1.4.0
✓ readr  1.3.1     ✓ forcats0.5.0

── Conflicts ────────────────────────────────── tidyverse_conflicts() ──
✕ dplyr::filter()  masks stats::filter()
✕ dplyr::lag()     masks stats::lag()
```

① EDA를 실시하여 결과값을 제시하고, 상관분석을 시행하여 변수 선택 및 파생 변수 생성과정을 풀이하시오.

Load Data

```
data_raw = read_csv("./data/E15Q1_data_raw.csv")
problems(data_raw)
```

```
Parsed with column specification:
cols(
  .default = col_double()
)

See spec(...) for full column specifications.
```

A tibble: 0 × 4

row	col	expected	actual
<int>	<int>	<chr>	<chr>

Check and Transform DType

```
glimpse(data_raw)
```

```
Rows: 1,941
Columns: 27
$ X_Minimum              <dbl> 42, 645, 829, 853, 1289, 430, 413, 190, 330, 74…
$ X_Maximum              <dbl> 50, 651, 835, 860, 1306, 441, 446, 200, 343, 90…
$ Y_Minimum              <dbl> 270900, 2538079, 1553913, 369370, 498078, 10025…
$ Y_Maximum              <dbl> 270944, 2538108, 1553931, 369415, 498335, 10033…
$ Pixels_Areas           <dbl> 267, 108, 71, 176, 2409, 630, 9052, 132, 264, 1…
```

```
$ X_Perimeter           <dbl> 17, 10, 8, 13, 60, 20, 230, 11, 15, 46, 13, 42,…
$ Y_Perimeter           <dbl> 44, 30, 19, 45, 260, 87, 432, 20, 26, 167, 48, …
$ Sum_of_Luminosity     <dbl> 24220, 11397, 7972, 18996, 246930, 62357, 14819…
$ Minimum_of_Luminosity <dbl> 76, 84, 99, 99, 37, 64, 23, 124, 53, 53, 76, 97…
$ Maximum_of_Luminosity <dbl> 108, 123, 125, 126, 126, 127, 199, 172, 148, 14…
$ Length_of_Conveyer    <dbl> 1687, 1687, 1623, 1353, 1353, 1387, 1687, 1687,…
$ Steel_Plate_Thickness <dbl> 80, 80, 100, 290, 185, 40, 150, 150, 150, 150, …
$ Edges_Index           <dbl> 0.0498, 0.7647, 0.9710, 0.7287, 0.0695, 0.6200,…
$ Empty_Index           <dbl> 0.2415, 0.3793, 0.3426, 0.4413, 0.4486, 0.3417,…
$ Square_Index          <dbl> 0.1818, 0.2069, 0.3333, 0.1556, 0.0662, 0.1264,…
$ Outside_X_Index       <dbl> 0.0047, 0.0036, 0.0037, 0.0052, 0.0126, 0.0079,…
$ Edges_X_Index         <dbl> 0.4706, 0.6000, 0.7500, 0.5385, 0.2833, 0.5500,…
$ Edges_Y_Index         <dbl> 1.0000, 0.9667, 0.9474, 1.0000, 0.9885, 1.0000,…
$ Outside_Global_Index  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ LogOfAreas            <dbl> 2.4265, 2.0334, 1.8513, 2.2455, 3.3818, 2.7993,…
$ Log_X_Index           <dbl> 0.9031, 0.7782, 0.7782, 0.8451, 1.2305, 1.0414,…
$ Log_Y_Index           <dbl> 1.6435, 1.4624, 1.2553, 1.6532, 2.4099, 1.9395,…
$ Orientation_Index     <dbl> 0.8182, 0.7931, 0.6667, 0.8444, 0.9338, 0.8736,…
$ Luminosity_Index      <dbl> -0.2913, -0.1756, -0.1228, -0.1568, -0.1992, -0…
$ SigmoidOfAreas        <dbl> 0.5822, 0.2984, 0.2150, 0.5212, 1.0000, 0.9874,…
$ SteelType             <dbl> 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,…
$ Fault                 <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
```

```
summary(data_raw)
```

```
   X_Minimum        X_Maximum      Y_Minimum          Y_Maximum
 Min.   :   0.0   Min.   :   4   Min.   :    6712   Min.   :    6724
 1st Qu.:  51.0   1st Qu.: 192   1st Qu.:  471253   1st Qu.:  471281
 Median : 435.0   Median : 467   Median : 1204128   Median : 1204136
 Mean   : 571.1   Mean   : 618   Mean   : 1650685   Mean   : 1650739
 3rd Qu.:1053.0   3rd Qu.:1072   3rd Qu.: 2183073   3rd Qu.: 2183084
 Max.   :1705.0   Max.   :1713   Max.   :12987661   Max.   :12987692
  Pixels_Areas     X_Perimeter       Y_Perimeter       Sum_of_Luminosity
 Min.   :     2   Min.   :    2.0   Min.   :    1.00   Min.   :     250
 1st Qu.:    84   1st Qu.:   15.0   1st Qu.:   13.00   1st Qu.:    9522
 Median :   174   Median :   26.0   Median :   25.00   Median :   19202
 Mean   :  1894   Mean   :  111.9   Mean   :   82.97   Mean   :  206312
 3rd Qu.:   822   3rd Qu.:   84.0   3rd Qu.:   83.00   3rd Qu.:   83011
 Max.   :152655   Max.   :10449.0   Max.   :18152.00   Max.   :11591414
 Minimum_of_Luminosity Maximum_of_Luminosity Length_of_Conveyer
 Min.   :  0.00        Min.   : 37.0         Min.   :1227
 1st Qu.: 63.00        1st Qu.:124.0         1st Qu.:1358
 Median : 90.00        Median :127.0         Median :1364
 Mean   : 84.55        Mean   :130.2         Mean   :1459
 3rd Qu.:106.00        3rd Qu.:140.0         3rd Qu.:1650
 Max.   :203.00        Max.   :253.0         Max.   :1794
 Steel_Plate_Thickness  Edges_Index      Empty_Index      Square_Index
 Min.   : 40.00        Min.   :0.0000   Min.   :0.0000   Min.   :0.0083
 1st Qu.: 40.00        1st Qu.:0.0604   1st Qu.:0.3158   1st Qu.:0.3613
 Median : 70.00        Median :0.2273   Median :0.4121   Median :0.5556
 Mean   : 78.74        Mean   :0.3317   Mean   :0.4142   Mean   :0.5708
 3rd Qu.: 80.00        3rd Qu.:0.5738   3rd Qu.:0.5016   3rd Qu.:0.8182
 Max.   :300.00        Max.   :0.9952   Max.   :0.9439   Max.   :1.0000
 Outside_X_Index   Edges_X_Index    Edges_Y_Index    Outside_Global_Index
 Min.   :0.00150   Min.   :0.0144   Min.   :0.0484   Min.   :0.0000
 1st Qu.:0.00660   1st Qu.:0.4118   1st Qu.:0.5968   1st Qu.:0.0000
 Median :0.01010   Median :0.6364   Median :0.9474   Median :1.0000
 Mean   :0.03336   Mean   :0.6105   Mean   :0.8135   Mean   :0.5757
 3rd Qu.:0.02350   3rd Qu.:0.8000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :0.87590   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
   LogOfAreas     Log_X_Index     Log_Y_Index    Orientation_Index
 Min.   :0.301   Min.   :0.301   Min.   :0.000   Min.   :-0.99100
 1st Qu.:1.924   1st Qu.:1.000   1st Qu.:1.079   1st Qu.:-0.33330
 Median :2.241   Median :1.176   Median :1.322   Median : 0.09520
 Mean   :2.492   Mean   :1.336   Mean   :1.403   Mean   : 0.08329
 3rd Qu.:2.915   3rd Qu.:1.518   3rd Qu.:1.732   3rd Qu.: 0.51160
 Max.   :5.184   Max.   :3.074   Max.   :4.259   Max.   : 0.99170
 Luminosity_Index  SigmoidOfAreas     SteelType       Fault
 Min.   :-0.9989   Min.   :0.1190   Min.   :1.0   Min.   :1.000
 1st Qu.:-0.1950   1st Qu.:0.2482   1st Qu.:1.0   1st Qu.:3.000
 Median :-0.1330   Median :0.5063   Median :2.0   Median :6.000
 Mean   :-0.1313   Mean   :0.5854   Mean   :1.6   Mean   :4.841
 3rd Qu.:-0.0666   3rd Qu.:0.9998   3rd Qu.:2.0   3rd Qu.:7.000
 Max.   : 0.6421   Max.   :1.0000   Max.   :2.0   Max.   :7.000
```

```
data_raw %>% select(Outside_Global_Index) %>% table
```

```
.
   0  0.5    1
 778   91 1072
```

```
data = data_raw %>%
    mutate(SteelType = as.factor(SteelType),
           Fault = as.factor(Fault))
```

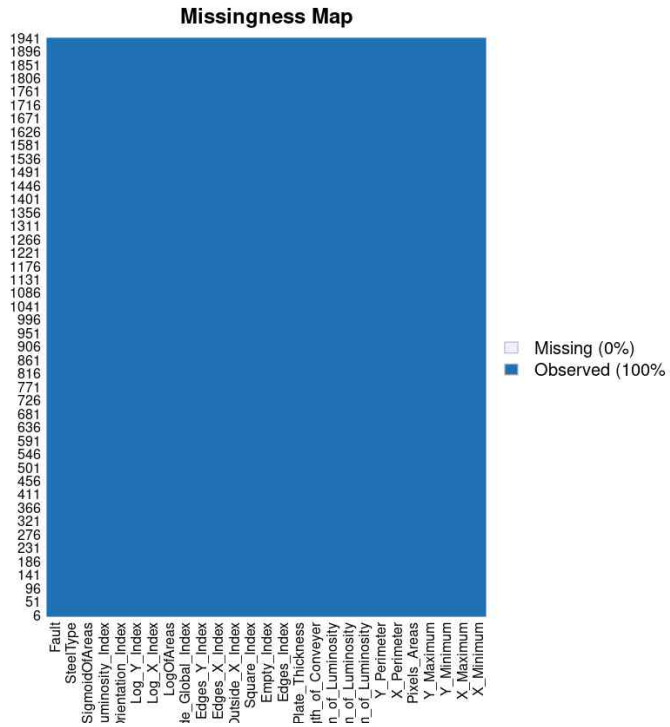
EDA: Check missing values

```
data %>%
    apply(2, function(arg) sum(is.na(arg))) %>%
    data.frame(MissingValueCnt=.)
```

A data.frame: 27 × 1

	MissingValueCnt
	<int>
X_Minimum	0
X_Maximum	0
Y_Minimum	0
Y_Maximum	0
Pixels_Areas	0
X_Perimeter	0
Y_Perimeter	0
Sum_of_Luminosity	0
Minimum_of_Luminosity	0
Maximum_of_Luminosity	0
Length_of_Conveyer	0
Steel_Plate_Thickness	0
Edges_Index	0
Empty_Index	0
Square_Index	0
Outside_X_Index	0
Edges_X_Index	0
Edges_Y_Index	0
Outside_Global_Index	0
LogOfAreas	0
Log_X_Index	0
Log_Y_Index	0
Orientation_Index	0
Luminosity_Index	0
SigmoidOfAreas	0
SteelType	0
Fault	0

```
missmap(data.frame(data))
```



EDA: Check Distributions

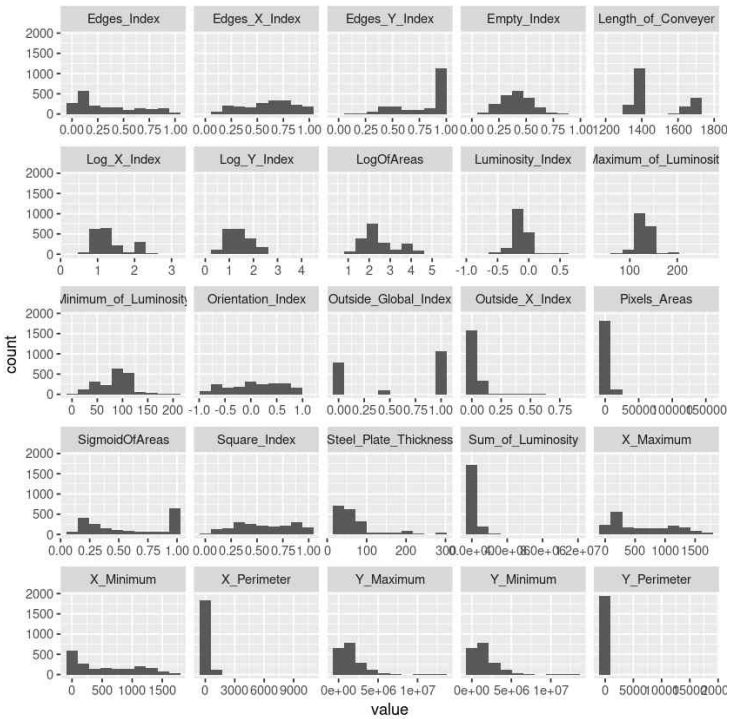
```
skim(data)
```

```
── Data Summary ────────────────────────
                           Values
Name                       data
Number of rows             1941
Number of columns          27
_____________________
Column type frequency:
  factor                   2
  numeric                  25
_____________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique
1 SteelType             0             1 FALSE          2
2 Fault                 0             1 FALSE          7
  top_counts
1 2: 1164, 1: 777
2 7: 673, 6: 402, 3: 391, 2: 190

── Variable type: numeric ─────────────────────────────────────────────────────
   skim_variable         n_missing complete_rate         mean           sd
 1 X_Minimum                     0             1     571.         521.
 2 X_Maximum                     0             1     618.         498.
 3 Y_Minimum                     0             1 1650685.     1774578.
 4 Y_Maximum                     0             1 1650739.     1774590.
 5 Pixels_Areas                  0             1    1894.        5168.
 6 X_Perimeter                   0             1     112.         301.
 7 Y_Perimeter                   0             1      83.0        426.
 8 Sum_of_Luminosity             0             1  206312.      512294.
 9 Minimum_of_Luminosity         0             1      84.5         32.1
10 Maximum_of_Luminosity         0             1     130.          18.7
11 Length_of_Conveyer            0             1    1459.         145.
12 Steel_Plate_Thickness         0             1      78.7         55.1
13 Edges_Index                   0             1       0.332        0.300
14 Empty_Index                   0             1       0.414        0.137
15 Square_Index                  0             1       0.571        0.271
16 Outside_X_Index               0             1       0.0334       0.0590
17 Edges_X_Index                 0             1       0.611        0.243
18 Edges_Y_Index                 0             1       0.813        0.234
19 Outside_Global_Index          0             1       0.576        0.482
20 LogOfAreas                    0             1       2.49         0.789
21 Log_X_Index                   0             1       1.34         0.482
22 Log_Y_Index                   0             1       1.40         0.454
```

```
23 Orientation_Index              0              1       0.0833       0.501
24 Luminosity_Index               0              1      -0.131        0.149
25 SigmoidOfAreas                 0              1       0.585        0.339
          p0          p25           p50          p75         p100 hist
 1    0           51            435          1053          1705
 2    4          192            467          1072          1713
 3 6712       471253        1204128       2183073      12987661
 4 6724       471281        1204136       2183084      12987692
 5    2           84            174           822        152655
 6    2           15             26            84         10449
 7    1           13             25            83         18152
 8  250         9522          19202         83011      11591414
 9    0           63             90           106           203
10   37          124            127           140           253
11 1227         1358           1364          1650          1794
12   40           40             70            80           300
13    0            0.0604         0.227         0.574         0.995
14    0            0.316          0.412         0.502         0.944
15    0.0083       0.361          0.556         0.818         1
16    0.0015       0.0066         0.0101        0.0235        0.876
17    0.0144       0.412          0.636         0.8           1
18    0.0484       0.597          0.947         1             1
19    0            0              1             1             1
20    0.301        1.92           2.24          2.91          5.18
21    0.301        1              1.18          1.52          3.07
22    0            1.08           1.32          1.73          4.26
23   -0.991       -0.333          0.0952        0.512         0.992
24   -0.999       -0.195         -0.133        -0.0666        0.642
25    0.119        0.248          0.506         1.00          1
```

```
g0 = ggplot(data = select_if(data, is.numeric) %>% pivot_longer(cols=everything()))
g11 = g0 +
    geom_histogram(aes(x=value), bins=10) +
    facet_wrap(~name, scales = "free_x")
```

```
print(g11)
```



Visualize Correlation with the Correlogram

```
data_cor = select_if(data, is.numeric)
```

```
cor_matx = cor(data_cor)
cor_matx[lower.tri(cor_matx, diag=TRUE)] = 0
```

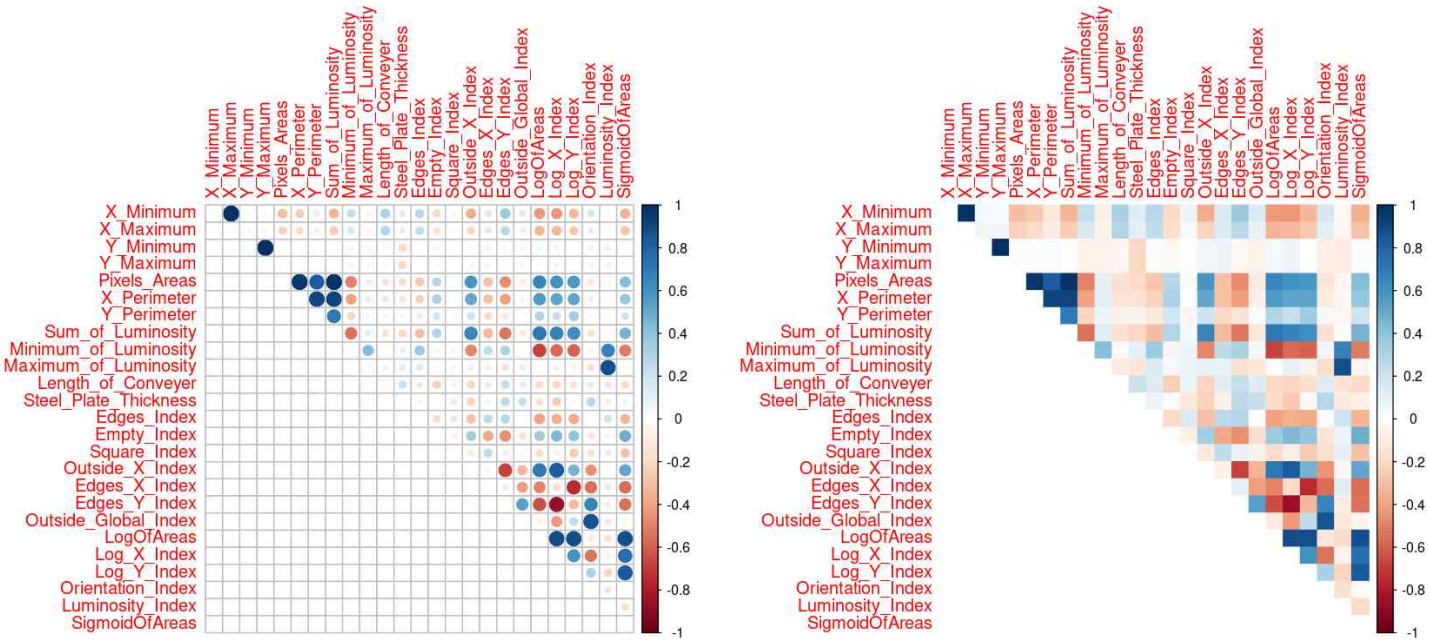
```
data.frame(cor_matx) %>%
    rownames_to_column %>%
```

```
    pivot_longer(cols=(-rowname)) %>%
    filter(value > 0.7)
```

A tibble: 18 × 3

rowname	name	value
<chr>	<chr>	<dbl>
X_Minimum	X_Maximum	0.9883135
Y_Minimum	Y_Maximum	1.0000000
Pixels_Areas	X_Perimeter	0.9666441
Pixels_Areas	Y_Perimeter	0.8271992
Pixels_Areas	Sum_of_Luminosity	0.9789516
X_Perimeter	Y_Perimeter	0.9124365
X_Perimeter	Sum_of_Luminosity	0.9129558
Y_Perimeter	Sum_of_Luminosity	0.7048757
Sum_of_Luminosity	LogOfAreas	0.7121281
Maximum_of_Luminosity	Luminosity_Index	0.8701602
Outside_X_Index	LogOfAreas	0.7108370
Outside_X_Index	Log_X_Index	0.8202233
Outside_Global_Index	Orientation_Index	0.8626697
LogOfAreas	Log_X_Index	0.8889194
LogOfAreas	Log_Y_Index	0.8829741
LogOfAreas	SigmoidOfAreas	0.8777683
Log_X_Index	SigmoidOfAreas	0.7573426
Log_Y_Index	SigmoidOfAreas	0.8381878

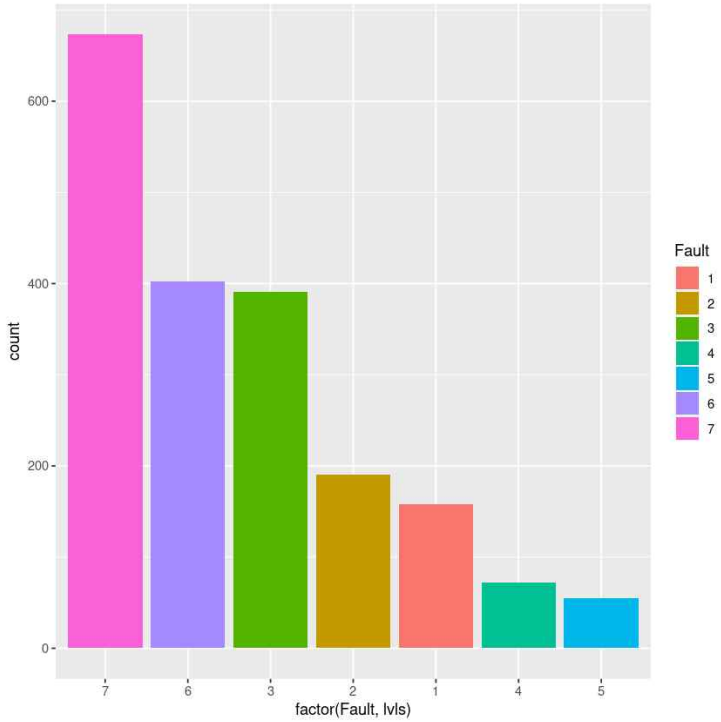
```
corrplot(cor_matx, method="circle")
corrplot(cor_matx, method="color")
```



EDA: Visualization Samples

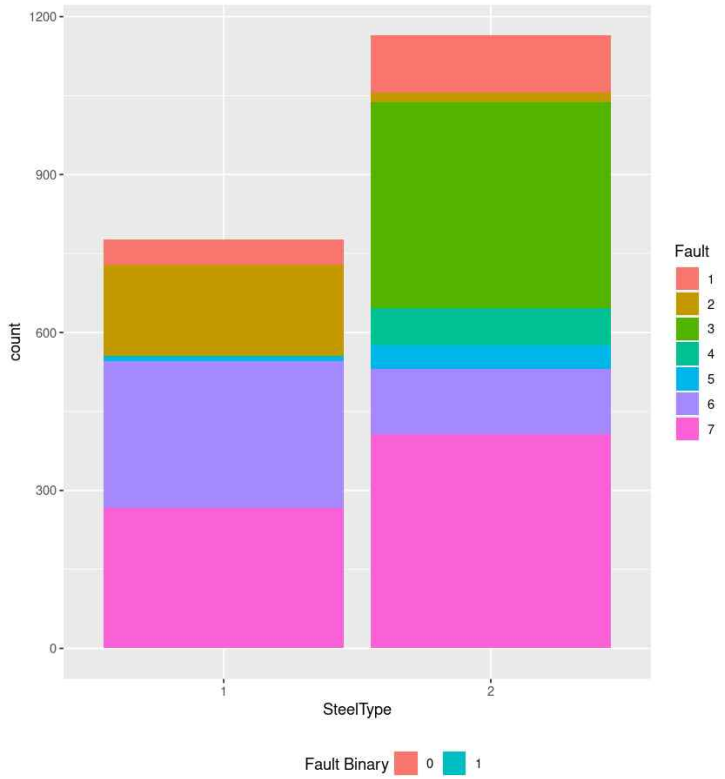
```
g0 = ggplot(data)
lvls = data %>% select(Fault) %>% table %>% data.frame %>% arrange(-Freq) %>% pull('.')
g11 = g0 + geom_bar(stat='count', aes(factor(Fault, lvls), fill=Fault))
```

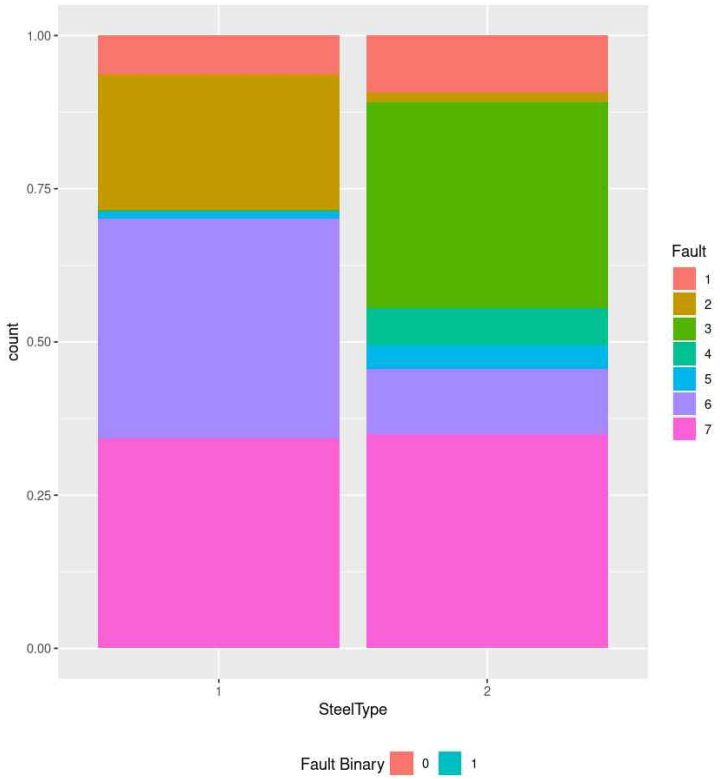
```
g11
```

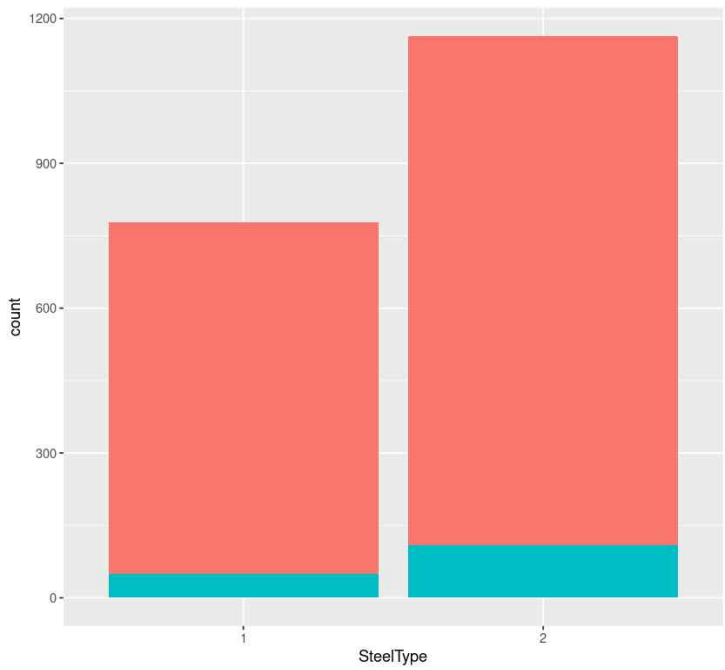

600
400
200
0
count
7
6
3
2
1
4
5
factor(Fault, lvls)
Fault
1
2
3
4
5
6
7

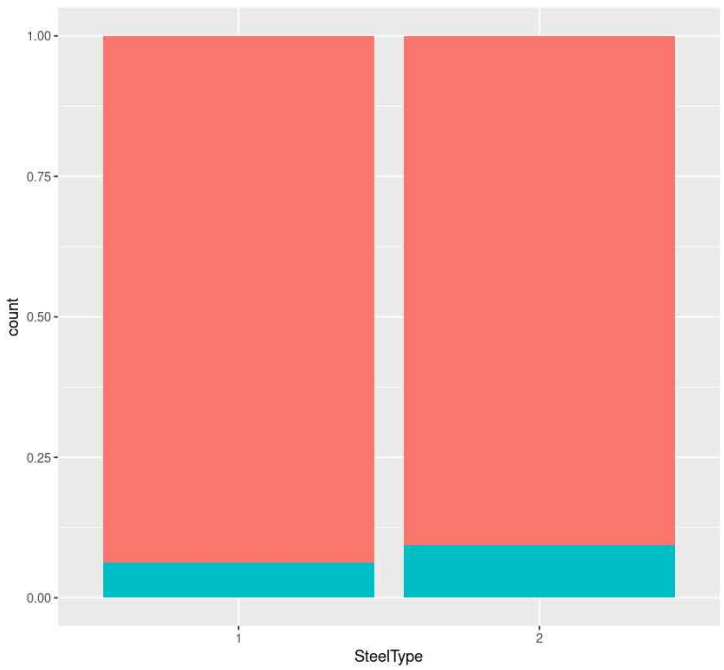
```
g21 = g0 + geom_bar(stat='count', aes(x=SteelType, fill=Fault))
g22 = g0 + geom_bar(stat='count', aes(x=SteelType, fill=Fault), position='fill')
g23 = g0 + geom_bar(stat='count', aes(x=SteelType, fill=as.factor(ifelse(Fault==1, 1, 0)))) +
    theme(legend.position='top') + scale_fill_discrete(name = "Fault Binary")
g24 = g0 + geom_bar(stat='count',
                    aes(x=SteelType, fill=as.factor(ifelse(Fault==1, 1, 0))), position='fill') +
    theme(legend.position='top') + scale_fill_discrete(name = "Fault Binary")
```

```
g21; g22; g23; g24
```



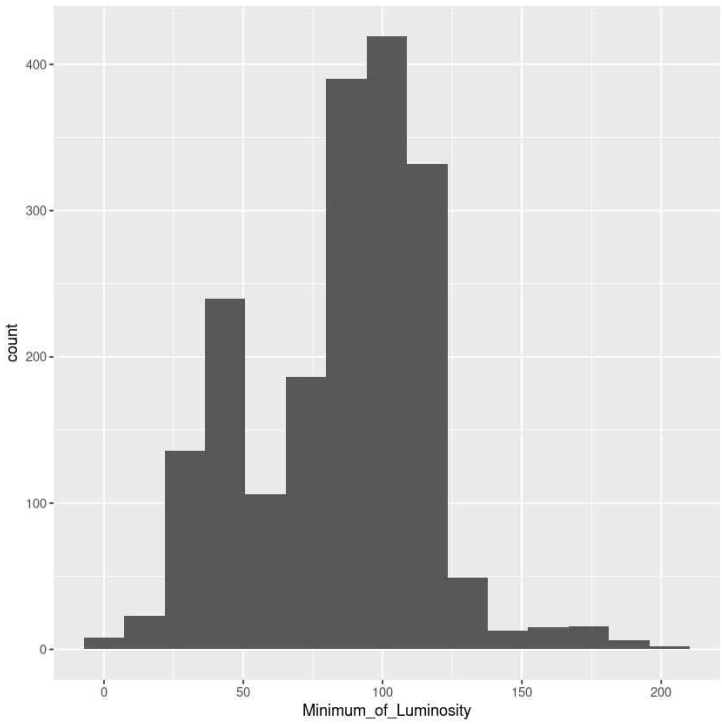


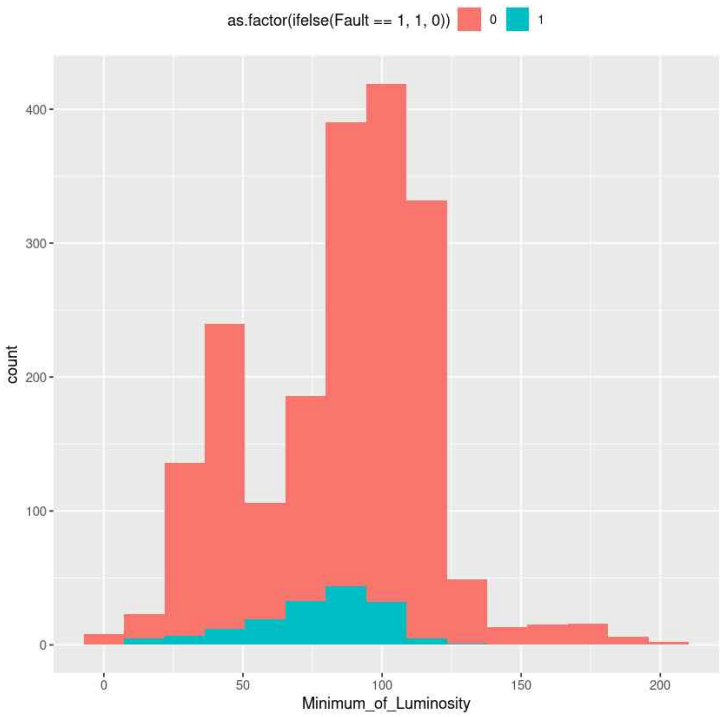


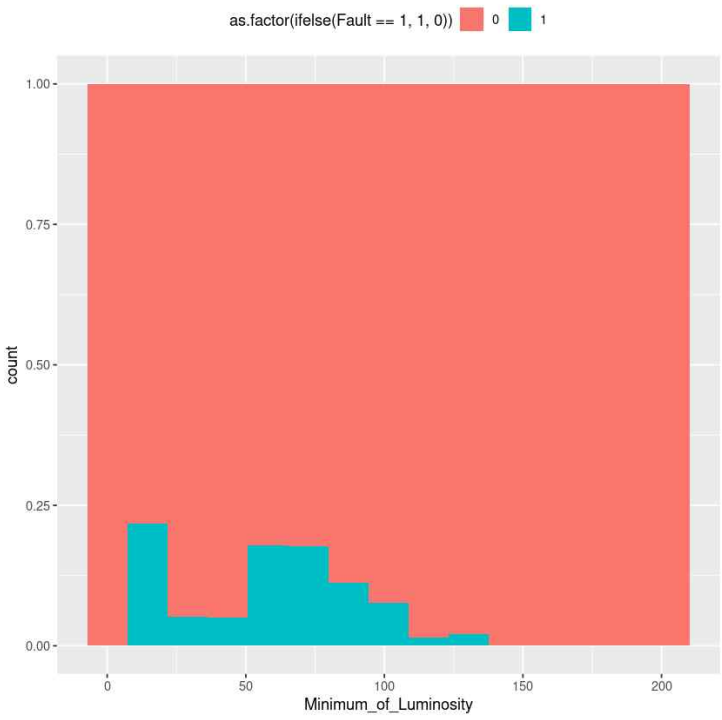


```
g31 = g0 + geom_histogram(aes(x=Minimum_of_Luminosity), bins=15)
g32 = g0 + geom_histogram(aes(x=Minimum_of_Luminosity,fill=Fault), bins=15)
g33 = g0 +
    geom_histogram(aes(x=Minimum_of_Luminosity, fill=as.factor(ifelse(Fault==1, 1, 0))), bins=15) +
    theme(legend.position = 'top')
g34 = g0 + geom_histogram(aes(x=Minimum_of_Luminosity, fill=Fault), position='fill', bins=15)
g35 = g0 +
    geom_histogram(aes(x=Minimum_of_Luminosity, fill=as.factor(ifelse(Fault==1, 1, 0))), position='fill', bi
    theme(legend.position = 'top')
```

```
g31; g33; g35
```







PCA: Variable Selection and Derived Variables

```
high_cor_cols_name = data.frame(cor_matx) %>%
    rownames_to_column() %>%
    pivot_longer(cols=(-rowname)) %>%
    filter(value > 0.7) %>%
    select(rowname, name) %>%
    pivot_longer(cols=c(rowname, name)) %>%
    pull(value) %>%
    unique

data_pca = select(data, all_of(high_cor_cols_name))
```

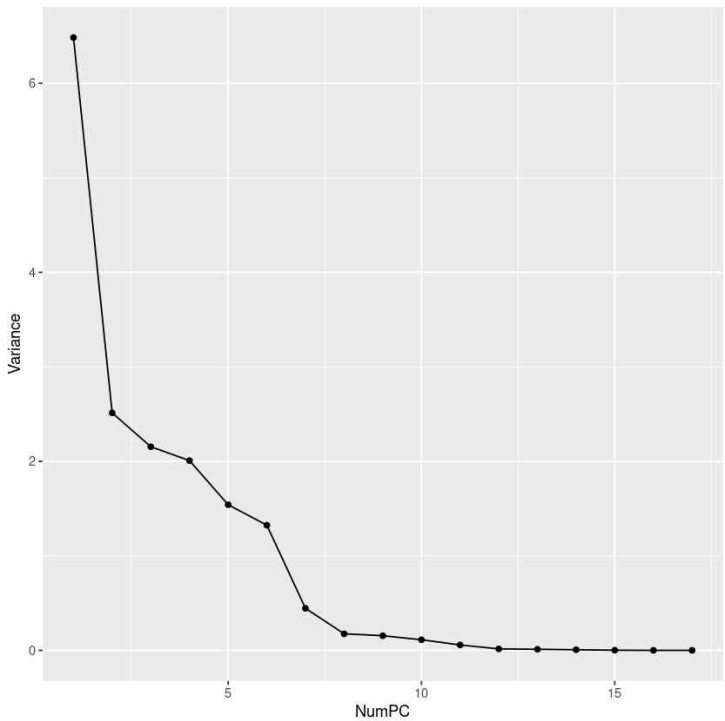
```
pca_rslt = prcomp(data_pca, center = TRUE, scale = TRUE)
```

PCA: Number of Principal Components

```
summary(pca_rslt)
```

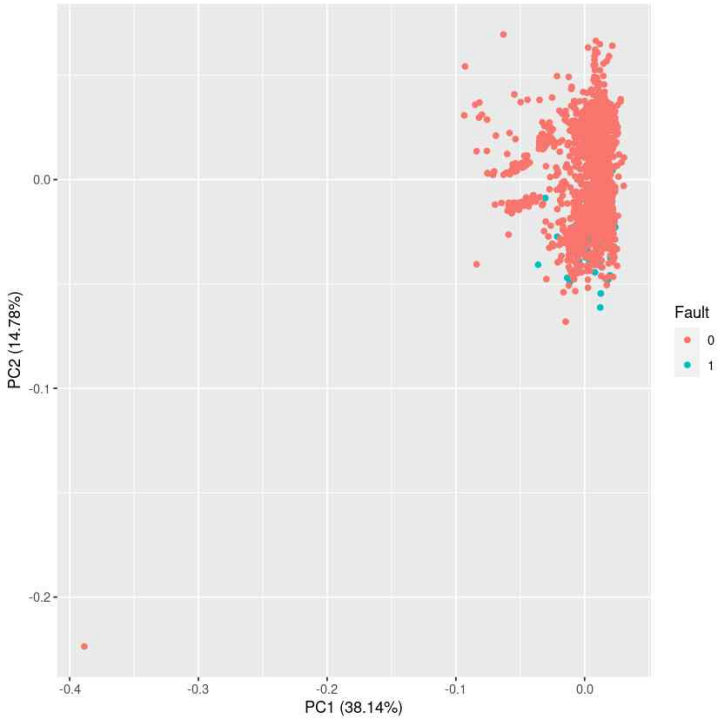
```
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6     PC7
Standard deviation     2.5463 1.5851 1.4677 1.4169 1.24126 1.15072 0.66666
Proportion of Variance 0.3814 0.1478 0.1267 0.1181 0.09063 0.07789 0.02614
Cumulative Proportion  0.3814 0.5292 0.6559 0.7740 0.86460 0.94249 0.96863
                          PC8     PC9    PC10    PC11   PC12    PC13    PC14
Standard deviation     0.4184 0.39355 0.33449 0.23894 0.1240 0.10539 0.08081
Proportion of Variance 0.0103 0.00911 0.00658 0.00336 0.0009 0.00065 0.00038
Cumulative Proportion  0.9789 0.98804 0.99462 0.99798 0.9989 0.99953 0.99992
                          PC15     PC16      PC17
Standard deviation     0.03735 0.004189 5.949e-06
Proportion of Variance 0.00008 0.000000 0.000e+00
Cumulative Proportion  1.00000 1.000000 1.000e+00
```

```
#plot(pca_rslt, type='l')
data_scree = data.frame(Variance = (pca_rslt$sdev)**2, NumPC = seq_along(pca_rslt$sdev))
ggplot(data_scree) + geom_line(aes(x = NumPC, y = Variance)) + geom_point(aes(x=NumPC, y=Variance))
```



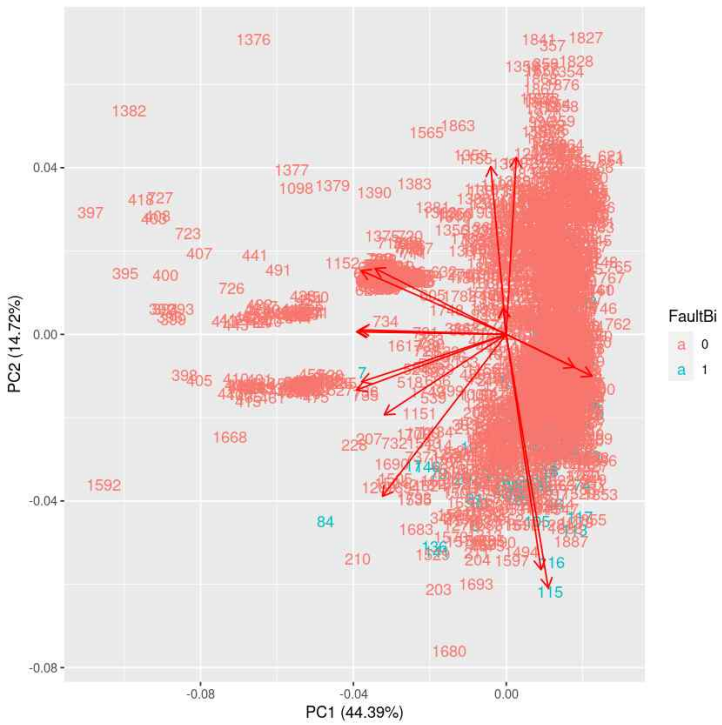
PCA: Visualization

```
autoplot(pca_rslt, data %>% mutate(Fault = as.factor(ifelse(Fault == 1, 1, 0))), colour = 'Fault')
```




```
data_pca2 = data_pca[-392, ]
```

```
# biplot(pca_rslt2, choice=c(1,2))
pca_rslt2 = prcomp(data_pca2, center = TRUE, scale = TRUE)
data_pca2_wFault = data.frame(data_pca2, FaultBi = as.factor(ifelse(data[-392, "Fault"] == 1, 1, 0)))
autoplot(pca_rslt2, data=data_pca2_wFault, x = 1, y = 2,
         colour = 'FaultBi', label = TRUE, shape = FALSE, loadings = TRUE)
```




```
data_aft_pca = data %>% select(-all_of(high_cor_cols_name)) %>% cbind(pca_rslt$x[,1:6])
```

```
str(data_aft_pca)
```

```
'data.frame':   1941 obs. of  16 variables:
 $ Minimum_of_Luminosity: num  76 84 99 99 37 64 23 124 53 53 ...
 $ Length_of_Conveyer   : num  1687 1687 1623 1353 1353 ...
 $ Steel_Plate_Thickness: num  80 80 100 290 185 40 150 150 150 150 ...
 $ Edges_Index          : num  0.0498 0.7647 0.971 0.7287 0.0695 ...
 $ Empty_Index          : num  0.241 0.379 0.343 0.441 0.449 ...
 $ Square_Index         : num  0.1818 0.2069 0.3333 0.1556 0.0662 ...
 $ Edges_X_Index        : num  0.471 0.6 0.75 0.538 0.283 ...
 $ Edges_Y_Index        : num  1 0.967 0.947 1 0.989 ...
```

```
 $ SteelType             : Factor w/ 2 levels "1","2": 1 1 1 2 2 2 2 2 2 2 ...
 $ Fault                 : Factor w/ 7 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ PC1                   : num  0.53 1.59 2.031 1.31 -0.557 ...
 $ PC2                   : num  -2.1 -1.51 -1.17 -1.88 -3.02 ...
 $ PC3                   : num  -0.704 0.352 -0.375 -0.997 -0.496 ...
 $ PC4                   : num  -1.826 0.2937 0.5681 -0.1565 -0.0524 ...
 $ PC5                   : num  -0.4346 -0.9605 -0.3631 0.0464 0.751 ...
 $ PC6                   : num  -1.328 -0.36 -0.218 0.442 2.216 ...
```

---

② 전체 데이터를 Train, Validataion, Test 용도로 분할하고 시각화하시오. (각 비율: 50%, 30%, 20%)

Split Data by Gnerated Index

```
set.seed(93)
folds = createFolds(data_aft_pca$Fault, k=10)
str(folds)
```

```
List of 10
 $ Fold01: int [1:194] 9 14 28 48 49 56 61 71 73 79 ...
 $ Fold02: int [1:195] 3 5 13 44 57 72 82 92 94 111 ...
 $ Fold03: int [1:196] 10 12 15 21 30 46 50 53 55 78 ...
 $ Fold04: int [1:193] 24 29 35 38 41 58 85 87 102 107 ...
 $ Fold05: int [1:194] 16 22 27 40 43 70 81 90 95 113 ...
 $ Fold06: int [1:194] 19 25 32 37 39 62 67 76 77 97 ...
 $ Fold07: int [1:194] 4 6 17 23 33 36 51 54 64 80 ...
 $ Fold08: int [1:193] 42 45 60 66 75 101 108 110 118 123 ...
 $ Fold09: int [1:195] 1 2 7 8 18 26 47 52 59 65 ...
 $ Fold10: int [1:193] 11 20 31 34 63 69 93 96 100 109 ...
```

```
idx_tr = folds[1:5] %>% unlist %>% unname
idx_vd = folds[6:8] %>% unlist %>% unname
idx_te = folds[9:10] %>% unlist %>% unname
```

```
data_tr = data_aft_pca[idx_tr,]  # data_tr = data[idx_tr,]
data_vd = data_aft_pca[idx_vd,]  # data_vd = data[idx_vd,]
data_te = data_aft_pca[idx_te,]  # data_te = data[idx_te,]
```

Visualize Proportion of Fault

```
data_grp = data_aft_pca %>%
    mutate(idx=seq(NROW(.)), Grp=ifelse(idx %in% idx_tr, "tr", ifelse(idx %in% idx_vd, "vd", "te"))) %>%
    select(-idx)
```

```
data_grp %>%
    select(Grp, Fault) %>%
    group_by(Grp) %>%
    count(Fault) %>%
    mutate(Grp_total = sum(n)) %>%
    ungroup() %>%
    mutate(Fault_prob=n/Grp_total) %>%
    arrange(Fault, Grp)
```

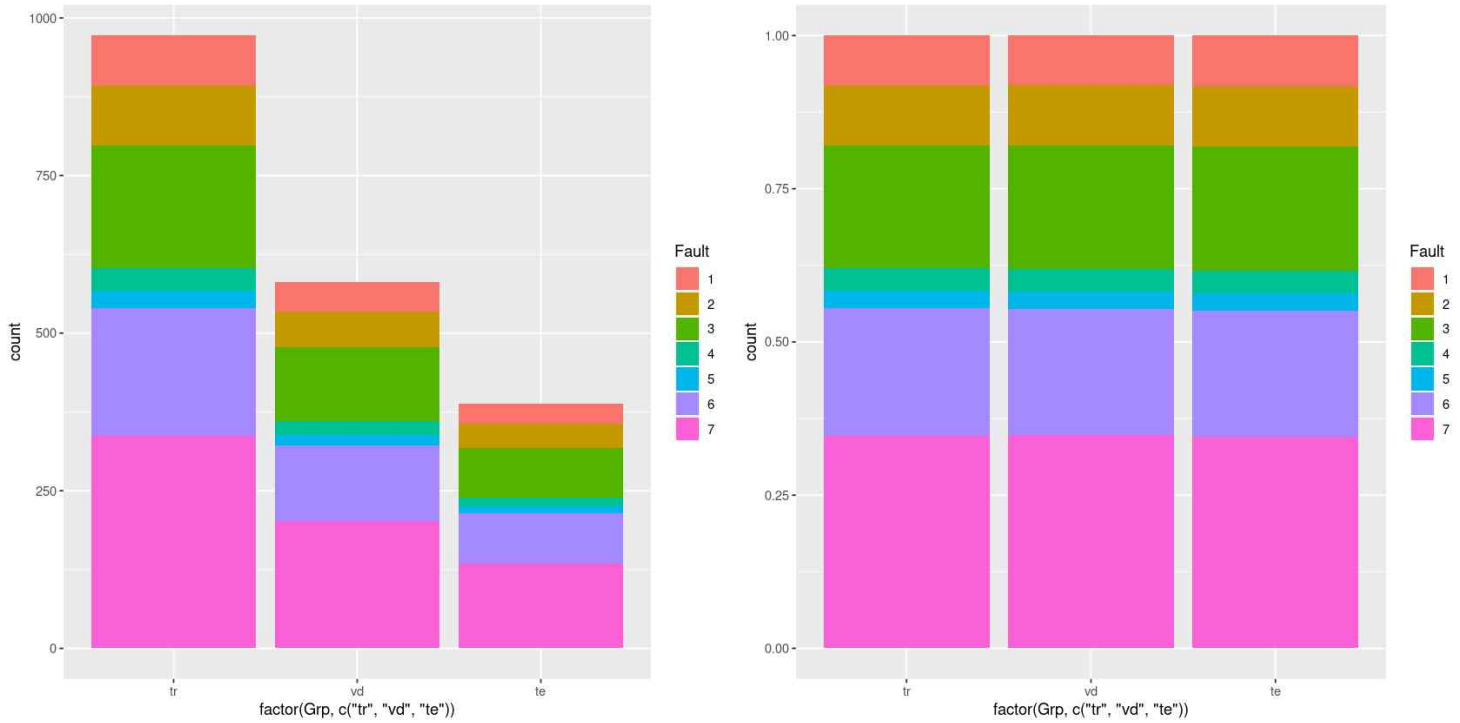
A tibble: 21 × 5

Grp	Fault	n	Grp_total	Fault_prob
<chr>	<fct>	<int>	<int>	<dbl>
te	1	32	388	0.08247423
tr	1	79	972	0.08127572
vd	1	47	581	0.08089501
te	2	38	388	0.09793814
tr	2	95	972	0.09773663
vd	2	57	581	0.09810671
te	3	79	388	0.20360825
tr	3	195	972	0.20061728
vd	3	117	581	0.20137694

Grp	Fault	n	Grp_total	Fault_prob
<chr>	<fct>	<int>	<int>	<dbl>
te	4	14	388	0.03608247
tr	4	36	972	0.03703704
vd	4	22	581	0.03786575
te	5	11	388	0.02835052
tr	5	28	972	0.02880658
vd	5	16	581	0.02753873
te	6	80	388	0.20618557
tr	6	202	972	0.20781893
vd	6	120	581	0.20654045
te	7	134	388	0.34536082
tr	7	337	972	0.34670782
vd	7	202	581	0.34767642

```
g0 = ggplot(data=data_grp)
g1 = g0 + geom_bar(aes(x=factor(Grp, c("tr", "vd", "te")), fill=Fault))
g2 = g0 + geom_bar(aes(x=factor(Grp, c("tr", "vd", "te")), fill=Fault), position="fill")
```

```
g1; g2
```



③ 불량코드 1에 대하여, Logistic Regression을 활용하여 이항분류 모형을 생성하라.

생성한 모형에 대한 최적의 Cut-Off Value를 선정 후, Confusion Matrix를 제시하라. (반드시 시각화와 통계량을 포함시킬 것)

```
data_tr_bin = data_tr %>%
    mutate(Fault = as.factor(ifelse(Fault == 1, 1, 0)))
target_vd = as.factor(ifelse(data_vd$Fault==1, 1, 0))
target_te = as.factor(ifelse(data_te$Fault==1, 1, 0))
```

```
fit_logit = train(
    form = Fault ~ .,
    data = data_tr_bin,
    trControl = trainControl(method="none"),
    method = "glmStepAIC",
    family = "binomial"
)
```

```
Start:  AIC=322.13
.outcome ~ Minimum_of_Luminosity + Length_of_Conveyer + Steel_Plate_Thickness +
    Edges_Index + Empty_Index + Square_Index + Edges_X_Index +
    Edges_Y_Index + SteelType2 + PC1 + PC2 + PC3 + PC4 + PC5 +
    PC6

                        Df Deviance    AIC
- PC3                    1   290.13 320.14
- PC2                    1   290.14 320.14
- PC4                    1   290.15 320.15
- Steel_Plate_Thickness  1   290.15 320.16
- Edges_Index            1   290.64 320.65
- Edges_X_Index          1   291.17 321.17
- PC6                    1   292.13 322.13
<none>                       290.13 322.13
- SteelType2             1   293.06 323.06
- PC1                    1   295.04 325.04
- Edges_Y_Index          1   296.09 326.09
- Empty_Index            1   297.83 327.83
- PC5                    1   300.50 330.50
- Square_Index           1   302.32 332.32
- Minimum_of_Luminosity  1   309.48 339.48
- Length_of_Conveyer     1   312.01 342.01

Step:  AIC=320.14
.outcome ~ Minimum_of_Luminosity + Length_of_Conveyer + Steel_Plate_Thickness +
    Edges_Index + Empty_Index + Square_Index + Edges_X_Index +
    Edges_Y_Index + SteelType2 + PC1 + PC2 + PC4 + PC5 + PC6



Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”


                        Df Deviance    AIC
- PC2                    1   290.14 318.14
- Steel_Plate_Thickness  1   290.16 318.16
- PC4                    1   290.21 318.21
- Edges_Index            1   290.65 318.65
- Edges_X_Index          1   291.22 319.22
<none>                       290.13 320.14
- SteelType2             1   293.06 321.06
- PC6                    1   293.55 321.55
- Edges_Y_Index          1   296.38 324.38
- PC1                    1   296.57 324.57
- Empty_Index            1   297.88 325.88
- PC5                    1   301.12 329.12
- Square_Index           1   303.47 331.47
- Minimum_of_Luminosity  1   309.69 337.69
- Length_of_Conveyer     1   312.21 340.21

Step:  AIC=318.14
.outcome ~ Minimum_of_Luminosity + Length_of_Conveyer + Steel_Plate_Thickness +
    Edges_Index + Empty_Index + Square_Index + Edges_X_Index +
    Edges_Y_Index + SteelType2 + PC1 + PC4 + PC5 + PC6

                        Df Deviance    AIC
- Steel_Plate_Thickness  1   290.17 316.17
- PC4                    1   290.22 316.22
- Edges_Index            1   290.67 316.67
- Edges_X_Index          1   291.62 317.62
<none>                       290.14 318.14
- SteelType2             1   293.15 319.15
- PC6                    1   293.61 319.61
- PC1                    1   296.83 322.83
- Empty_Index            1   297.89 323.89
- Edges_Y_Index          1   299.49 325.49
- PC5                    1   301.31 327.31
- Square_Index           1   306.83 332.83
- Length_of_Conveyer     1   312.28 338.28
- Minimum_of_Luminosity  1   316.96 342.96

Step:  AIC=316.17
.outcome ~ Minimum_of_Luminosity + Length_of_Conveyer + Edges_Index +
```

```
    Empty_Index + Square_Index + Edges_X_Index + Edges_Y_Index +
    SteelType2 + PC1 + PC4 + PC5 + PC6

                        Df Deviance    AIC
- PC4                    1   290.27 314.27
- Edges_Index            1   290.70 314.70
- Edges_X_Index          1   291.64 315.64
<none>                       290.17 316.17
- SteelType2             1   293.37 317.37
- PC6                    1   293.64 317.64
- PC1                    1   296.84 320.84
- Empty_Index            1   297.89 321.89
- Edges_Y_Index          1   299.51 323.51
- PC5                    1   301.35 325.35
- Square_Index           1   306.90 330.90
- Length_of_Conveyer     1   314.46 338.46
- Minimum_of_Luminosity  1   317.56 341.56

Step:  AIC=314.27
.outcome ~ Minimum_of_Luminosity + Length_of_Conveyer + Edges_Index +
    Empty_Index + Square_Index + Edges_X_Index + Edges_Y_Index +
    SteelType2 + PC1 + PC5 + PC6

                        Df Deviance    AIC
- Edges_Index            1   290.80 312.80
- Edges_X_Index          1   291.71 313.71
<none>                       290.27 314.27
- SteelType2             1   294.04 316.04
- PC6                    1   297.75 319.75
- Empty_Index            1   297.95 319.95
- Edges_Y_Index          1   302.06 324.06
- PC1                    1   303.74 325.74
- PC5                    1   305.12 327.12
- Square_Index           1   309.06 331.06
- Length_of_Conveyer     1   315.10 337.10
- Minimum_of_Luminosity  1   320.01 342.01

Step:  AIC=312.8
.outcome ~ Minimum_of_Luminosity + Length_of_Conveyer + Empty_Index +
    Square_Index + Edges_X_Index + Edges_Y_Index + SteelType2 +
    PC1 + PC5 + PC6

                        Df Deviance    AIC
- Edges_X_Index          1   292.09 312.09
<none>                       290.80 312.80
- SteelType2             1   294.60 314.60
- PC6                    1   297.91 317.91
- Empty_Index            1   298.30 318.30
- Edges_Y_Index          1   302.76 322.76
- PC1                    1   304.78 324.78
- PC5                    1   305.21 325.21
- Square_Index           1   313.06 333.06
- Length_of_Conveyer     1   315.58 335.58
- Minimum_of_Luminosity  1   322.36 342.36

Step:  AIC=312.09
.outcome ~ Minimum_of_Luminosity + Length_of_Conveyer + Empty_Index +
    Square_Index + Edges_Y_Index + SteelType2 + PC1 + PC5 + PC6

                        Df Deviance    AIC
<none>                       292.09 312.09
- SteelType2             1   295.37 313.37
- Empty_Index            1   298.37 316.37
- PC6                    1   301.42 319.42
- PC1                    1   305.13 323.13
- PC5                    1   306.10 324.10
- Edges_Y_Index          1   314.63 332.63
- Length_of_Conveyer     1   315.93 333.93
- Minimum_of_Luminosity  1   323.50 341.50
- Square_Index           1   350.22 368.22
```

```
summary(fit_logit)
```

```
Call:
NULL
```

```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.08238  -0.25724  -0.06165  -0.00530   3.04848

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -15.382215   5.410210  -2.843 0.004467 **
Minimum_of_Luminosity  -0.045338   0.008804  -5.149 2.61e-07 ***
Length_of_Conveyer      0.005164   0.001101   4.690 2.73e-06 ***
Empty_Index            -4.056392   1.644133  -2.467 0.013618 *
Square_Index           -5.228820   0.779536  -6.708 1.98e-11 ***
Edges_Y_Index          11.868189   5.124318   2.316 0.020555 *
SteelType2              0.667904   0.372309   1.794 0.072821 .
PC1                     0.877803   0.253854   3.458 0.000544 ***
PC5                    -0.584472   0.159199  -3.671 0.000241 ***
PC6                     0.475441   0.160225   2.967 0.003004 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 547.96  on 971  degrees of freedom
Residual deviance: 292.09  on 962  degrees of freedom
AIC: 312.09

Number of Fisher Scoring iterations: 10
```

```
pred_logit = predict(fit_logit, data_vd, type="prob")[["1"]]

co_range = seq(0, 1, 0.01)
pred_matx_logit = as.matrix(pred_logit) %*% rep(1, length(co_range))
co_matx_logit = rep(1, NROW(pred_logit)) %*% t(co_range)

pred_class_by_cutoff = as.data.frame(pred_matx_logit > co_matx_logit) %>%
    mutate_all(~as.factor(as.numeric(.)))
colnames(pred_class_by_cutoff) = co_range
```

```
cnfm = list()
crit = list()
for (i in seq_along(pred_class_by_cutoff)) {
    cm = confusionMatrix(pred_class_by_cutoff[,i], target_vd)
    cnfm[[i]] = cm
    crit[[i]] = round(cm$byClass, 4)
}
logit_eval_by_cutoff = as_tibble(cbind(co_range, (t(data.frame(crit)))))
```

```
(co_metric_ba = logit_eval_by_cutoff %>% mutate(SS = Sensitivity + Specificity - 1) %>%
    arrange(-SS)) %>% head()
(co_metric_f1 = logit_eval_by_cutoff %>% arrange(-F1)) %>% head()
```

A tibble: 6 × 13

co_range	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy	SS
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.06	0.8127	0.9149	0.9909	0.3007	0.9909	0.8127	0.8930	0.9191	0.7470	0.7539	0.8638	0.7276
0.14	0.8839	0.8298	0.9833	0.3861	0.9833	0.8839	0.9310	0.9191	0.8124	0.8262	0.8568	0.7137
0.08	0.8390	0.8723	0.9868	0.3228	0.9868	0.8390	0.9069	0.9191	0.7711	0.7814	0.8556	0.7113
0.13	0.8801	0.8298	0.9833	0.3786	0.9833	0.8801	0.9289	0.9191	0.8090	0.8227	0.8550	0.7099
0.05	0.7940	0.9149	0.9907	0.2810	0.9907	0.7940	0.8815	0.9191	0.7298	0.7367	0.8545	0.7089
0.16	0.8970	0.8085	0.9816	0.4086	0.9816	0.8970	0.9374	0.9191	0.8244	0.8399	0.8528	0.7055

A tibble: 6 × 12

co_range	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>

co_range	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.38	0.9700	0.5532	0.9610	0.6190	0.9610	0.9700	0.9655	0.9191	0.8916	0.9277	0.7616
0.39	0.9700	0.5532	0.9610	0.6190	0.9610	0.9700	0.9655	0.9191	0.8916	0.9277	0.7616
0.44	0.9831	0.3830	0.9477	0.6667	0.9477	0.9831	0.9651	0.9191	0.9036	0.9535	0.6831
0.45	0.9831	0.3617	0.9459	0.6538	0.9459	0.9831	0.9642	0.9191	0.9036	0.9552	0.6724
0.43	0.9775	0.4255	0.9508	0.6250	0.9508	0.9775	0.9640	0.9191	0.8985	0.9449	0.7015
0.40	0.9719	0.4894	0.9558	0.6053	0.9558	0.9719	0.9638	0.9191	0.8933	0.9346	0.7306

```
co_ba = co_metric_ba$co_range[1]
co_f1 = co_metric_f1$co_range[1]
```

(LOGISTIC) EVALUATE WITH TEST DATASET

```
te_pred_logit = predict(fit_logit, data_te, type="prob")[,2]
te_pred_logit_bin = as.factor(as.numeric(te_pred_logit > co_f1))
```

```
confusionMatrix(te_pred_logit_bin, target_te, mode = "prec_recall", positive="1") # or mode = "everything"
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 347  16
         1   9  16

               Accuracy : 0.9356
                 95% CI : (0.9064, 0.9579)
    No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.1125

                  Kappa : 0.5272

 Mcnemar's Test P-Value : 0.2301

              Precision : 0.64000
                 Recall : 0.50000
                     F1 : 0.56140
            Prevalence : 0.08247
        Detection Rate : 0.04124
   Detection Prevalence : 0.06443
      Balanced Accuracy : 0.73736

       'Positive' Class : 1
```

```
confusionMatrix(te_pred_logit_bin, target_te)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 347  16
         1   9  16

               Accuracy : 0.9356
                 95% CI : (0.9064, 0.9579)
    No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.1125

                  Kappa : 0.5272

 Mcnemar's Test P-Value : 0.2301

            Sensitivity : 0.9747
            Specificity : 0.5000
```

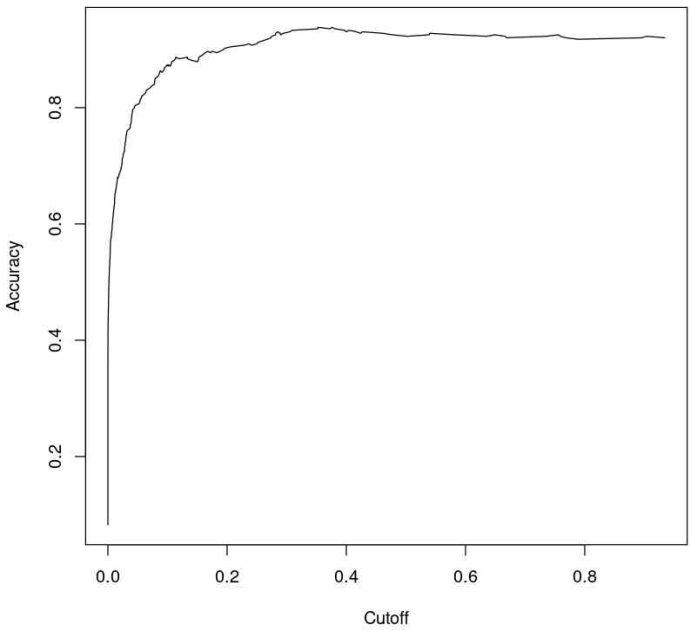
```
          Pos Pred Value : 0.9559
          Neg Pred Value : 0.6400
              Prevalence : 0.9175
          Detection Rate : 0.8943
    Detection Prevalence : 0.9356
       Balanced Accuracy : 0.7374

        'Positive' Class : 0



< RESULT OF ANALYSIS WITHOUT PCA >

[ co_f1 = 0.5 ]                                 |    [ co_ba = 0.17 ]
                                                |
Confusion Matrix and Statistics                 |    Confusion Matrix and Statistics
                                                |
          Reference                             |              Reference
Prediction   0   1                              |    Prediction   0   1
         0 348  22                              |             0 322   6
         1   8  10                              |             1  34  26
                                                |
               Accuracy : 0.9227                |                  Accuracy : 0.8969
                 95% CI : (0.8915, 0.9472)      |                    95% CI : (0.8623, 0.9253)
    No Information Rate : 0.9175                |       No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.40021               |       P-Value [Acc > NIR] : 0.9378
                                                |
                  Kappa : 0.3621                |                     Kappa : 0.5128
                                                |
 Mcnemar's Test P-Value : 0.01762               |     Mcnemar's Test P-Value : 1.963e-05
                                                |
            Sensitivity : 0.9775                |                Sensitivity : 0.9045
            Specificity : 0.3125                |                Specificity : 0.8125
         Pos Pred Value : 0.9405                |             Pos Pred Value : 0.9817
         Neg Pred Value : 0.5556                |             Neg Pred Value : 0.4333
             Prevalence : 0.9175                |                 Prevalence : 0.9175
         Detection Rate : 0.8969                |             Detection Rate : 0.8299
   Detection Prevalence : 0.9536                |       Detection Prevalence : 0.8454
      Balanced Accuracy : 0.6450                |          Balanced Accuracy : 0.8585
                                                |
       'Positive' Class : 0                     |           'Positive' Class : 0
```

```
prediction_obj_logit = prediction(te_pred_logit, target_te)
```

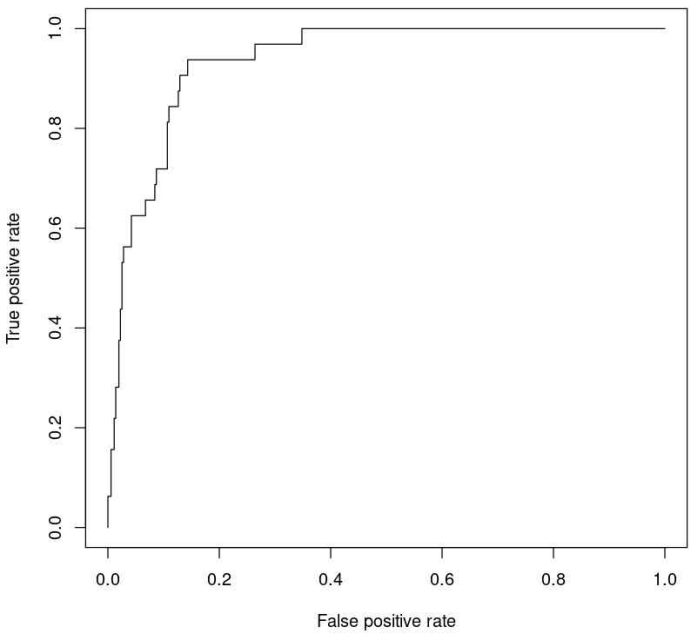
```
performance(prediction_obj_logit, "f")
```

```
A performance instance
  'Cutoff' vs. 'Precision-Recall F measure' (alpha: 'none')
  with 389 data points
```

```
plot(performance(prediction_obj_logit, "acc", "cutoff"))
```




```
plot(performance(prediction_obj_logit, "tpr", "fpr")) # ROC CURVE
# ROC 커브의 곡선 자체는 모형을 변경해야 변화함
# ROC 커브의 포인트는 Cut-off value에 따라 이동함
```

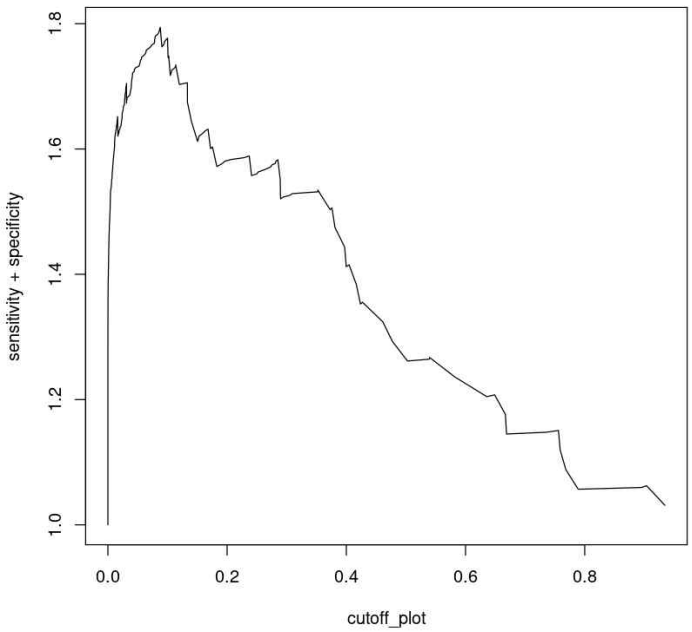



```
str(performance(prediction_obj_logit, "tpr", "fpr"))
# @alpha.values: cut-off values
# @y.values: fpr
# @x.values: tpr
```

```
Formal class 'performance' [package "ROCR"] with 6 slots
  ..@ x.name      : chr "False positive rate"
  ..@ y.name      : chr "True positive rate"
  ..@ alpha.name  : chr "Cutoff"
  ..@ x.values    :List of 1
  .. ..$ : num [1:389] 0 0 0 0.00281 0.00562 ...
  ..@ y.values    :List of 1
  .. ..$ : num [1:389] 0 0.0312 0.0625 0.0625 0.0625 ...
  ..@ alpha.values:List of 1
  .. ..$ : num [1:389] Inf 0.934 0.904 0.896 0.789 ...
```

```
sensitivity = performance(prediction_obj_logit, "tpr")@y.values %>% unlist
specificity = performance(prediction_obj_logit, "tnr")@y.values %>% unlist
```

```
cutoff_plot = performance(prediction_obj_logit, "tpr")@x.values %>% unlist
plot(x=cutoff_plot, y=sensitivity+specificity, type="l") #Balanced Accuracy
```



---

④ Logistic Regression을 제외하고 SVM을 포함하여 3가지 다항 분류 모형을 만들어 Precision과 Sensitivity(TPR)를 제시하시오.

또한 모델향상과정과 최적화 과정을 통해 Confusion Matrix를 도출하시오.

```
fit_svm = train(
    form = paste0("Y", Fault) ~ .,
    data = data_tr_bin,
    trControl = trainControl(method = "none", classProbs = TRUE),
    method = "svmLinear",
    preProcess = c("center","scale")
)

fit_rf = train(
    form = Fault ~ .,
    data = data_tr_bin,
    trControl = trainControl(method="none"),
    method = "rf",
    preProcess = c("center","scale")
)

fit_knn = train(
    form = Fault ~ .,
    data = data_tr_bin,
    trControl = trainControl(method="none"),
    method = "knn",
    preProcess = c("center","scale")
)
```

```
pred_svm = predict(fit_svm, data_te, type="prob")[,2]
pred_rf = predict(fit_rf, data_te, type="prob")[,2]
pred_knn = predict(fit_knn, data_te, type="prob")[,2]
```

```
co_value = co_f1
```

```
pred_svm_bin = as.factor(as.numeric(pred_svm > co_value))
pred_rf_bin = as.factor(as.numeric(pred_rf > co_value))
pred_knn_bin = as.factor(as.numeric(pred_knn > co_value))
```

```
confusionMatrix(pred_svm_bin, target_te)
confusionMatrix(pred_rf_bin, target_te)
confusionMatrix(pred_knn_bin, target_te)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 349  19
         1   7  13

               Accuracy : 0.933
                 95% CI : (0.9034, 0.9558)
    No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.15461

                  Kappa : 0.4661

 Mcnemar's Test P-Value : 0.03098

            Sensitivity : 0.9803
            Specificity : 0.4062
         Pos Pred Value : 0.9484
         Neg Pred Value : 0.6500
             Prevalence : 0.9175
         Detection Rate : 0.8995
   Detection Prevalence : 0.9485
      Balanced Accuracy : 0.6933

       'Positive' Class : 0




Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 352  18
         1   4  14

               Accuracy : 0.9433
                 95% CI : (0.9154, 0.9641)
    No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.034634

                  Kappa : 0.5322

 Mcnemar's Test P-Value : 0.005578

            Sensitivity : 0.9888
            Specificity : 0.4375
         Pos Pred Value : 0.9514
         Neg Pred Value : 0.7778
             Prevalence : 0.9175
         Detection Rate : 0.9072
   Detection Prevalence : 0.9536
      Balanced Accuracy : 0.7131

       'Positive' Class : 0




Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 336  14
         1  20  18

               Accuracy : 0.9124
                 95% CI : (0.8797, 0.9386)
    No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.6849
```

```
                 Kappa : 0.4665

 Mcnemar's Test P-Value : 0.3912

            Sensitivity : 0.9438
            Specificity : 0.5625
         Pos Pred Value : 0.9600
         Neg Pred Value : 0.4737
             Prevalence : 0.9175
         Detection Rate : 0.8660
   Detection Prevalence : 0.9021
      Balanced Accuracy : 0.7532

       'Positive' Class : 0
```

⑤ 상기 ③번과 ④번 4가지 모형 중 1가지를 선택하여 최적의 클러스터링 개수(단일집단~5개)를 제시하시오.

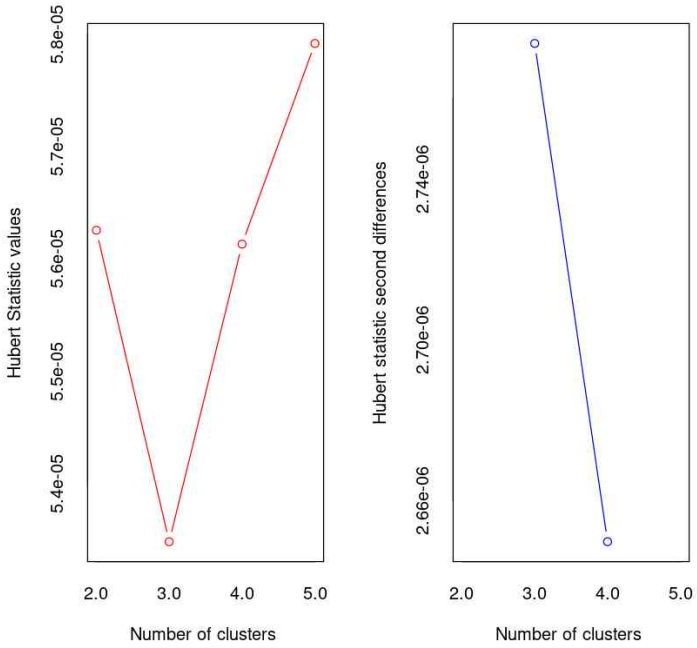
모형성능 향상 과정을 수행하여 Clustering 전후의 F1 Score와 모형 평가 결과를 제시하시오.

ADD CLUSTER INFORMATION

```
data_analog_scale = data %>%
    select(-Fault) %>%
    mutate(SteelType = as.numeric(SteelType)) %>%
    mutate_all(~scale(.))
```

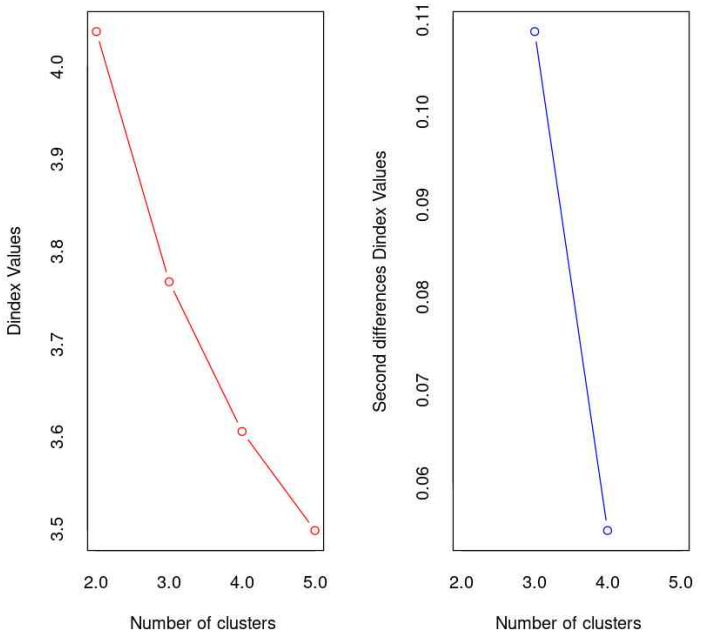
```
nc = NbClust(data_analog_scale, min.nc = 2, max.nc = 5, method = "kmeans")
```

```
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.
```

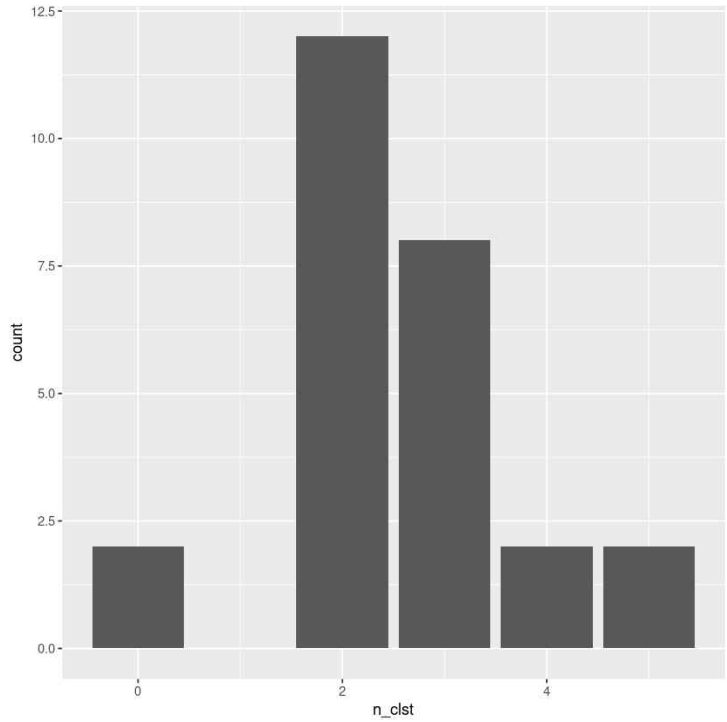



```
*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 12 proposed 2 as the best number of clusters
* 8 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters
* 2 proposed 5 as the best number of clusters
```

```
                  ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2


*******************************************************************
```




```
ggplot(data=data.frame(n_clst=nc$Best.nc[1,])) + geom_bar(stat='count', aes(x=n_clst))
```




```
km_rslt = kmeans(x = data_analog_scale, centers = 2)
```

```
data_clst = data.frame(data, Cluster=km_rslt$cluster) %>%
    mutate(Fault = as.factor(ifelse(Fault == 1, 1, 0)),
           Cluster = as.factor(Cluster))
```

```
data_clst_tr = data_clst[idx_tr,]
data_clst_vd = data_clst[idx_vd,]
data_clst_te = data_clst[idx_te,]
```

(SVM) FIT

```
fit_svm_clst = train(
    form = paste0("Y", Fault) ~ .,
    data = data_clst_tr,
    trControl = trainControl(method = "none", classProbs = TRUE),
    method = "svmLinear",
    preProcess = c("center","scale")
)
```

(SVM) SEARCH CUT-OFF VALUE

```
pred_svm_clst = predict(fit_svm_clst, data_clst_vd, type="prob")[,2]

co_range = seq(0, 1, 0.01)
pred_matx_svm_clst = as.matrix(pred_svm_clst) %*% rep(1, length(co_range))
co_matx_svm_clst = rep(1, NROW(pred_matx_svm_clst)) %*% t(co_range)

pred_class_by_cutoff_svm_clst = as.data.frame(pred_matx_svm_clst > co_matx_svm_clst) %>%
    mutate_all(~as.factor(as.numeric(.)))
colnames(pred_class_by_cutoff_svm_clst) = co_range
```

```
cnfm_svm_clst = list()
crit_svm_clst = list()
for (i in seq_along(pred_class_by_cutoff_svm_clst)) {
    cm_svm_clst = confusionMatrix(pred_class_by_cutoff_svm_clst[,i], data_clst_vd$Fault)
    cnfm_svm_clst[[i]] = cm_svm_clst
    crit_svm_clst[[i]] = round(cm_svm_clst$byClass, 4)
}
result_svm_clst = as_tibble(cbind(co_range, (t(data.frame(crit_svm_clst)))))
```

```
result_svm_clst %>% mutate(SS = Sensitivity + Specificity - 1) %>% filter(SS == max(SS))
result_svm_clst %>% filter(F1 == max(F1, na.rm=TRUE))
```

A tibble: 1 × 13

co_range	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy	SS
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.14	0.8783	0.8085	0.9812	0.3689	0.9812	0.8783	0.9269	0.9191	0.8072	0.8227	0.8434	0.6868

A tibble: 1 × 12

co_range	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.43	0.985	0.2979	0.941	0.6364	0.941	0.985	0.9625	0.9191	0.9053	0.9621	0.6414

(SVM) EVALUATE WITH TEST DATASET

```
pred_svm_clst = predict(fit_svm_clst, data_clst_te, type="prob")[,2]
```

```
co_value = 0.46
pred_svm_clst_bin = as.factor(as.numeric(pred_svm_clst > co_value))
```

```
confusionMatrix(pred_svm_clst_bin, target_te)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 350  25
         1   6   7
```

```
               Accuracy : 0.9201
                 95% CI : (0.8885, 0.9451)
    No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.473440

                  Kappa : 0.2766

 Mcnemar's Test P-Value : 0.001225

            Sensitivity : 0.9831
            Specificity : 0.2188
         Pos Pred Value : 0.9333
         Neg Pred Value : 0.5385
             Prevalence : 0.9175
         Detection Rate : 0.9021
   Detection Prevalence : 0.9665
      Balanced Accuracy : 0.6009

       'Positive' Class : 0
```

```
    (BEFORE ADJUSTING CUT-OFF VALUE)

    Confusion Matrix and Statistics

              Reference
    Prediction   0   1
             0 322   9
             1  34  23

                   Accuracy : 0.8892
                     95% CI : (0.8536, 0.9186)
        No Information Rate : 0.9175
        P-Value [Acc > NIR] : 0.9796069

                      Kappa : 0.4598

     Mcnemar's Test P-Value : 0.0002522

                Sensitivity : 0.9045
                Specificity : 0.7188
             Pos Pred Value : 0.9728
             Neg Pred Value : 0.4035
                 Prevalence : 0.9175
             Detection Rate : 0.8299
       Detection Prevalence : 0.8531
          Balanced Accuracy : 0.8116

           'Positive' Class : 0
```

```
    (AFTER ADJUSTING CUT-OFF VALUE)

    Confusion Matrix and Statistics

              Reference
  Prediction   0   1
             0 257   2
             1  99  30

                   Accuracy : 0.7397
                     95% CI : (0.693, 0.7827)
        No Information Rate : 0.9175
        P-Value [Acc > NIR] : 1

                      Kappa : 0.2771

   Mcnemar's Test P-Value : <2e-16

                Sensitivity : 0.7219
                Specificity : 0.9375
             Pos Pred Value : 0.9923
             Neg Pred Value : 0.2326
                 Prevalence : 0.9175
             Detection Rate : 0.6624
       Detection Prevalence : 0.6675
          Balanced Accuracy : 0.8297

           'Positive' Class : 0
```

(LOGISTIC) FIT

```
fit_logit_clst = train(
    form = Fault ~ .,
    data = data_clst_tr,
    trControl = trainControl(method="none"),
    method = "glmStepAIC",
    family = "binomial"
)
```

(LOGISTIC) SEARCH CUT-OFF VALUE

```
pred_logit_clst = predict(fit_logit_clst, data_clst_vd, type="prob")[,2]

co_range = seq(0, 1, 0.01)
pred_matx_logit_clst = as.matrix(pred_logit_clst) %*% rep(1, length(co_range))
co_matx_logit_clst = rep(1, NROW(pred_matx_logit_clst)) %*% t(co_range)

pred_class_by_cutoff_logit_clst = as.data.frame(pred_matx_logit_clst > co_matx_logit_clst) %>%
    mutate_all(~as.factor(as.numeric(.)))
colnames(pred_class_by_cutoff_logit_clst) = co_range
```

```
cnfm_logit_clst = list()
crit_logit_clst = list()

for (i in seq_along(pred_class_by_cutoff_logit_clst)) {
    cm_logit_clst = confusionMatrix(pred_class_by_cutoff_logit_clst[,i], data_clst_vd$Fault)
    cnfm_logit_clst[[i]] = cm_logit_clst
    crit_logit_clst[[i]] = round(cm_logit_clst$byClass, 4)
}
result_logit_clst = as_tibble(cbind(co_range, (t(data.frame(crit_logit_clst)))))
```

```
Warning message in confusionMatrix.default(pred_class_by_cutoff_logit_clst[, i], :
“Levels are not in the same order for reference and data. Refactoring data to match.”
Warning message in confusionMatrix.default(pred_class_by_cutoff_logit_clst[, i], :
“Levels are not in the same order for reference and data. Refactoring data to match.”
Warning message in confusionMatrix.default(pred_class_by_cutoff_logit_clst[, i], :
“Levels are not in the same order for reference and data. Refactoring data to match.”
Warning message in confusionMatrix.default(pred_class_by_cutoff_logit_clst[, i], :
“Levels are not in the same order for reference and data. Refactoring data to match.”
Warning message in confusionMatrix.default(pred_class_by_cutoff_logit_clst[, i], :
“Levels are not in the same order for reference and data. Refactoring data to match.”
Warning message in confusionMatrix.default(pred_class_by_cutoff_logit_clst[, i], :
“Levels are not in the same order for reference and data. Refactoring data to match.”
Warning message in confusionMatrix.default(pred_class_by_cutoff_logit_clst[, i], :
“Levels are not in the same order for reference and data. Refactoring data to match.”
```

```
result_logit_clst %>% mutate(SS = Sensitivity + Specificity - 1) %>% filter(SS == max(SS))
result_logit_clst %>% filter(F1 == max(F1, na.rm=TRUE))
```

A tibble: 1 × 13

co_range	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy	SS
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.18	0.9195	0.8085	0.982	0.4691	0.982	0.9195	0.9497	0.9191	0.8451	0.8606	0.864	0.728

A tibble: 2 × 12

co_range	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Precision	Recall	F1	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.48	0.9775	0.4255	0.9508	0.625	0.9508	0.9775	0.964	0.9191	0.8985	0.9449	0.7015
0.49	0.9775	0.4255	0.9508	0.625	0.9508	0.9775	0.964	0.9191	0.8985	0.9449	0.7015

(LOGISTIC) EVALUATE WITH TEST DATASET

```
pred_logit_clst = predict(fit_logit_clst, data_clst_te, type="prob")[,2]
```

```
co_value = co_f1
pred_logit_clst_bin = as.factor(as.numeric(pred_logit_clst > co_value))

confusionMatrix(pred_logit_clst_bin, target_te)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 343  19
         1  13  13

               Accuracy : 0.9175
                 95% CI : (0.8856, 0.9429)
    No Information Rate : 0.9175
    P-Value [Acc > NIR] : 0.5469

                  Kappa : 0.4042

 Mcnemar's Test P-Value : 0.3768
```

```
            Sensitivity : 0.9635
            Specificity : 0.4062
         Pos Pred Value : 0.9475
         Neg Pred Value : 0.5000
             Prevalence : 0.9175
         Detection Rate : 0.8840
   Detection Prevalence : 0.9330
      Balanced Accuracy : 0.6849

       'Positive' Class : 0
```

Question No. 2

주어진 3개의 파일들은 한 공장의 전력 사용량에 대한 데이터로써, 각각 날씨와 온도, 용도별 전력량계, 전력 총 사용량을 담고있다.
해당 데이터를 종합적으로 이용하여 다음 문제를 풀이하시오.

- 각 데이터 파일에 대한 설명:
  1. E15Q21_usage.csv
     - 900초마다 기록된 900초 단위 전력 총 사용량
     - 1번 컬럼: Datetime (UnixTimestamp)
     - 2번 컬럼: Usage
  2. E15Q22_weather.csv
     - 일자별 평균 기온
     - 1번 컬럼: Date (YYYY-MM-DD)
     - 2번 컬럼: Daily Average Temperature
  3. E15Q23_usage_history.tsv
     - 1분에 2번씩 기록된 각 용도별 전력 누적사용량
     - 1번 컬럼: Time (HH:MM)
     - 2번 컬럼: Weather Class (A/B/C/D)
     - 3-7번 컬럼: 각 용도(A/B/C/D/E)별 전력 누적 사용량

LOAD DATA

```
usage = read_delim(file = "./data/E15Q21_usage.csv", delim = ",")
weather = read_delim(file = "./data/E15Q22_weather.csv", delim = ",")
usage_history = read_delim(file = "./data/E15Q23_usage_history.tsv", delim = "\t")
```

```
Parsed with column specification:
cols(
  timestamp =col_double(),
  amount =col_double()
)

Parsed with column specification:
cols(
  dt = col_date(format = ""),
  avg_temp =col_double()
)

Parsed with column specification:
cols(
  time = col_time(format = ""),
  wclass = col_character(),
  A =col_double(),
  B =col_double(),
  C =col_double(),
  D =col_double(),
  E =col_double()
)
```

① 첫번째 제공 파일의 총사용량 컬럼을 용도별로 분류하고, 연월과 사용 목적별로 전력의 하루 평균 사용량을 구하여 도표를 도출하시오.

YYYYMM	A	B	C	D	E
202001	__	__	__	__	__
202002	__	__	__	__	__
202003	__	__	__	__	__

CREATE DIFF FROM CUMSUM

```
usage_history_lag = usage_history %>%
    transmute_at(vars(A, B, C, D, E), ~lag(., 1, 0))
```

```
usage_history_diff = usage_history %>%
    select(time) %>%
    cbind(usage_history[,LETTERS[1:5]] - usage_history_lag) %>%
    mutate(timestamp_idx=rep(1:(NROW(.)/30), each=30))
```

CREATE JOIN KEY

```
timestamp_usage_history = usage_history_diff %>%
    group_by(timestamp_idx) %>%
    summarise(ts_amount=sum(A, B, C, D, E))
```

```
# PK TEST FOR JOINING
prcs = 10
PK_TEST = inner_join(timestamp_usage_history %>%
    mutate(jkey=signif(ts_amount, prcs)), usage %>%
    mutate(jkey=signif(amount, prcs)), by="jkey")
NROW(PK_TEST) != NROW(timestamp_usage_history) # DUPLICATE OR NON-MAPPING? (FALSE IS OKAY)
```

FALSE

```
timestamp_key = inner_join(timestamp_usage_history %>% mutate(jkey=signif(ts_amount, prcs)),
                           usage %>% mutate(jkey=signif(amount, prcs)),
                           by="jkey") %>%
    select(timestamp_idx, timestamp, jkey)
```

JOIN `usage_history_diff` AND `timestamp_key` TO MAP DATE

```
usage_history_diff %>%
    left_join(timestamp_key, by="timestamp_idx") %>%
    mutate(dt = as.Date(as.POSIXct(timestamp, origin="1970-01-01"))) %>%
    select(dt, LETTERS[1:5]) %>%
    pivot_longer(cols=LETTERS[1:5], names_to="obj",
                                    values_to="min_obj_usage") %>% # 분단위 목적별 Usage (분당 2레코드)
    group_by(dt, obj) %>%
    summarise(daily_obj_usage = sum(min_obj_usage)) %>%            # 일단위 목적별 Usage
    group_by(ym=str_sub(dt, 1, 7), obj) %>%
    summarise(monthly_obj_mean_usage = mean(daily_obj_usage)) %>%  # 월별 목적별 하루 평균 Usage
    pivot_wider(id_cols=ym, names_from=obj, values_from=monthly_obj_mean_usage)
```

A grouped_df: 3 × 6

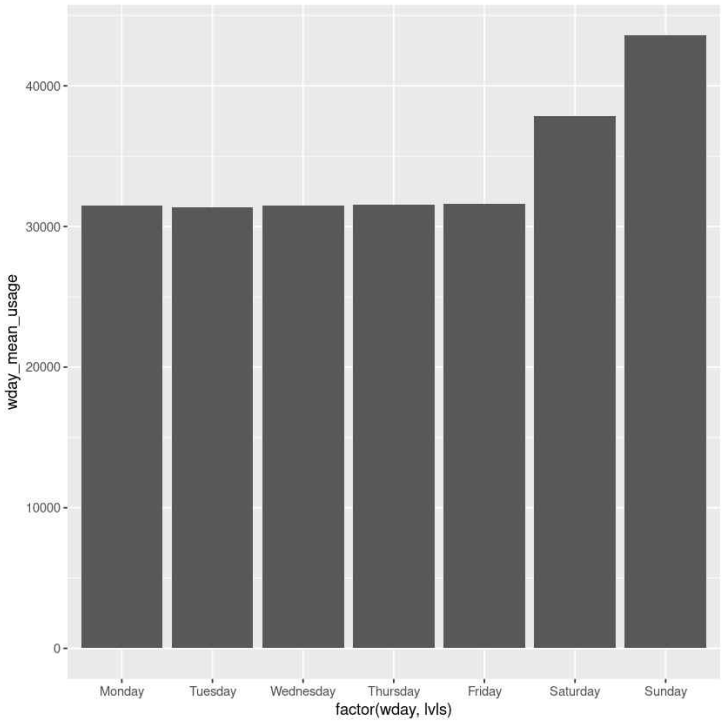
ym	A	B	C	D	E
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
2017-09	4419.741	7586.948	7291.643	6788.298	8155.831
2017-10	4338.153	7729.573	7198.394	6771.779	8177.260
2017-11	4248.502	7488.022	7108.825	6783.897	7836.024

---

② 요일별 평균 전력사용량을 도출하시오. 또한 가로축을 요일, 세로축을 평균사용량으로 하여 요일별 평균 사용량을 시각화하여 제출하시오.

```
wday_mean_usage = usage %>%
    transmute(dt = as.Date(as.POSIXct(timestamp, origin="1970-01-01")),
              wday = weekdays(dt),
              amount=amount) %>%
    group_by(dt, wday) %>%
    summarise(daily_usage = sum(amount)) %>%
    group_by(wday) %>%
    summarise(wday_mean_usage = mean(daily_usage)) # 요일별 평균 Daily Usage
```

```
lvls = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
ggplot(data = wday_mean_usage) + geom_bar(stat='identity', aes(factor(wday, lvls), wday_mean_usage))
```



---

③ 요일별 총 전력 사용량의 평균값의 차이를 분석하여, 가장 큰 차이를 보이는 요일은 어떤 요일인지 제시하시오.

```
data_aov = usage %>%
    transmute(dt = as.Date(as.POSIXct(timestamp, origin="1970-01-01")),
              amount=amount) %>%
    group_by(dt) %>%
    summarise(daily_usage = sum(amount)) %>%
    mutate(wday = weekdays(dt))
```

```
data_aov %>% group_by(wday) %>% summarise(d=mean(daily_usage))
```

A tibble: 7 × 2

wday	d
<chr>	<dbl>
Friday	31627.42
Monday	31467.46
Saturday	37872.24
Sunday	43575.78
Thursday	31522.36
Tuesday	31361.24
Wednesday	31486.61

```
fit_aov = aov(daily_usage ~ wday, data=data_aov)
```

```
summary(fit_aov)
```

```
             Df    Sum Sq   Mean Sq F value Pr(>F)
wday          6 1.380e+09 230054259    2779 <2e-16 ***
```

```
Residuals    63 5.215e+06     82776
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
bartlett.test(daily_usage ~ wday, data=data_aov) # Homogeneity of Variance Test
```

```
        Bartlett test of homogeneity of variances

data:  daily_usage by wday
Bartlett's K-squared = 3.6197, df = 6, p-value = 0.728
```

---

④ 각 날짜별 평균 기온과 용도별 전력사용량의 관계를 분석하여, 기온과 가장 밀접한 관계를 지닌 사용 용도의 종류를 제시하시오.

```
obj_corr = usage_history_diff %>%
    left_join(timestamp_key, by="timestamp_idx") %>%
    mutate(dt = as.Date(as.POSIXct(timestamp, origin="1970-01-01"))) %>%
    group_by(dt) %>%
    summarise_at(vars(LETTERS[1:5]), ~sum(.)) %>%
    left_join(weather, by="dt") %>%
    pivot_longer(cols=LETTERS[1:5], names_to='obj', values_to="daily_obj_usage") %>%
    group_by(obj) %>%
    summarise(corr = cor(avg_temp, daily_obj_usage))
```

```
obj_corr
```

A tibble: 5 × 2

obj	corr
<chr>	<dbl>
A	0.730820745
B	0.080761375
C	0.178267511
D	-0.009414051
E	0.170544094

```
ggplot(data=obj_corr) + geom_bar(stat='identity', aes(obj, corr))
```