2020 데이터사이언스 프로젝트 최종 발표

국내 병원의 개/폐업 예측

팀원 : 김영민, 강민정

본 프로젝트는 병원에 대한 회계 데이터를 통해 국내 병원 개/폐업 예측한다.

(1) KDD 분석 절차에 따라 진행되며 EDA의 시각화를 통해 데이터의 특징 파악

(2) 머신러닝 모델과 인공 신경망 모델의 성능을 비교하고 이를 통해 보다 높은 정확도를 얻어냄

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
plt.style.use('ggplot')
```

```
# 구글 드라이브 마운트
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=94731898980

Enter your authorization code:
··········
Mounted at /content/gdrive
```

데이터 설명

train.csv - 의료기관이 폐업했는지 여부를 포함하여 최근 2개년의 재무정보와 병원 기본정보

test.csv - 폐업 여부를 제외하고 train.csv와 동일

sample_submission.csv - inst_id와 open과 close를 예측하는 OC 두개의 열로 구성. OC의 값은 open 예측일 경우 1, close 예측일 경우 0.

inst_id - 각 파일에서의 병원 고유 번호

OC – 영업/폐업 분류, 2018년 폐업은 2017년 폐업으로 간주함

sido – 병원의 광역 지역 정보

sgg – 병원의 시군구 자료

openDate – 병원 설립일

bedCount - 병원이 갖추고 있는 병상의 수

instkind – 병원, 의원, 요양병원, 한의원, 종합병원 등 병원의 종류

· 종합병원 : 입원환자 100명 이상 수용 가능

· 병원 : 입원 환자 30명 이상 100명 미만 수용 가능

· 의원 : 입원 환자 30명 이하 수용 가능

· 한방 병원(한의원) : 침술과 한약으로 치료하는 의료 기관.

revenue1 – 매출액, 2017(회계년도)년 데이터를 의미함

salescost1 – 매출원가, 2017(회계년도)년 데이터를 의미함

sga1 - 판매비와 관리비, 2017(회계년도)년 데이터를 의미함

salary1 – 급여, 2017(회계년도)년 데이터를 의미함

noi1 – 영업외수익, 2017(회계년도)년 데이터를 의미함

noe1 – 영업외비용, 2017(회계년도)년 데이터를 의미함

Interest1 – 이자비용, 2017(회계년도)년 데이터를 의미함

ctax1 – 법인세비용, 2017(회계년도)년 데이터를 의미함

Profit1 – 당기순이익, 2017(회계년도)년 데이터를 의미함

liquidAsset1 – 유동자산, 2017(회계년도)년 데이터를 의미함

quickAsset1 – 당좌자산, 2017(회계년도)년 데이터를 의미함

receivableS1 - 미수금(단기), 2017(회계년도)년 데이터를 의미함

inventoryAsset1 – 재고자산, 2017(회계년도)년 데이터를 의미함

nonCAsset1 – 비유동자산, 2017(회계년도)년 데이터를 의미함

tanAsset1 – 유형자산, 2017(회계년도)년 데이터를 의미함

OnonCAsset1 - 기타 비유동자산, 2017(회계년도)년 데이터를 의미함

receivableL1 – 장기미수금, 2017(회계년도)년 데이터를 의미함

debt1 – 부채총계, 2017(회계년도)년 데이터를 의미함

liquidLiabilities1 – 유동부채, 2017(회계년도)년 데이터를 의미함

shortLoan1 – 단기차입금, 2017(회계년도)년 데이터를 의미함

NCLiabilities1 – 비유동부채, 2017(회계년도)년 데이터를 의미함

longLoan1 – 장기차입금, 2017(회계년도)년 데이터를 의미함

netAsset1 – 순자산총계, 2017(회계년도)년 데이터를 의미함

surplus1 – 이익잉여금, 2017(회계년도)년 데이터를 의미함

revenue2 – 매출액, 2016(회계년도)년 데이터를 의미함

salescost2 – 매출원가, 2016(회계년도)년 데이터를 의미함

sga2 - 판매비와 관리비, 2016(회계년도)년 데이터를 의미함

salary2 – 급여, 2016(회계년도)년 데이터를 의미함

noi2 – 영업외수익, 2016(회계년도)년 데이터를 의미함

noe2 – 영업외비용, 2016(회계년도)년 데이터를 의미함

interest2 – 이자비용, 2016(회계년도)년 데이터를 의미함

ctax2 – 법인세비용, 2016(회계년도)년 데이터를 의미함

profit2 – 당기순이익, 2016(회계년도)년 데이터를 의미함

liquidAsset2 – 유동자산, 2016(회계년도)년 데이터를 의미함

quickAsset2 – 당좌자산, 2016(회계년도)년 데이터를 의미함

receivableS2 - 미수금(단기), 2016(회계년도)년 데이터를 의미함

inventoryAsset2 – 재고자산, 2016(회계년도)년 데이터를 의미함

nonCAsset2 – 비유동자산, 2016(회계년도)년 데이터를 의미함

tanAsset2 – 유형자산, 2016(회계년도)년 데이터를 의미함

OnonCAsset2 - 기타 비유동자산, 2016(회계년도)년 데이터를 의미함

receivableL2 – 장기미수금, 2016(회계년도)년 데이터를 의미함

Debt2 – 부채총계, 2016(회계년도)년 데이터를 의미함

liquidLiabilities2 – 유동부채, 2016(회계년도)년 데이터를 의미함

shortLoan2 – 단기차입금, 2016(회계년도)년 데이터를 의미함

NCLiabilities2 – 비유동부채, 2016(회계년도)년 데이터를 의미함

longLoan2 – 장기차입금, 2016(회계년도)년 데이터를 의미함

netAsset2 – 순자산총계, 2016(회계년도)년 데이터를 의미함

surplus2 – 이익잉여금, 2016(회계년도)년 데이터를 의미함

employee1 – 고용한 총 직원의 수, 2017(회계년도)년 데이터를 의미함

employee2 – 고용한 총 직원의 수, 2016(회계년도)년 데이터를 의미함

ownerChange – 대표자의 변동

```
train = pd.read_csv('/content/gdrive/My Drive/hospital close or open/train.csv',parse_dates=['op
test=pd.read_csv('/content/gdrive/My Drive/hospital close or open/test.csv',parse_dates=['openDa
answer = pd.read_csv('/content/gdrive/My Drive/hospital close or open/submission_sample.csv')
```

```
train_len = train.shape[0]
train_id = train.inst_id
test_len = test.shape[0]
test_id = test.inst_id
# 나중에 train과 test 따로 분류할 때 필요한 것들
```

```
merge_data = pd.concat([train,test]).reset_index() # data 병합
data=merge_data.copy()
```

```
test.shape
```

```
(127, 58)
```

EDA

```
data.head()
```

	index	inst_id	OC	sido	sgg	openDate	bedCount	instkind	re
0	0	1	open	choongnam	73	2007-12-28	175.0	nursing_hospital	4.2175
1	1	3	open	gyeongnam	32	1997-04-01	410.0	general_hospital	
2	2	4	open	gyeonggi	89	2016-12-28	468.0	nursing_hospital	1.0045
3	3	7	open	incheon	141	2000-08-14	353.0	general_hospital	7.2507
4	4	9	open	gyeongnam	32	2005-09-01	196.0	general_hospital	4.9043

```
train.shape
```

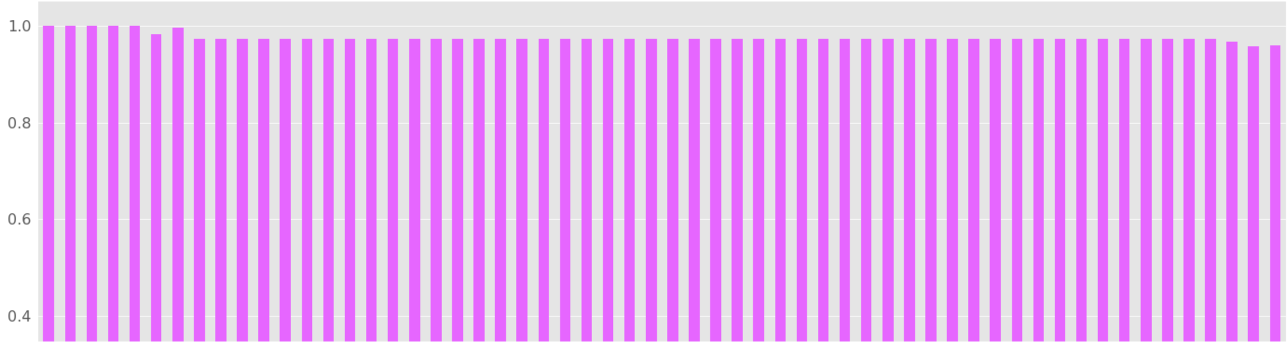
```
(301, 58)
```

```
# 결측치 확인
train.isnull().sum()/train.shape[0]*100
```

```
inst_id               0.000000
OC                    0.000000
sido                  0.000000
sgg                   0.000000
openDate              0.000000
bedCount              1.661130
instkind              0.332226
revenue1              2.657807
salescost1            2.657807
sga1                  2.657807
salary1               2.657807
noi1                  2.657807
noe1                  2.657807
interest1             2.657807
ctax1                 2.657807
profit1               2.657807
liquidAsset1          2.657807
quickAsset1           2.657807
receivableS1          2.657807
inventoryAsset1       2.657807
nonCAsset1            2.657807
tanAsset1             2.657807
OnonCAsset1           2.657807
receivableL1          2.657807
debt1                 2.657807
liquidLiabilities1    2.657807
shortLoan1            2.657807
NCLiabilities1        2.657807
longLoan1             2.657807
netAsset1             2.657807
surplus1              2.657807
revenue2              2.657807
salescost2            2.657807
sga2                  2.657807
salary2               2.657807
noi2                  2.657807
noe2                  2.657807
interest2             2.657807
ctax2                 2.657807
profit2               2.657807
liquidAsset2          2.657807
quickAsset2           2.657807
```

```python
import missingno as msno
msno.bar(train,color=(0.9,0.4,1)) # 결측치가 거의 없음을 알 수 있다.
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdcfc24fba8>
```



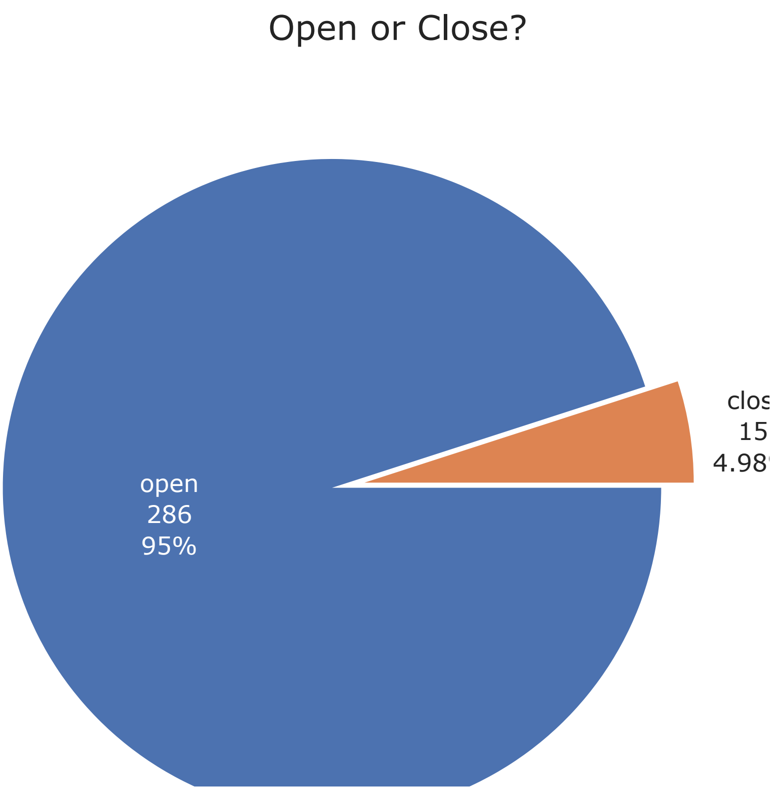
```
data.describe(include='object')
```

	OC	sido	instkind	employee1	employee2	ownerChange
count	301	428	425	410.0	400.0	401
unique	2	17	7	268.0	263.0	2
top	open	gyeonggi	nursing_hospital	73.0	70.0	same
freq	286	68	207	6.0	6.0	348

```
data.describe()
```

	index	inst_id	sgg	bedCount	revenue1	salescost1	
count	428.000000	428.000000	428.000000	415.000000	4.180000e+02	4.180000e+02	4.18
mean	124.184579	215.154206	85.436916	153.474699	1.448543e+10	2.600554e+09	1.12
std	85.469723	124.453370	51.867136	121.469400	2.429913e+10	9.986687e+09	1.68
min	0.000000	1.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	0.00
25%	53.000000	107.750000	37.750000	63.000000	3.238344e+09	0.000000e+00	2.88
50%	106.500000	214.500000	79.500000	143.000000	5.801359e+09	2.055491e+08	5.04
75%	193.250000	322.250000	132.000000	195.000000	1.422214e+10	9.618685e+08	1.14
max	300.000000	431.000000	178.000000	771.000000	1.810000e+11	1.160000e+11	1.27

```
import plotly.express as px # 전체 open close
oc = pd.DataFrame(train['OC'].value_counts())
oc['status']=oc.index
oc.rename(columns={"OC":"count"},inplace=True)
fig=px.pie(oc,values="count",names="status",title="Open or Close?",template='seaborn')
fig.update_traces(rotation=90,pull=0.05,textinfo="value+percent+label")
fig.show()
```



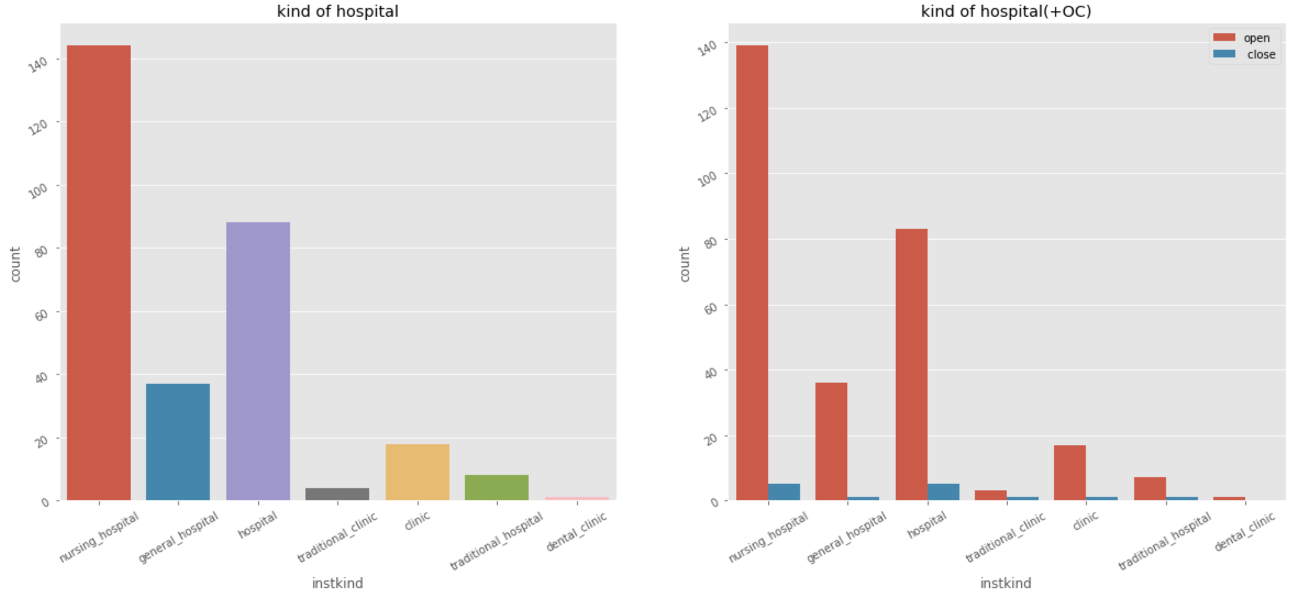
```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,8))
sns.countplot(train['sido'],ax=ax1)
ax1.set(title="Distribution of hospitals by region") #지역별 병원 분포
ax1.tick_params(labelrotation=60)

sns.countplot(train['sido'],hue=train['OC'],ax=ax2)
plt.xticks(rotation=60)
plt.title("Distribution of hospitals by region(+OC)") # 지역별 병원 분포에 open close
```

```
Text(0.5, 1.0, 'Distribution of hospitals by region(+OC)')
```

Distribution of hospitals by region Distribution of hospitals by region(+OC)

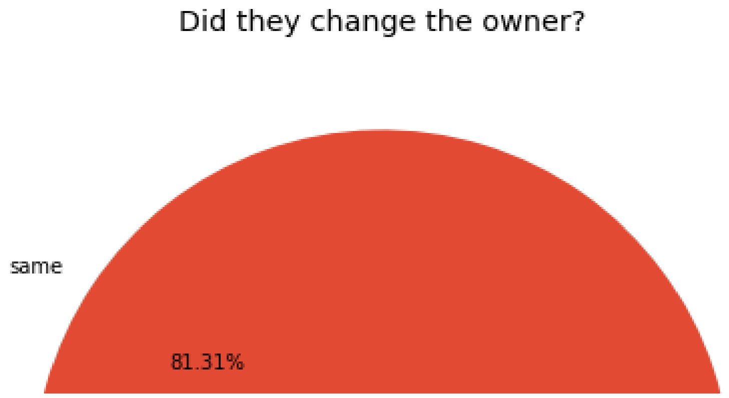
```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,8))
sns.countplot(train['instkind'],ax=ax1)
ax1.set(title='kind of hospital') # 병원 종류
ax1.tick_params(labelrotation=30)
sns.countplot(train['instkind'],hue=train['OC'],ax=ax2)
ax2.set(title='kind of hospital(+OC)') # 병원 종류 + open close
ax2.tick_params(labelrotation=30)
plt.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x7fdcf929d6a0>
```



```
plt.figure(figsize=(12,8))
data['ownerChange'].value_counts(dropna=False).plot(kind='pie',autopct="%.2f%%")
plt.title("Did they change the owner?") # 오너가 바뀌었냐
```

```
Text(0.5, 1.0, 'Did they change the owner?')
```



PreProcessing

```
data['OC'] = data['OC'].apply(lambda x: x.lstrip() if type(x) == str else x) # close가 안이쁘게
```

```
for i in range(data.shape[0]): # employee가 변화 없다고 가정
    if pd.isnull(data['employee1'][i]) != pd.isnull(data['employee2'][i]): # 직원 하나 있고 하ㄴ
        if pd.isnull(data['employee1'][i]) == True:
            data['employee1'][i] = data['employee2'][i]
        elif pd.isnull(data['employee2'][i]) == True:
            data['employee2'][i] = data['employee1'][i]
    else:
        continue
```

```
# 직원 수 변화
def change_type(x):
    if type(x) == str:
        x= x.replace(',','') # ,가 있는게 있음
        return float(x)
    else:
        return x
data['employee1'] = data['employee1'].apply(lambda x: change_type(x))
data['employee2'] = data['employee2'].apply(lambda x: change_type(x))
data['employee_change']=data['employee1']-data['employee2']
```

```
import datetime as dt
data['openYear'] = data['openDate'].dt.year # 오픈 연도만 저장
data.drop('openDate',axis=1,inplace=True) # 자세한 날짜 필요 없으므로 삭제
```

```
debt = ['debt1','debt2','liquidLiabilities1','liquidLiabilities2','shortLoan1','shortLoan2','NCL
        'longLoan1','longLoan2'] # 부채는 모두 마이너스 처리
for i in debt:
    data[i]=data[i].apply(lambda x: 0 if x==0 else -x)
data.head()
```

	index	inst_id	OC	sido	sgg	bedCount	instkind	revenue1	sa
0	0	1	open	choongnam	73	175.0	nursing_hospital	4.217530e+09	
1	1	3	open	gyeongnam	32	410.0	general_hospital	NaN	
2	2	4	open	gyeonggi	89	468.0	nursing_hospital	1.004522e+09	51
3	3	7	open	incheon	141	353.0	general_hospital	7.250734e+10	

PreProcessing

- Processing Missing value

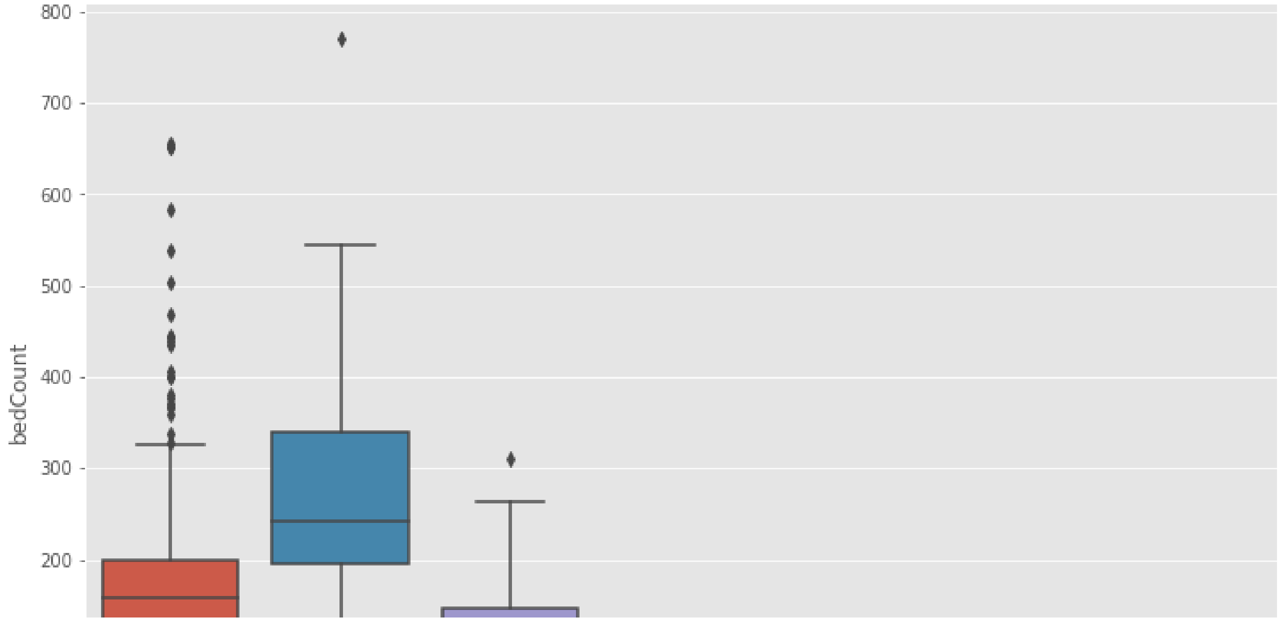
```
data['openYear'].fillna(int(data['openYear'].median()),inplace=True)
```

```
data[data['bedCount'].isnull()][['bedCount','instkind']]
```

	bedCount	instkind
71	NaN	traditional_hospital
193	NaN	NaN
297	NaN	hospital
298	NaN	hospital
300	NaN	traditional_hospital
311	NaN	traditional_clinic
323	NaN	hospital
341	NaN	nursing_hospital
379	NaN	hospital
385	NaN	nursing_hospital
400	NaN	nursing_hospital
424	NaN	traditional_hospital
426	NaN	NaN

```
plt.figure(figsize=(12,8))
sns.boxplot(x=data['instkind'],y=data['bedCount'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdcfc124320>
```



이상치가 많으므로 결측치 대체는 중위값으로 한다.

```
data['bedCount'].fillna(data.groupby('instkind')['bedCount'].transform('median'),inplace=True)
```

```
data[data['bedCount'].isnull()][['bedCount','instkind']] # 둘다 결측치인 것은 영향 안받음
```

```
data[data['instkind'].isnull()][['bedCount','instkind']] # instkind 결측치 확인
```

```
data.groupby('instkind')['bedCount'].median() # bedCount 가 49는 traditional hospital에 가까움ㄷ
```

```python
data['instkind'].iloc[421] = 'traditional_hospital'
```

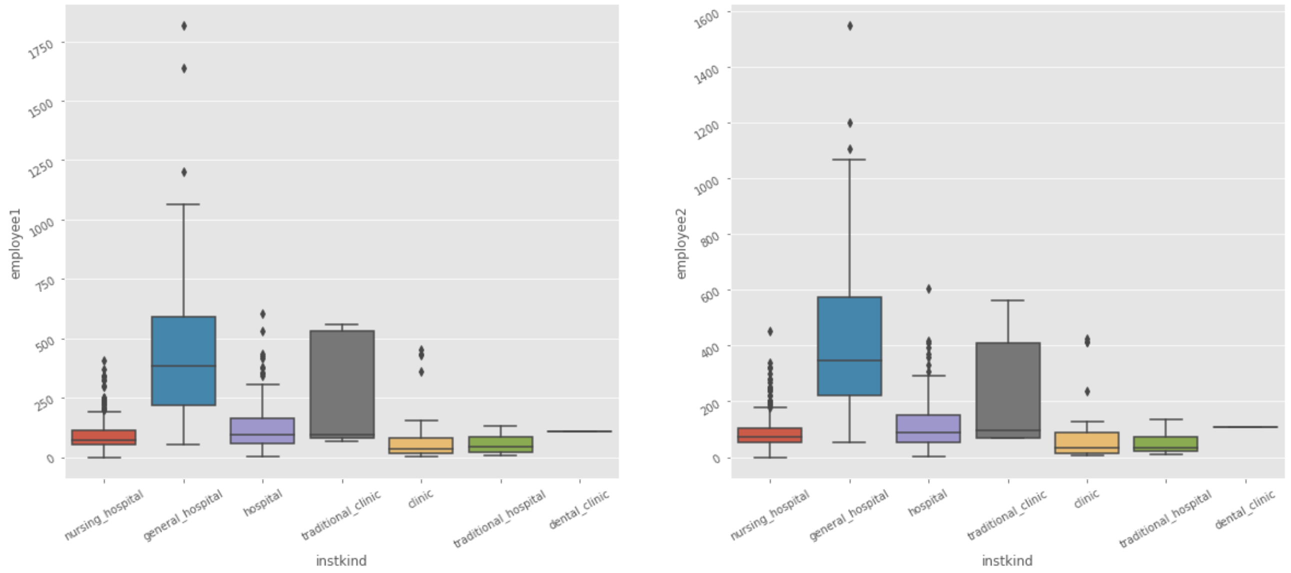
```python
data[['employee1','employee2','bedCount']].corr() # 직원 수와 bedCount의 상관관계가 0.5로 약간
```

	employee1	employee2	bedCount
employee1	1.000000	0.972793	0.505255
employee2	0.972793	1.000000	0.509500
bedCount	0.505255	0.509500	1.000000

```python
# 위 상관계수를 기반으로 직원 수에 따라서 bedCount->bedC0unt 와 instkind 상관관계 높음(데이터 설
data[data['instkind'].isnull()]
```

	index	inst_id	OC	sido	sgg	bedCount	instkind	revenue1	salesc
193	193	281	close	gyeonggi	12	NaN	NaN	3.054388e+08	2.241614
426	125	430	NaN	jeju	76	NaN	NaN	4.892710e+10	4.157148

```python
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(20,8))
sns.boxplot(data=data,x='instkind',y='employee1',ax=ax1)
sns.boxplot(data=data,x='instkind',y='employee2',ax=ax2)
ax1.tick_params(labelrotation=30)
ax2.tick_params(labelrotation=30)
```



```python
data.groupby('instkind')['employee1'].median()
```

```
# 15 = clinc
# 343 = general_hospital
data['instkind'].iloc[193] = 'clinic'
data['instkind'].iloc[426] = 'general_hospital'
data['bedCount'].iloc[193] = 0 # clinc의 중위값으로 대체
data['bedCount'].iloc[426] = 243 # general_hospital 의 중위값으로 대체
```

```
data['ownerChange'].fillna(data['ownerChange'].mode()[0],inplace=True) # 최빈값으로 대체
```

- 직원 수 결측치 처리

```
df=pd.DataFrame(columns=['col','corr'])
for i in range(len(data.corr().keys())):
    df.loc[i,'col'] = data.corr().keys()[i]
    df.loc[i,'corr'] = data.corr()['employee1'][i]
df
```

	col	corr
0	index	-0.235603
1	inst_id	-0.271752
2	sgg	0.0251274
3	bedCount	0.507025
4	revenue1	0.88632
5	salescost1	0.593692
6	sga1	0.868367
7	salary1	0.870879
8	noi1	0.585692
9	noe1	0.621877
10	interest1	0.439253
11	ctax1	0.54814
12	profit1	0.390353
13	liquidAsset1	0.743854
14	quickAsset1	0.740648
15	receivableS1	0.427205
16	inventoryAsset1	0.653327
17	nonCAsset1	0.760217
18	tanAsset1	0.750643
19	OnonCAsset1	0.502299
20	receivableL1	-0.0234059
21	debt1	-0.785368
22	liquidLiabilities1	-0.779534
23	shortLoan1	-0.423056
24	NCLiabilities1	-0.648732
25	longLoan1	-0.317576
26	netAsset1	0.589399
27	surplus1	0.257669
28	revenue2	0.879571
29	salescost2	0.603817
30	sga2	0.86562
31	salary2	0.858302
32	noi2	0.725655

32	noi2	0.725655
33	noe2	0.640054
34	interest2	0.429575
35	ctax2	0.460125
36	profit2	0.424901

```python
corr_features=[]
for i in df[abs(df['corr'])>0.7].reset_index()['col']: # 해당년도의 변수만 고려
    if i[-1] != '2':
        corr_features.append(i)
corr_features
```

```
['revenue1',
 'sga1',
 'salary1',
 'liquidAsset1',
 'quickAsset1',
 'nonCAsset1',
 'tanAsset1',
 'debt1',
 'liquidLiabilities1',
 'employee1']
```

47	shortLoan2	-0.3257

```python
emp = data[corr_features]
train_emp = emp[emp['employee1'].notnull()]
train_emp.dropna(axis=0,inplace=True)
train_emp # 회계데이터가 결측치이면 삭제
```

	revenue1	sga1	salary1	liquidAsset1	quickAsset1	nonCAsse
0	4.217530e+09	3.961135e+09	2.033835e+09	1.012700e+09	9.976719e+08	2.514586e+
2	1.004522e+09	4.472197e+08	2.964023e+08	2.724421e+08	2.536822e+08	1.204810e+
3	7.250734e+10	7.067740e+10	3.178605e+10	1.304154e+10	1.153475e+10	4.317936e+
4	4.904354e+10	4.765605e+10	2.446078e+10	6.317084e+09	5.873265e+09	4.366733e+
5	3.358054e+10	2.372791e+10	1.665533e+10	5.635105e+09	5.481680e+09	1.864970e+
...	...	...	...	...	...	
421	5.583625e+08	5.482900e+08	2.826852e+08	0.000000e+00	0.000000e+00	0.000000e+
422	4.471030e+08	2.581514e+08	1.191270e+08	2.811359e+08	2.336135e+08	2.143994e+
423	2.233031e+10	1.849255e+10	1.232241e+10	1.829292e+10	1.818429e+10	1.307623e+
424	1.833906e+10	1.760117e+10	6.824241e+09	3.706256e+09	3.706256e+09	7.787147e+
426	4.892710e+10	4.721485e+09	1.514547e+09	1.028286e+10	9.002630e+09	2.952030e+

403 rows × 10 columns

```python
test_emp = emp[emp['employee1'].isnull()]
test_emp
```

```
test_emp.drop([test_emp.index[1],test_emp.index[5]],inplace=True)
```

```
train_x1 = train_emp.drop(['employee1'],axis=1)
train_y1 = train_emp['employee1']
test_x1 = test_emp.drop(['employee1'],axis=1)
test_y1 = test_emp['employee1']
```

```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor()
rf.fit(train_x1,train_y1)
pred1 = rf.predict(test_x1)
pred1
```

```
array([103.91  ,  81.69  , 435.4   ,  21.91  ,  61.27  , 149.8275,
       115.36  , 228.09  ,  90.33  , 114.19  , 225.2   , 172.652 ,
        44.24  , 140.02  , 124.75  ])
```

```
df=pd.DataFrame(columns=['col','corr'])
for i in range(len(data.corr().keys())):
    df.loc[i,'col'] = data.corr().keys()[i]
    df.loc[i,'corr'] = data.corr()['employee2'][i]
df
```

	col	corr
0	index	-0.252003
1	inst_id	-0.284224
2	sgg	0.0232896
3	bedCount	0.511327
4	revenue1	0.871015
5	salescost1	0.593576
6	sga1	0.855941
7	salary1	0.847373
8	noi1	0.592255
9	noe1	0.584647
10	interest1	0.449565
11	ctax1	0.487203
12	profit1	0.325222
13	liquidAsset1	0.731837
14	quickAsset1	0.727266
15	receivableS1	0.470095
16	inventoryAsset1	0.676041
17	nonCAsset1	0.785131
18	tanAsset1	0.778555
19	OnonCAsset1	0.480342

```
corr_features2=[]
for i in df[abs(df['corr'])>0.7].reset_index()['col']: # 해당년도의 변수만 고려
    if i[-1] != '1':
        corr_features2.append(i)
corr_features2
```

```
['revenue2',
 'sga2',
 'salary2',
 'noi2',
 'liquidAsset2',
 'quickAsset2',
 'nonCAsset2',
 'tanAsset2',
 'debt2',
 'liquidLiabilities2',
 'employee2']
```

30 sga2 0.859028

```
emp2 = data[corr_features2]
train_emp2 = emp2[emp2['employee2'].notnull()]
```

```
train_emp2.dropna(axis=0,inplace=True)
train_emp2 # 회계데이터가 결측치이면 삭제
```

	revenue2	sga2	salary2	noi2	liquidAsset2	quickAsset
0	4.297848e+09	4.057422e+09	2.063787e+09	16194675.0	8.301695e+08	8.165705e+0
2	0.000000e+00	0.000000e+00	0.000000e+00	0.0	0.000000e+00	0.000000e+0
3	6.685834e+10	6.492419e+10	2.971135e+10	476807804.0	1.112572e+10	9.890540e+0
4	4.808280e+10	4.712580e+10	2.346004e+10	597748128.0	4.906776e+09	4.464017e+0
5	3.433445e+10	2.409622e+10	1.638792e+10	125681154.0	4.869419e+09	4.725857e+0
...	...	...	...	...	...	
421	1.160742e+09	7.614171e+08	4.590994e+08	1045466.0	0.000000e+00	0.000000e+0
422	4.649570e+08	3.153264e+08	7.356901e+07	368343.0	3.813002e+07	0.000000e+0
423	2.239509e+10	1.805503e+10	1.138885e+10	483447584.0	1.708819e+10	1.701879e+1
424	1.911503e+10	1.824717e+10	7.250614e+09	41234195.0	4.588941e+09	4.588941e+0
426	4.758477e+10	5.061219e+09	1.404341e+09	99795507.0	7.848233e+09	6.707102e+0

403 rows × 11 columns

```
test_emp2 = emp2[emp2['employee2'].isnull()]
test_emp2.drop([test_emp2.index[1],test_emp2.index[5]],inplace=True)
```

```
train_x2 = train_emp2.drop(['employee2'],axis=1)
train_y2 = train_emp2['employee2']
test_x2 = test_emp2.drop(['employee2'],axis=1)
test_y2 = test_emp2['employee2']
```

```
rf.fit(train_x2,train_y2)
pred2 = rf.predict(test_x2)
pred2
```

```
idx = list(data[data['employee1'].isnull()].index)
idx.remove(60)
idx.remove(258)
idx
```

```
data['employee1'][idx] = pred1
data['employee2'][idx] = pred2
```

```
data.groupby('instkind')['employee1','employee2','bedCount'].median() # 두개 다 결측치인 것은 병
```

instkind	employee1	employee2	bedCount
clinic	28.0	27.5	0.0
dental_clinic	107.0	109.0	0.0
general_hospital	377.0	346.0	243.0
hospital	96.5	90.0	85.0
nursing_hospital	74.5	74.0	157.5
traditional_clinic	86.0	82.5	0.0
traditional_hospital	42.0	34.0	44.0

```
data['employee1'].iloc[60] = 75
data['employee2'].iloc[60] = 74
data['employee1'].iloc[258] = 28
data['employee2'].iloc[258] = 28
```

```
data['employee1'] = data['employee1'].apply(lambda x: round(x)) # 직원 수 반올림
data['employee2'] = data['employee2'].apply(lambda x: round(x))
```

```
data['employee_change'] = data['employee1'] - data['employee2'] #  update employee_change colum
```

Preprocessing

- 회계데이터 결측치 처리

```
tmp2016=[]
tmp2017=[]
for i in data.columns:
  if i[-1] == '2':
    tmp2016.append(i)
  elif i[-1] == '1':
    tmp2017.append(i)
tmp2016.remove('employee2')
tmp2017.remove('employee1')
acc2016 = data[tmp2016]
acc2017 = data[tmp2017]
```

```
zero_idx2016=[]
zero_idx2017=[]
for i in range(acc2016.shape[0]):
  if sum(acc2016.iloc[i].values==0) == 0:
    zero_idx2016.append(i)
for j in range(acc2017.shape[0]):
  if sum(acc2017.iloc[j].values==0) == 0:
    zero_idx2017.append(j)
print('2016: ',zero_idx2016) # 2016년 회계 데이터 전체가 0인 index들
print('2017: ',zero_idx2017) # 2017년 회계 데이터 전체가 0인 index들
# 회계데이터가 모두 0인 데이터는 없습니다. 나온 결과값 모두 결측치 값입니다.
```

```
2016:  [1, 14, 32, 55, 60, 248, 257, 258, 316, 357]
2017:  [1, 14, 32, 55, 60, 248, 257, 258, 316, 357]
```

```
data['old'] = 2018-data['openYear'] # 2018년 데이터이므로 오픈한지 얼마나 지났는지
```

```
from xgboost import XGBRegressor # xgb로 회계데이터 모델링
xgb = XGBRegressor(n_estimators=1000)
```

```
df= data.drop(['index','inst_id','OC','sido','sgg','ownerChange','openYear','instkind','employee
```

```
idx=list(data.index)
for i in zero_idx2016:
  idx.remove(i)
```

```
na_col = []
for i,j in zip(tmp2016,tmp2017):
  na_col.append(i)
  na_col.append(j)
na_col # 결측치가 있는 데이터 모아놓기
```

```
['revenue2',
 'revenue1',
 'salescost2',
 'salescost1',
 'sga2',
 'sga1',
 'salary2',
 'salary1',
 'noi2',
 'noi1',
 'noe2',
 'noe1',
 'interest2',
 'interest1',
 'ctax2',
 'ctax1',
 'profit2',
 'profit1',
 'liquidAsset2',
 'liquidAsset1',
 'quickAsset2',
```

```
for i in na_col:
  train_x = df.drop(i,axis=1).iloc[idx]
  train_y = df[i].iloc[idx]
  test_x = df.drop(i,axis=1).iloc[zero_idx2016]
  xgb.fit(train_x,train_y)
  pred = xgb.predict(test_x)
  data[i][zero_idx2016] = pred
data.isnull().sum()
```

```
[05:57:11] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:13] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:14] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:15] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:16] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:17] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:19] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:20] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:21] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:24] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:26] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:27] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:28] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:29] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:30] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:32] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:33] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:34] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:35] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:36] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:38] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:39] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:40] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:41] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:42] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:43] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:45] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
[05:57:49] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
```

Preprocessing

- final

```
[05:57:58] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
```

```
data['profit_diff'] = data['profit1'] - data['profit2']
data['debt_diff'] = data['debt1'] - data['debt2']
data['surplus_diff'] = data['surplus1'] - data['surplus2']
data['netasset_diff'] = data['netAsset1'] - data['netAsset2']
```

```
[05:58:05] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprec
```

```
dic={'open':1,'close':0}
data['OC'] = data['OC'].map(dic)
```

```
inst_id                  0
```

```
dic={'same':1,'change':0}
data['ownerChange'] = data['ownerChange'].map(dic)
```

```
sgg                      0
```

```
data.drop(['sgg','openYear','index'],axis=1,inplace=True) # 필요없는 것 다른 것으로 대체할 수 있
```

```
ownerChange              0
```

```
# 충남 충북 -> 충청
# 경남 경북 -> 경상
# 전남 전북 -> 전라
# 경기 -> 경기
# 강원 -> 강원
# 제주 -> 제주
# 인천 부산 대구 울산 세종 대전 광주 ->광역시
```

```
sido_dic={"choongnam":"충청","choongbuk":"충청","gyeongnam":"경상","gyeongbuk":"경상", "jeonnam'
         'seoul':'서울','gangwon':'강원','jeju':'제주',"sejong":"광역시","daejeon":"광역시",
         "busan":"광역시","daegu":"광역시",'ulsan':"광역시","gwangju":'광역시','incheon':"광역ㅅ
data['sido'] = data.sido.map(sido_dic)
```

```
from sklearn.preprocessing import OneHotEncoder # onehotencoding dataframe 변환 함수
def ohe_trans(data,col):
    ohe=OneHotEncoder()
    x= ohe.fit_transform(data[col].values.reshape(-1,1)).toarray()
    tp = []
    for i in range(data[col].unique().size): # onehot 컬럼 생성
        tp.append(col[0]+str(i))
    ohe_df = pd.DataFrame(x,columns = tp)
    return ohe_df

```

```
ohe_sido = ohe_trans(data,'sido')
ohe_kind = ohe_trans(data,'instkind')
data.drop(['sido','instkind'],axis=1,inplace=True)
```

```
data=pd.concat([data,ohe_sido,ohe_kind],axis=1)
```

```
train_data = data.iloc[:train_len]
test_data = data.iloc[train_len:]
```

```
train_data.drop('inst_id',axis=1,inplace=True)
test_data.drop('inst_id',axis=1,inplace=True)
```

MODELING

- use ML(XGBCLASSIFIER,RANDOMFORESTCLASSIFIER)

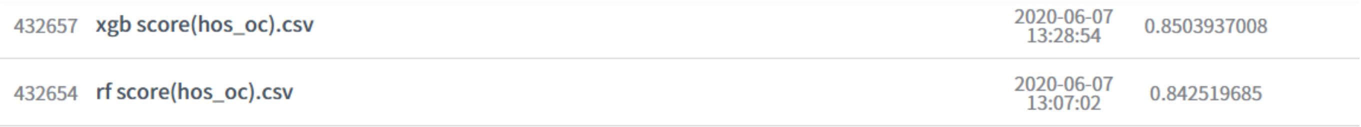
```
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
rf = RandomForestClassifier(n_estimators = 1000)
xgb = XGBClassifier(n_estimators = 1000)
```

```
train_x = train_data.drop('OC',axis=1)
train_y = train_data['OC']
test_x = test_data.drop('OC',axis=1)
test_y = test_data['OC']
test_y = test_data['OC']
```

```
rf.fit(train_x,train_y)
pred_rf = rf.predict(test_x)
xgb.fit(train_x,train_y)
pred_xgb = xgb.predict(test_x)
```

```
answer['OC'] = pred_rf
#answer.to_csv('rf_predict(hos oc).csv',index=False)
```

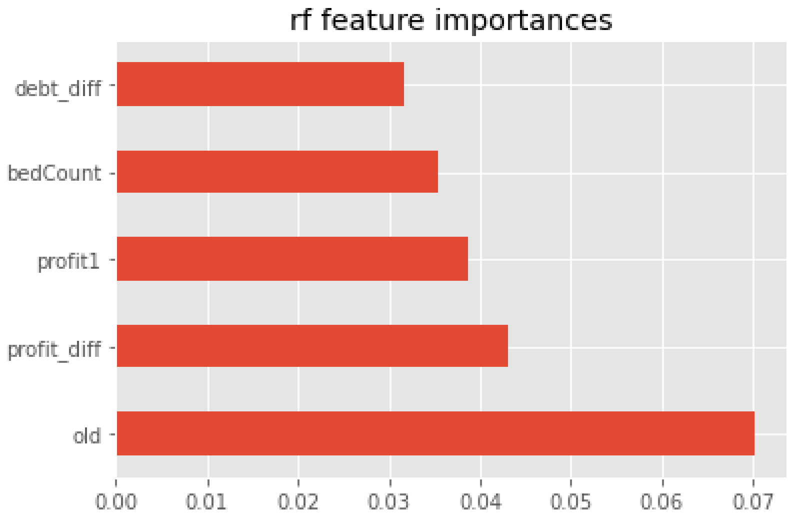
```
answer['OC'] = pred_xgb
#answer.to_csv('xgb_predict(hos oc).csv',index=False)
```



432657	xgb score(hos_oc).csv	2020-06-07 13:28:54	0.8503937008
432654	rf score(hos_oc).csv	2020-06-07 13:07:02	0.842519685

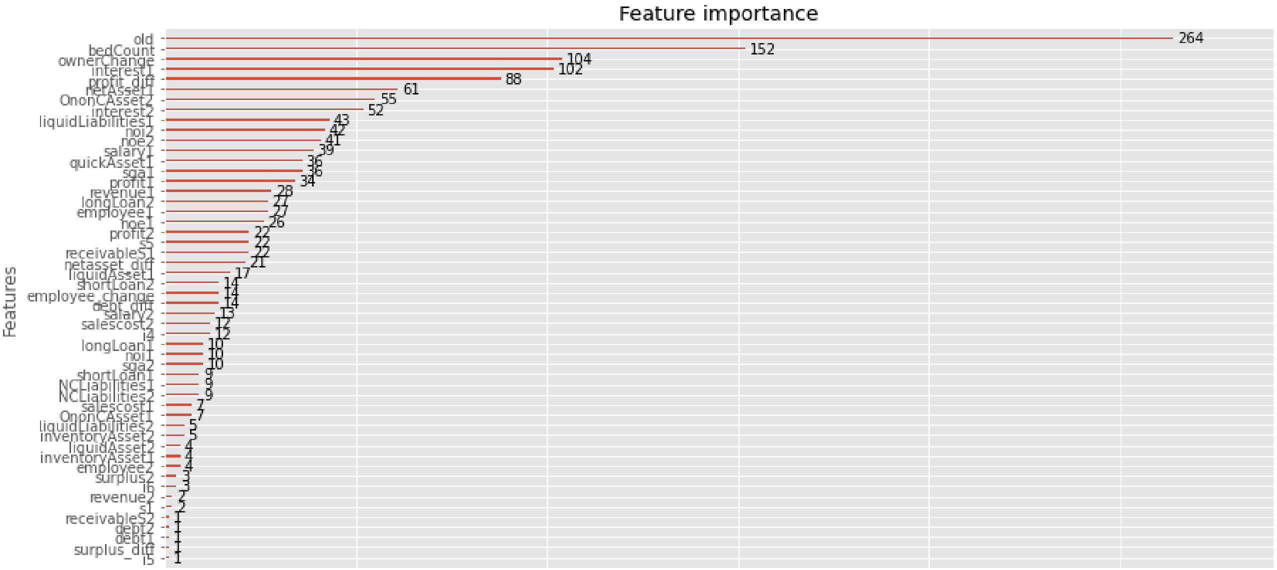
```
feat_importance=pd.Series(rf.feature_importances_,index=train_x.columns)
feat_importance.nlargest(5).plot(kind='barh')
plt.title('rf feature importances')
```

```
Text(0.5, 1.0, 'rf feature importances')
```



```
from xgboost import plot_importance
plt.rcParams["figure.figsize"] = (14, 7)
plot_importance(xgb)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc43231b70>
```



- 두 모델 공통적으로 old 변수가 OC를 가장 판단하는데 가장 중요한 변수로 작용하였다. 이어서 profit_diff,bedCount 등의 변수가 예측하는데 중요한 변수로 되었다.

MODELING

- 인공신경망

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.init as init

# + CNN할때 필요한 것
import torch.utils.data as data
import torchvision.datasets as dset
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
```

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

```
train_df = train_data.drop(['OC'],axis=1)
train_OC = train_data['OC']
```

```
from sklearn.model_selection import train_test_split # 검정용 데이터 생성
valid_train_x,valid_x,valid_train_y,valid_y = train_test_split(train_df,train_OC,test_size=.2,ra

valid_train_tensor = torch.tensor(valid_train_x.values,dtype=torch.float) # 검정용 데이터 train
valid_train_tensor_y = torch.tensor(valid_train_y.values,dtype=torch.float) # 검정용 데이터 tra

train_tensor = valid_train_tensor.type(torch.FloatTensor).to(device)
train_tensor_y = valid_train_tensor_y.type(torch.LongTensor).to(device)

valid_test_tensor = torch.tensor(valid_x.values,dtype=torch.float).to(device)
```

```
num_epoch = 10000
lr = [0.005,0.002,0.0005,0.0002]
s_lr = [0.05,0.02,0.005,0.002] # 시그모이드는 학습률 크게
```

```
# 활성화 함수를 변경 시키면서 성능 확인 필요

w = 73  # input의 개수
model = nn.Sequential(
          nn.Linear(1*w,6*w),
          nn.ReLU(),
          nn.Linear(6*w,10*w),
          nn.ReLU(),
          nn.Linear(10*w,6*w),
          nn.ReLU(),
          nn.Linear(6*w,1*w),
          nn.LeakyReLU(),
          nn.Linear(w,2),
      )

loss_func = nn.CrossEntropyLoss()
# optimizer = optim.Adam(model.parameters(),lr=0.002)
#optimizer2 = optim.Adam(model.parmeters(),lr=0.0002)
model.to(device)
```

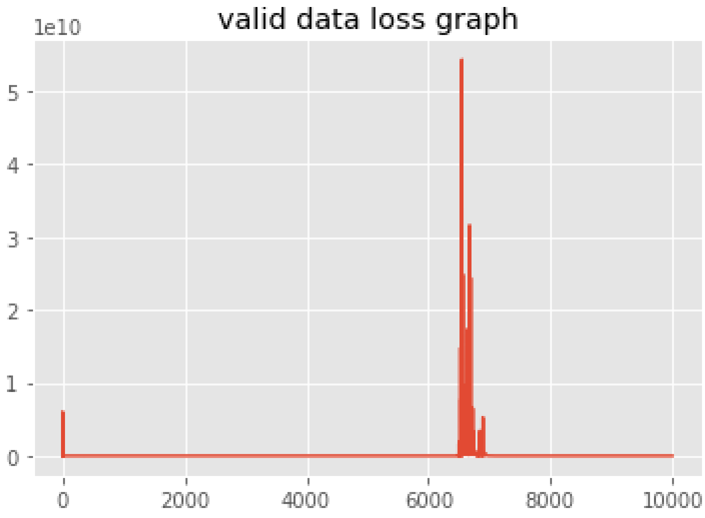
```
# 모델 함수로 만들어서 계속 초기화 할 수 있도록

def model_RRRLR():
    w = 73  # input의 개수
    model = nn.Sequential(
          nn.Linear(1*w,6*w), nn.ReLU(),
          nn.Linear(6*w,10*w), nn.ReLU(),
          nn.Linear(10*w,6*w), nn.ReLU(),
          nn.Linear(6*w,1*w), nn.LeakyReLU(),
          nn.Linear(w,2) )
    loss_func = nn.CrossEntropyLoss()
    return model.to(device)

def model_RRRR():
    w = 73  # input의 개수
    model = nn.Sequential(
          nn.Linear(1*w,6*w), nn.ReLU(),
          nn.Linear(6*w,10*w), nn.ReLU(),
          nn.Linear(10*w,6*w), nn.ReLU(),
          nn.Linear(6*w,1*w), nn.ReLU(),
          nn.Linear(w,2) )
    loss_func = nn.CrossEntropyLoss()
    return model.to(device)

def model_RRRT():
    w = 73  # input의 개수
    model = nn.Sequential(
          nn.Linear(1*w,6*w), nn.ReLU(),
          nn.Linear(6*w,10*w), nn.ReLU(),
          nn.Linear(10*w,6*w), nn.ReLU(),
          nn.Linear(6*w,1*w), nn.Tanh(),
          nn.Linear(w,2) )
    loss_func = nn.CrossEntropyLoss()
    return model.to(device)

def model_SSSS():
    w = 73  # input의 개수
    model = nn.Sequential(
          nn.Linear(1*w,6*w), nn.Sigmoid(),
          nn.Linear(6*w,10*w), nn.Sigmoid(),
          nn.Linear(10*w,6*w), nn.Sigmoid(),
          nn.Linear(6*w,1*w), nn.Sigmoid(),
          nn.Linear(w,2) )
    loss_func = nn.CrossEntropyLoss()
    return model.to(device)

def model_TTTT():
    w = 73  # input의 개수
    model = nn.Sequential(
          nn.Linear(1*w,6*w), nn.Tanh(),
          nn.Linear(6*w,10*w), nn.Tanh(),
          nn.Linear(10*w,6*w), nn.Tanh(),
          nn.Linear(6*w,1*w), nn.Tanh(),
          nn.Linear(w,2) )
    loss_func = nn.CrossEntropyLoss()
    return model.to(device)

def model_LRLRLRLR():
    w = 73  # input의 개수
    model = nn.Sequential(
          nn.Linear(1*w,6*w), nn.LeakyReLU(),
          nn.Linear(6*w,10*w), nn.LeakyReLU(),
          nn.Linear(10*w,6*w), nn.LeakyReLU(),
          nn.Linear(6*w,1*w), nn.LeakyReLU(),
          nn.Linear(w,2) )
    loss_func = nn.CrossEntropyLoss()
    return model.to(device)
```

ReLU - ReLU - RELU - LeakyReLU

```
loss_array_array = []
model_array = []
for r in lr:
    loss_array = []
    model = model_RRRLR()
    optimizer = optim.Adam(model.parameters(),r)
    print("학습률 : {}".format(r))
    for i in range(num_epoch):
        optimizer.zero_grad()
        output = model(train_tensor).to(device)
        loss = loss_func(output,train_tensor_y)
        if i%1000==0:
            print(loss)
        loss.backward()
        optimizer.step()
        loss_array.append(loss)
    model_array.append(model)
    loss_array_array.append(loss_array)
```

```
학습률 : 0.005
tensor(1828784.7500, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(96.9898, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(37.1877, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(147.8693, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(88.9446, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(161.2969, device='cuda:0', grad_fn=<NllLossBackward>)
```

```
for loss_array in loss_array_array:
    plt.plot(loss_array)
    plt.title("valid data loss graph")
    plt.show()
```



```
output_array = []
for m in model_array:
    output = m(valid_test_tensor)
    output_array.append(output)
# output
```

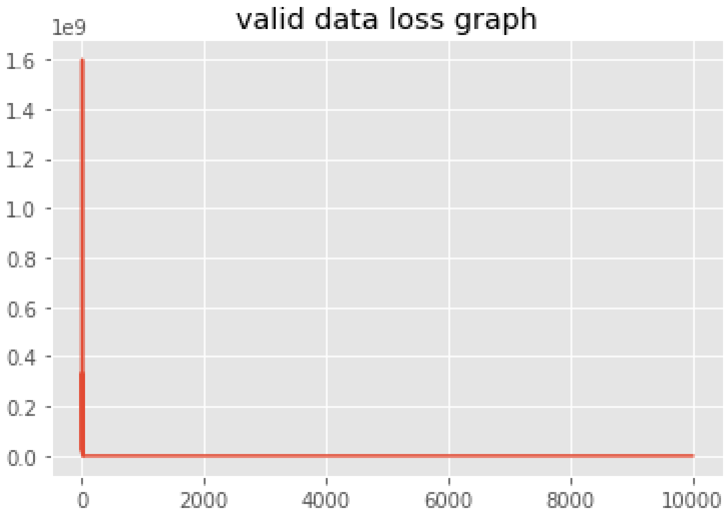
```
valid_result_array=[]
for output in output_array:
    valid_result = []
    for i in output:
        if i.argmax() == 1:
            valid_result.append(1)
        else:
            valid_result.append(0)
    valid_result_array.append(valid_result)
```

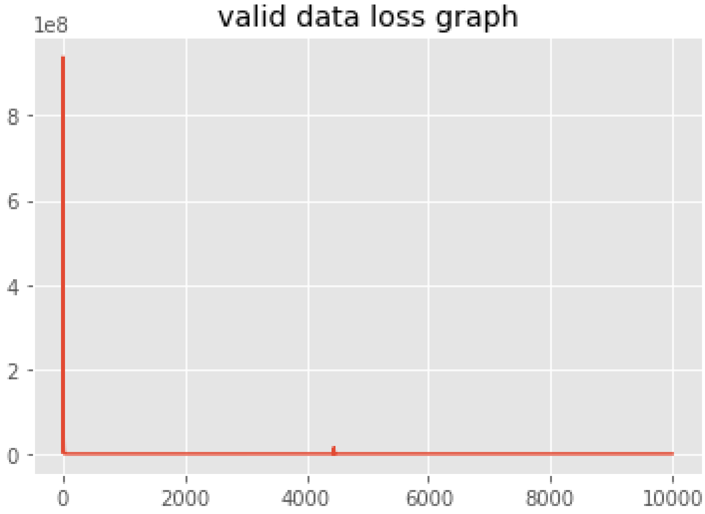
```
from sklearn.metrics import accuracy_score # 정확도 측정
for valid_result in valid_result_array:
    print(accuracy_score(valid_y,valid_result)) # 오류나면 데이터type 통일해주기
```

ReLU - ReLU - ReLU - ReLU

```
loss_array_array = []
model_array = []
for r in lr:
    loss_array = []
    model = model_RRRR()
    optimizer = optim.Adam(model.parameters(),r)
    print("학습률 : {}".format(r))
    for i in range(num_epoch):
        optimizer.zero_grad()
        output = model(train_tensor).to(device)
        output = model(train_tensor).to(device)
        loss = loss_func(output,train_tensor_y)
        if i%5000==0:
            print(loss)
        loss.backward()
        optimizer.step()
        loss_array.append(loss)
    model_array.append(model)
    loss_array_array.append(loss_array)
```

```
for loss_array in loss_array_array:
    plt.plot(loss_array)
    plt.title("valid data loss graph")
    plt.show()
```





```
output_array = []
for m in model_array:
    output = m(valid_test_tensor)
    output_array.append(output)
```

```
valid_result_array=[]
for output in output_array:
    valid_result = []
    for i in output:
        if i.argmax() == 1:
            valid_result.append(1)
        else:
            valid_result.append(0)
    valid_result_array.append(valid_result)
```

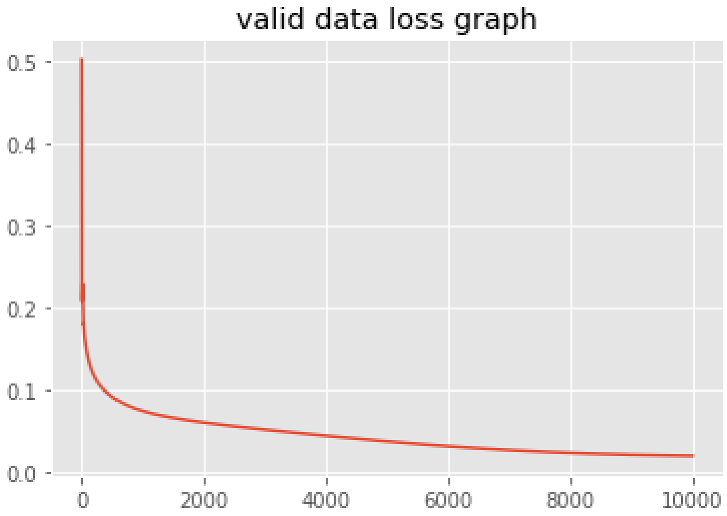
```
for valid_result in valid_result_array:
    print(accuracy_score(valid_y,valid_result)) # 오류나면 데이터type 통일해주기
```

```
0.9672131147540983
0.9508196721311475
0.9672131147540983
0.9672131147540983
```

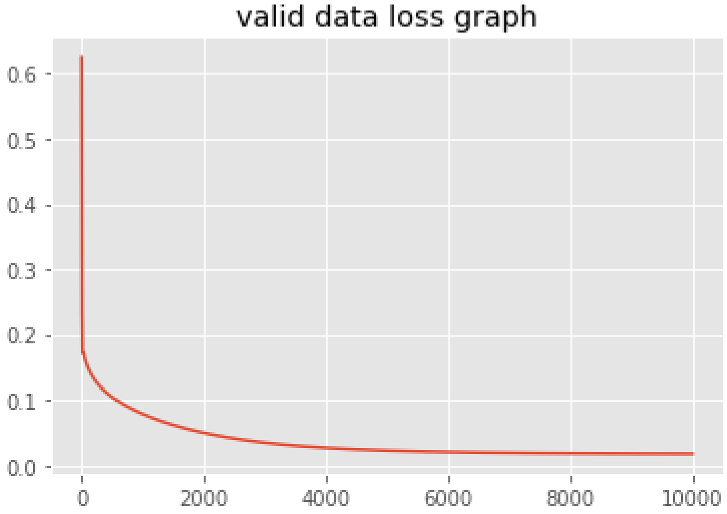
ReLU - ReLU - ReLU - Tanh

```
loss_array_array = []
model_array = []
for r in lr:
    loss_array = []
    model = model_RRRT()
    optimizer = optim.Adam(model.parameters(),r)
    print("학습률 : {}".format(r))
    for i in range(num_epoch):
        optimizer.zero_grad()
        output = model(train_tensor).to(device)
        loss = loss_func(output,train_tensor_y)
        if i%5000==0:
            print(loss)
        loss.backward()
        optimizer.step()
        loss_array.append(loss)
    model_array.append(model)
    loss_array_array.append(loss_array)
```

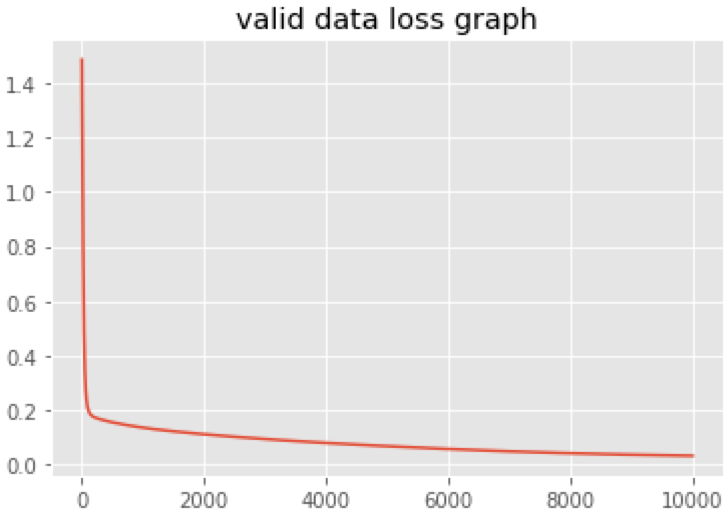
```
for loss_array in loss_array_array:
    plt.plot(loss_array)
    plt.title("valid data loss graph")
    plt.show()
```



valid data loss graph

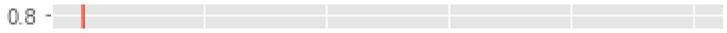


valid data loss graph



valid data loss graph

```
output_array = []
for m in model_array:
    output = m(valid_test_tensor)
    output_array.append(output)
```



```
valid_result_array=[]
for output in output_array:
    valid_result = []
    for i in output:
        if i.argmax() == 1:
            valid_result.append(1)
        else:
            valid_result.append(0)
    valid_result_array.append(valid_result)
```

```
for valid_result in valid_result_array:
    print(accuracy_score(valid_y,valid_result)) # 오류나면 데이터type 통일해주기
```

Sigmoid - Sigmoid - Sigmoid - Sigmoid

```
loss_array_array = []
model_array = []
for r in s_lr:
    loss_array = []
    model = model_RRRT()
    optimizer = optim.Adam(model.parameters(),r)
    print("학습률 : {}".format(r))
    for i in range(num_epoch):
        optimizer.zero_grad()
        output = model(train_tensor).to(device)
        loss = loss_func(output,train_tensor_y)
        if i%5000==0:
            print(loss)
        loss.backward()
        optimizer.step()
        loss_array.append(loss)
    model_array.append(model)
    loss_array_array.append(loss_array)
```

```
for loss_array in loss_array_array:
    plt.plot(loss_array)
    plt.title("valid data loss graph")
    plt.show()
```

```
output_array = []
for m in model_array:
    output = m(valid_test_tensor)
    output_array.append(output)
```

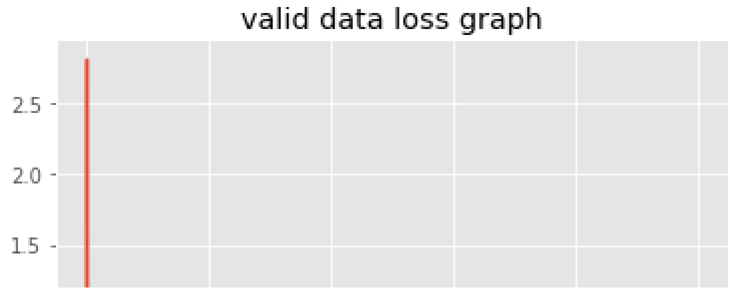
```
valid_result_array=[]
for output in output_array:
    valid_result = []
    for i in output:
        if i.argmax() == 1:
            valid_result.append(1)
        else:
            valid_result.append(0)
    valid_result_array.append(valid_result)
```

```
for valid_result in valid_result_array:
    print(accuracy_score(valid_y,valid_result)) # 오류나면 데이터type 통일해주기
```

Tanh - Tanh - Tanh - Tanh

```
loss_array_array = []
model_array = []
for r in lr:
    loss_array = []
    model = model_TTTT()
    optimizer = optim.Adam(model.parameters(),r)
    print("학습률 : {}".format(r))
    for i in range(num_epoch):
        optimizer.zero_grad()
        output = model(train_tensor).to(device)
        loss = loss_func(output,train_tensor_y)
        if i%5000==0:
            print(loss)
        loss.backward()
        optimizer.step()
        loss_array.append(loss)
    model_array.append(model)
    loss_array_array.append(loss_array)
```

```
for loss_array in loss_array_array:
    plt.plot(loss_array)
    plt.title("valid data loss graph")
    plt.show()
```



```
output_array = []
for m in model_array:
    output = m(valid_test_tensor)
    output_array.append(output)
```

```
valid_result_array=[]
for output in output_array:
    valid_result = []
    for i in output:
        if i.argmax() == 1:
            valid_result.append(1)
        else:
            valid_result.append(0)
    valid_result_array.append(valid_result)
```

```
for valid_result in valid_result_array:
    print(accuracy_score(valid_y,valid_result)) # 오류나면 데이터type 통일해주기
```

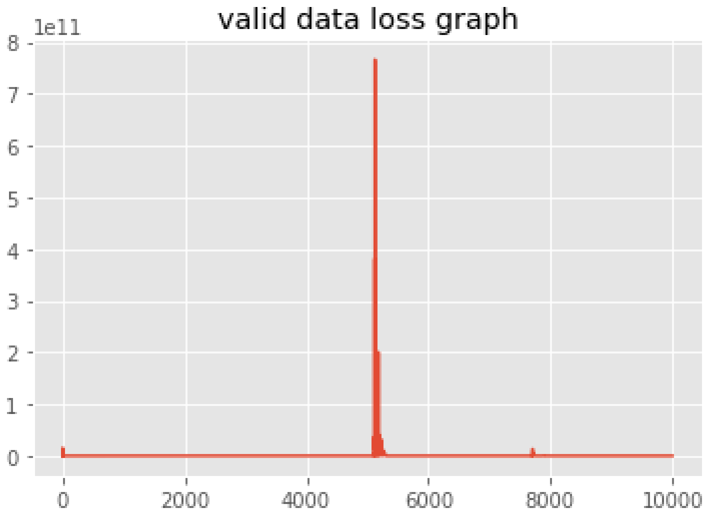
```
0.9180327868852459
0.9508196721311475
0.9344262295081968
0.9508196721311475
```

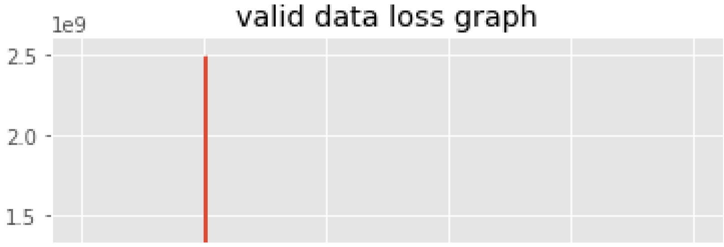
LeakyReLU - LeakyReLU - LeakyReLU - LeakyReLU

```
loss_array_array = []
model_array = []
for r in lr:
    loss_array = []
    model = model_LRLRLRLR()
    optimizer = optim.Adam(model.parameters(),r)
    print("학습률 : {}".format(r))
    for i in range(num_epoch):
        optimizer.zero_grad()
        output = model(train_tensor).to(device)
        loss = loss_func(output,train_tensor_y)
        if i%5000==0:
            print(loss)
        loss.backward()
        optimizer.step()
        loss_array.append(loss)
    model_array.append(model)
    loss_array_array.append(loss_array)
    loss_array_array.append(loss_array)
```

```
학습률 : 0.005
tensor(2566436.2500, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(267.5387, device='cuda:0', grad_fn=<NllLossBackward>)
학습률 : 0.002
tensor(11951636., device='cuda:0', grad_fn=<NllLossBackward>)
tensor(902.5310, device='cuda:0', grad_fn=<NllLossBackward>)
학습률 : 0.0005
tensor(14195371., device='cuda:0', grad_fn=<NllLossBackward>)
tensor(1443517.8750, device='cuda:0', grad_fn=<NllLossBackward>)
학습률 : 0.0002
tensor(2753409.5000, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(2489.4126, device='cuda:0', grad_fn=<NllLossBackward>)
```

```
for loss_array in loss_array_array:
    plt.plot(loss_array)
    plt.title("valid data loss graph")
    plt.show()
```





```
output_array = []
for m in model_array:
    output = m(valid_test_tensor)
    output_array.append(output)
```

```
valid_result_array=[]
for output in output_array:
    valid_result = []
    for i in output:
        if i.argmax() == 1:
            valid_result.append(1)
        else:
            valid_result.append(0)
    valid_result_array.append(valid_result)
```

```
for valid_result in valid_result_array:
    print(accuracy_score(valid_y,valid_result)) # 오류나면 데이터type 통일해주기
```

```
0.9344262295081968
0.9508196721311475
0.9672131147540983
0.9672131147540983
```

정확도 측정 결과

	RRRLR	RRRR	RRRT	TTTT	LRLRLRLR		SSSS
0.005	0.93	0.95	0.86	0.967	0.93	0.05	0.91

Modeling test data

0.0002	0.967	0.95	0.868	0.95	0.967	0.002	0.91

```

train_df_tensor = torch.tensor(train_df.values,dtype=torch.float)
train_data_Output = torch.tensor(train_OC.values, dtype=torch.float)
test_df = test_data.drop('OC',axis=1)
test_df_tensor = torch.tensor(test_df.values,dtype=torch.float)
```

```
num_epoch = 10000

# train_tensor = train_tensor.type(torch.FloatTensor)
#train_tensor = train_tensor.cuda() # GPU로 보냄
#noise = init.normal_(train_tensor,std=1)
# x = init.uniform_(train_tensor,-15,15)
lr = [0.005,0.002,0.0005,0.0002]
```

```
# # 아래 코드는 특성의 개수가 73 -> 6*73 -> 10*73 -> 6*73 -> 73 -> 1개로 변하는 인공신경망입니다

# w = 73  # input의 개수
# model = nn.Sequential(
#           nn.Linear(1*w,6*w),
#           nn.ReLU(),
#           nn.Linear(6*w,10*w),
#           nn.ReLU(),
#           nn.Linear(10*w,6*w),
#           nn.ReLU(),
#           nn.Linear(6*w,1*w),
#           nn.LeakyReLU(),
#           nn.Linear(w,2),
#       )

# loss_func = nn.CrossEntropyLoss()
# optimizer = optim.Adam(model.parameters(),lr=0.002)
# #optimizer2 = optim.Adam(model.parmeters(),lr=0.0002)
# model.to(device)
```

```
final_train_tensor = torch.tensor(train_df.values,dtype=torch.float).to(device)
train_y_tensor = torch.tensor(train_OC.values,dtype=torch.float).to(device)
final_train_tensor = final_train_tensor.type(torch.FloatTensor).to(device)
train_y_tensor = train_y_tensor.type(torch.LongTensor).to(device)
```

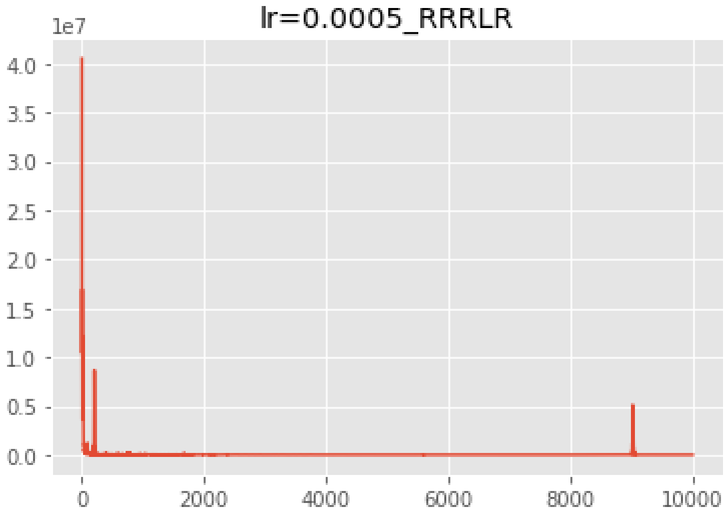
```
test_x = test_data.drop('OC',axis=1)
test_tensor = torch.tensor(test_x.values,dtype=torch.float).to(device)
```

```
model = model_RRRLR()
loss_func = nn.CrossEntropyLoss()
```

```
optimizer = optim.Adam(model.parameters(),lr=0.0005)
model.to(device)
loss_array = []
for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(final_train_tensor).to(device)
    loss = loss_func(output,train_y_tensor)
    if i%1000==0:
        print(loss)
    loss.backward()
    optimizer.step()
    loss_array.append(loss)
```

```
tensor(40525448., device='cuda:0', grad_fn=<NllLossBackward>)
tensor(5737.2456, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(19227.3457, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(8675.5332, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(2017.1469, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(958.8677, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(486.0355, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(70.5464, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(2.4756, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(14675.6328, device='cuda:0', grad_fn=<NllLossBackward>)
```

```
# 손실 그래프
plt.plot(loss_array)
plt.title("lr=0.0005_RRRLR")
plt.show()
print(loss_array[-1])
```



```
tensor(218.8876, device='cuda:0', grad_fn=<NllLossBackward>)
```

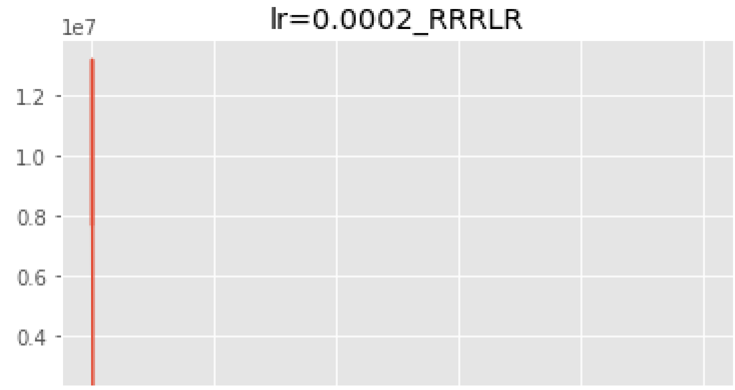
```
model = model_RRRLR()
loss_func = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),lr=0.0002)
model.to(device)
loss_array = []
for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(final_train_tensor).to(device)
    loss = loss_func(output,train_y_tensor)
    if i%1000==0:
        print(loss)
    loss.backward()
    optimizer.step()
    loss_array.append(loss)
```

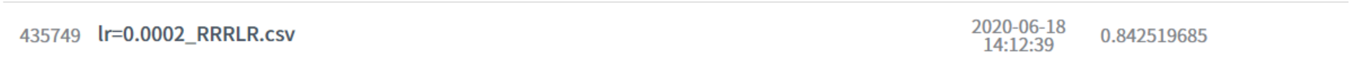
```
tensor(7725928.5000, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(7151.9966, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(31342.8203, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(5864.6211, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(3355.1101, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(1041.5259, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(2898.9993, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(2320.8613, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(813.8175, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(306.7185, device='cuda:0', grad_fn=<NllLossBackward>)
```

```
model = model_RRRR()
loss_func = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),lr=0.0005)
model.to(device)
loss_array = []
for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(final_train_tensor).to(device)
    loss = loss_func(output,train_y_tensor)
    if i%1000==0:
        print(loss)
    loss.backward()
    optimizer.step()
    loss_array.append(loss)
```

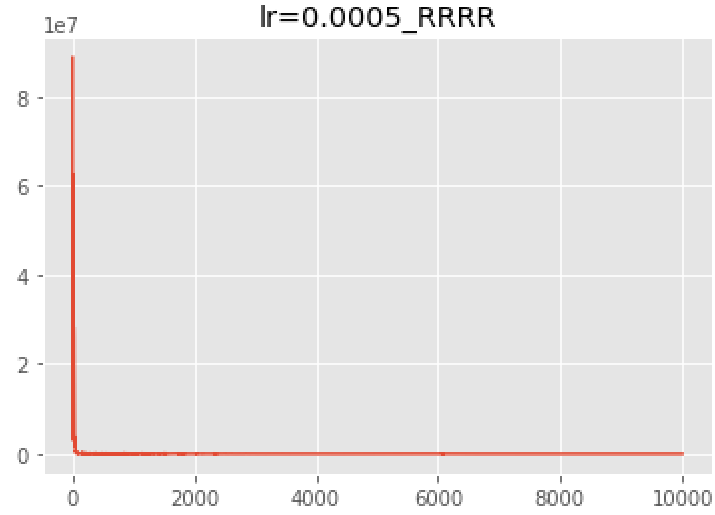
```
tensor(88838728., device='cuda:0', grad_fn=<NllLossBackward>)
tensor(6955.7974, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(5976.5698, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(7903.1162, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(5816.8589, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(621.4550, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(178.6750, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(140.7881, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(57.2482, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.8540, device='cuda:0', grad_fn=<NllLossBackward>)
```

```
# 손실 그래프
plt.plot(loss_array)
plt.title("lr=0.0002_RRRLR")
plt.show()
print(loss_array[-1])
```

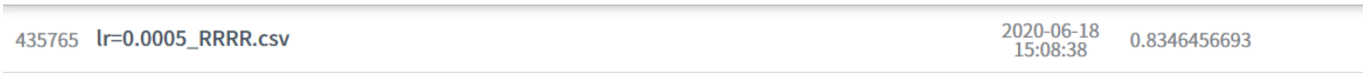



435749 lr=0.0002_RRRLR.csv 2020-06-18 14:12:39 0.842519685

```
plt.plot(loss_array)
plt.title("lr=0.0005_RRRR")
plt.show()
print(loss_array[-1])
```

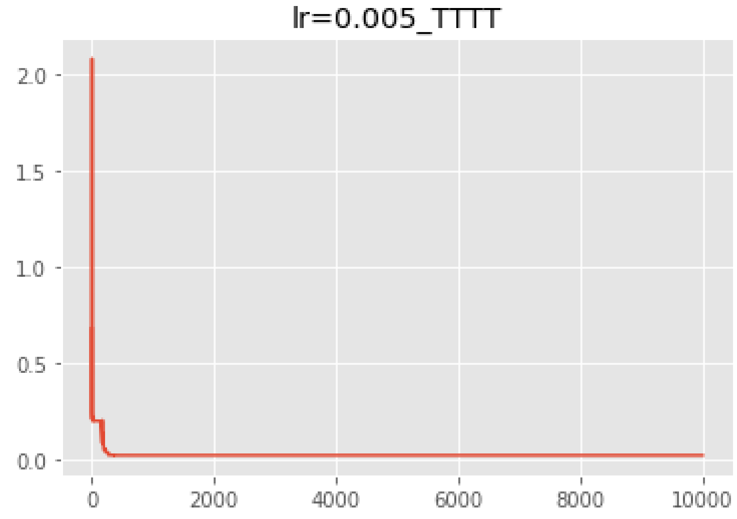


```
tensor(0.1018, device='cuda:0', grad_fn=<NllLossBackward>)
```

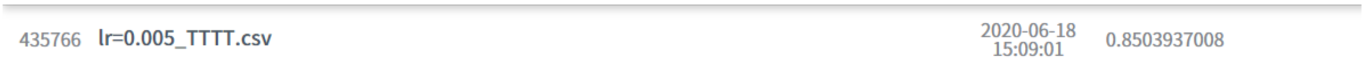

435765 lr=0.0005_RRRR.csv 2020-06-18 15:08:38 0.8346456693

```
model = model_TTTT()
loss_func = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),lr=0.005)
model.to(device)
loss_array = []
for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(final_train_tensor).to(device)
    loss = loss_func(output,train_y_tensor)
    if i%1000==0:
        print(loss)
    loss.backward()
    optimizer.step()
    loss_array.append(loss)
plt.plot(loss_array)
plt.title("lr=0.005_TTTT")
plt.show()
print(loss_array[-1])
```

```
tensor(0.6759, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0185, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
```



```
tensor(0.0184, device='cuda:0', grad_fn=<NllLossBackward>)
```



435766 lr=0.005_TTTT.csv 2020-06-18 15:09:01 0.8503937008

```
model = model_LRLRLRLR()
loss_func = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),lr=0.0002)
model.to(device)
loss_array = []
for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(final_train_tensor).to(device)
    loss = loss_func(output,train_y_tensor)
    if i%1000==0:
        print(loss)
    loss.backward()
    optimizer.step()
    loss_array.append(loss)

plt.plot(loss_array)
plt.title("lr=0.0002_LRLRLRLR")
plt.show()
print(loss_array[-1])
```