데이터 전처리

```
import pandas as pd
import numpy as np
import warnings
```

전체 데이터셋 : train

```
train = pd.read_excel('C:/Users/user/Desktop/Statistical_Data_Idea_Contest/DATA/
데이터/기술통계19.xlsx',index_col='global_id')
```

```
print('기술통계자료의 데이터 개수')
print('train: {}'.format(len(train)))
```

```
기술통계자료의 데이터 개수
train: 3994
```

서비스업: service, 일반업: normal, 제조업: product

```
service =
pd.read_excel('C:/Users/user/Desktop/Statistical_Data_Idea_Contest/DATA/데이터/실
태조사_서비스업19.xlsx',index_col='global_id')
normal =
pd.read_excel('C:/Users/user/Desktop/Statistical_Data_Idea_Contest/DATA/데이터/실
태조사_일반항목19.xlsx',index_col='global_id')
product =
pd.read_excel('C:/Users/user/Desktop/Statistical_Data_Idea_Contest/DATA/데이터/실
태조사_제조업19.xlsx',index_col='global_id')
```

```
print('실태조사 자료 데이터의 개수')
print('sevice:{}, normal:{}, product:{}'.format(len(service), len(normal),
len(product)))
```

```
실태조사 자료 데이터의 개수
sevice:8500, normal:4000, product:7500
```

기술통계조사와 실태조사 자료의 중복자료 추출

```
warnings.filterwarnings(action='ignore')
print('기술통계조사와 실태조사 자료의 중복항목 수는 다음과 같다.')
# 서비스업
line=[]
for i in train.index:
    if i in service.index:
        line.append(i)
    else:
        line.append(0)

service_train =train[train.index == line]
service_train['bt'] = 1
print('서비스업종 데이터: {}'.format(len(service_train)))

# 일반업
line=[]
for i in train.index:
    if i in normal.index:
        line.append(i)
    else:
        line.append(0)

normal_train =train[train.index == line]
normal_train['bt'] = 2
print('일반업종 데이터: {}'.format(len(normal_train)))

# 제조업
line = []
for i in train.index:
    if i in product.index:
        line.append(i)
    else:
        line.append(0)

product_train =train[train.index == line]
product_train['bt'] = 2
print('제조업종 데이터: {}'.format(len(product_train)))
```

```
기술통계조사와 실태조사 자료의 중복항목 수는 다음과 같다.
서비스업종 데이터: 195
일반업종 데이터: 183
제조업종 데이터: 834
```

데이터 병합

```
train = pd.concat([service_train, normal_train, product_train])
index_list = ['A1S1', 'A1N1', 'C2S2', 'H1_1', 'H4_1', 'I4Q1', 'I4Q2',  'I4Q3',
'I4Q4', 'I4Q5', 'I4Q6', 'I4Q7']
my_train = train.loc[:, index_list]
```

```
print(my_train.head(5))
```

```
           A1S1  A1N1  C2S2  H1_1  H4_1  I4Q1  I4Q2  I4Q3  I4Q4  I4Q5  I4Q6  \
global_id
111016.0    1.0   1.0   2.0   3.0   1.0   3.0   3.0   1.0   1.0   1.0   3.0
111037.0    1.0   1.0   4.0   NaN   1.0   3.0   3.0   1.0   1.0   3.0   3.0
111039.0    4.0   1.0   3.0   3.0   7.0   3.0   3.0   1.0   1.0   3.0   3.0
111041.0    4.0   5.0   2.0   5.0   NaN   3.0   3.0   3.0   3.0   2.0   2.0
111055.0    5.0   5.0   2.0   NaN   NaN   4.0   4.0   4.0   2.0   2.0   2.0

           I4Q7
global_id
```

```
111016.0    3.0
111037.0    3.0
111039.0    3.0
111041.0    2.0
111055.0    2.0
```

결측치 최빈값 처리

```
for index in index_list:
    freq = my_train[index].value_counts(dropna=True).idxmax()
    my_train[index].fillna(freq, inplace = True)
print(my_train.head(5))
```

```
           A1S1  A1N1  C2S2  H1_1  H4_1  I4Q1  I4Q2  I4Q3  I4Q4  I4Q5  I4Q6  \
global_id                                                                     
111016.0    1.0   1.0   2.0   3.0   1.0   3.0   3.0   1.0   1.0   1.0   3.0   
111037.0    1.0   1.0   4.0   3.0   1.0   3.0   3.0   1.0   1.0   3.0   3.0   
111039.0    4.0   1.0   3.0   3.0   7.0   3.0   3.0   1.0   1.0   3.0   3.0   
111041.0    4.0   5.0   2.0   5.0   1.0   3.0   3.0   3.0   3.0   2.0   2.0   
111055.0    5.0   5.0   2.0   3.0   1.0   4.0   4.0   4.0   2.0   2.0   2.0   

           I4Q7  
global_id        
111016.0    3.0  
111037.0    3.0  
111039.0    3.0  
111041.0    2.0  
111055.0    2.0  
```

응답범주 축소화

```
warnings.filterwarnings(action='ignore')

my_train['A1S1'][(my_train['A1S1']==1) | (my_train['A1S1']==2)] = 1
my_train['A1S1'][my_train['A1S1']==3] = 2
my_train['A1S1'][(my_train['A1S1']==4) | (my_train['A1S1']==5)|
(my_train['A1S1']==6)] = 3
my_train['A1S1'][my_train['A1S1']==7] = 4

my_train['A1N1'] [(my_train['A1N1']==1) | (my_train['A1N1']==2)] = 1
my_train['A1N1'] [(my_train['A1N1']==10) | (my_train['A1N1']==11)] = 2
my_train['A1N1'] [(my_train['A1N1']==5) | (my_train['A1N1']==6)|
(my_train['A1N1']==12| (my_train['A1N1']==8| (my_train['A1N1']==9)))] = 3
my_train['A1N1'] [(my_train['A1N1']==3) | (my_train['A1N1']==7)] = 4
my_train['A1N1'] [(my_train['A1N1']==13) | (my_train['A1N1']==14)|
(my_train['A1N1']==15)] = 5

my_train['C2S2'][(my_train['C2S2']==2) | (my_train['C2S2']==4)] = 2

my_train['H1_1'][(my_train['H1_1']==2) | (my_train['H1_1']==3)] = 1
my_train['H1_1'][(my_train['H1_1']==4) | (my_train['H1_1']==5)|
(my_train['H1_1']==6)| (my_train['H1_1']==7)] = 2
my_train['H1_1'][(my_train['H1_1']==1)] = 3
my_train['H1_1'][(my_train['H1_1']==9) | (my_train['H1_1']==8)]= 4
```

```
my_train['H4_1'][(my_train['H4_1']==1) | (my_train['H4_1']==3)] = 1
my_train['H4_1'][(my_train['H4_1']==4)] = 2
my_train['H4_1'][(my_train['H4_1']==5) | (my_train['H4_1']==6)] = 3
my_train['H4_1'][(my_train['H4_1']==2)] = 4
my_train['H4_1'][(my_train['H4_1']==7) | (my_train['H4_1']==9)] = 5
my_train['H4_1'][(my_train['H4_1']==8) | (my_train['H4_1']==10)] = 6

my_train['I4Q1'][(my_train['I4Q1']==3) | (my_train['I4Q1']==4)] = 1
my_train['I4Q1'][(my_train['I4Q1']==2)] = 2
my_train['I4Q1'][(my_train['I4Q1']==1)] = 3

my_train['I4Q2'][(my_train['I4Q2']==3) | (my_train['I4Q2']==4)] = 1
my_train['I4Q2'][(my_train['I4Q2']==2)] = 2
my_train['I4Q2'][(my_train['I4Q2']==1)] = 3

my_train['I4Q3'][(my_train['I4Q3']==3) | (my_train['I4Q3']==4)] = 1
my_train['I4Q3'][(my_train['I4Q3']==2)] = 2
my_train['I4Q3'][(my_train['I4Q3']==1)] = 3

my_train['I4Q4'][(my_train['I4Q4']==3) | (my_train['I4Q4']==4)] = 1
my_train['I4Q4'][(my_train['I4Q4']==2)] = 2
my_train['I4Q4'][(my_train['I4Q4']==1)] = 3

my_train['I4Q5'][(my_train['I4Q5']==3) | (my_train['I4Q5']==4)] = 1
my_train['I4Q5'][(my_train['I4Q5']==2)] = 2
my_train['I4Q5'][(my_train['I4Q5']==1)] = 3

my_train['I4Q6'][(my_train['I4Q6']==3) | (my_train['I4Q6']==4)] = 1
my_train['I4Q6'][(my_train['I4Q6']==2)] = 2
my_train['I4Q6'][(my_train['I4Q6']==1)] = 3

my_train['I4Q7'][(my_train['I4Q7']==3) | (my_train['I4Q7']==4)] = 1
my_train['I4Q7'][(my_train['I4Q7']==2)] = 2
my_train['I4Q7'][(my_train['I4Q7']==1)] = 3

# #문자열처리 및 더미변수화 하려면 아래코드돌리기
# for index in index_list:
#     my_train[index] = my_train[index].astype(str)
#     my_train = pd.get_dummies(my_train, columns=[index])

print(my_train.head(5))
print(my_train.dtypes)
```

```
           A1S1  A1N1  C2S2  H1_1  H4_1  I4Q1  I4Q2  I4Q3  I4Q4  I4Q5  I4Q6  \
global_id
111016.0    1.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   3.0   3.0   3.0
111037.0    1.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   3.0   3.0   3.0
111039.0    3.0   4.0   3.0   3.0   5.0   3.0   3.0   3.0   3.0   3.0   3.0
111041.0    3.0   4.0   2.0   2.0   1.0   3.0   3.0   3.0   3.0   2.0   2.0
111055.0    3.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   2.0   2.0   2.0

           I4Q7
global_id
111016.0    3.0
111037.0    3.0
111039.0    3.0
```

```
111041.0    2.0
111055.0    2.0
A1S1     float64
A1N1     float64
C2S2     float64
H1_1     float64
H4_1     float64
I4Q1     float64
I4Q2     float64
I4Q3     float64
I4Q4     float64
I4Q5     float64
I4Q6     float64
I4Q7     float64
dtype: object
```

K-MEANS 클러스터

```
from sklearn.cluster import KMeans
```

```
model = KMeans(n_clusters=3)
model.fit(my_train)
y_predict = model.fit_predict(my_train)
my_train['k_means'] = y_predict
print(my_train.head(5))
print(my_train.groupby('k_means').size())
```

```
           A1S1  A1N1  C2S2  H1_1  H4_1  I4Q1  I4Q2  I4Q3  I4Q4  I4Q5  I4Q6  \
global_id
111016.0    1.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   3.0   3.0   3.0
111037.0    1.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   3.0   3.0   3.0
111039.0    3.0   4.0   3.0   3.0   5.0   3.0   3.0   3.0   3.0   3.0   3.0
111041.0    3.0   4.0   2.0   2.0   1.0   3.0   3.0   3.0   3.0   2.0   2.0
111055.0    3.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   2.0   2.0   2.0

           I4Q7  k_means
global_id
111016.0    3.0        1
111037.0    3.0        1
111039.0    3.0        0
111041.0    2.0        1
111055.0    2.0        1
k_means
0    343
1    849
2     20
dtype: int64
```

스펙트럴 군집

```
from sklearn.cluster import SpectralClustering
```

```
spectral = SpectralClustering(n_clusters=3, n_init=10)
y_predict = spectral.fit_predict(my_train)
my_train['Spectral'] = y_predict
print(my_train.head(5))

my_train.to_csv('C:/Users/user/Desktop/Statistical_Data_Idea_Contest/0423/kyumin
/cluster_result.csv', index=True)
```

```
           A1S1  A1N1  C2S2  H1_1  H4_1  I4Q1  I4Q2  I4Q3  I4Q4  I4Q5  I4Q6  \
global_id
111016.0    1.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   3.0   3.0   3.0
111037.0    1.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   3.0   3.0   3.0
111039.0    3.0   4.0   3.0   3.0   5.0   3.0   3.0   3.0   3.0   3.0   3.0
111041.0    3.0   4.0   2.0   2.0   1.0   3.0   3.0   3.0   3.0   2.0   2.0
111055.0    3.0   4.0   2.0   3.0   1.0   3.0   3.0   3.0   2.0   2.0   2.0

           I4Q7  k_means  Spectral
global_id
111016.0    3.0        1         0
111037.0    3.0        1         0
111039.0    3.0        0         0
111041.0    2.0        1         0
111055.0    2.0        1         0
```

Clustrering 결과분석

```
print(my_train.groupby('k_means').size())
print(my_train.groupby('Spectral').size())
```

```
k_means
0    343
1    849
2     20
dtype: int64
Spectral
0    1192
1       4
2      16
dtype: int64
```

혹시몰라서 int로도 string으로도 해봤으나 결과 값은 똑같다!

더미 변수처리는 했는데 Sptectral에서 너무 오래걸린다(안돌아감)

https://m.blog.naver.com/PostView.nhn?blogId=ssdyka&logNo=221284738829&proxyReferer=https:%2F%2Fwww.google.com%2F

```

```