빅데이터 통계분석
final project

소프트웨어융합전공
2015682 이보경

1. READ TRAIN DATA

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("final").getOrCreate()
spark.conf.set("spark.sql.execution.arrow.pyspark.enabled","true")
```

```python
part1 = spark.read.parquet("hdfs://localhost:9000/finalDB/Partial1.parquet")
part2 = spark.read.parquet("hdfs://localhost:9000/finalDB/Partial2.parquet")
part3 = spark.read.option('header', 'true').csv("hdfs://localhost:9000/finalDB/Partial3.csv", inferSchema=True)
part4 = spark.read.parquet("hdfs://localhost:9000/finalDB/Partial4.parquet")
part5 = spark.read.option('header', 'true').csv("hdfs://localhost:9000/finalDB/Partial5.csv")
```

```python
part1.show()
```

ID	X1	X2	X3	X4	X5	Y
P1000002	A3	0.1972	6.563	8.4542	Group3	27612.3830690663
P1000007	A1	-0.9084	10.491	9.8017	Group2	9150.26570738822
P1000030	A3	0.3164	11.341	14.8871	Group2	76579.6846859015
P1000046	A2	0.7928	11.831	8.7676	Group2	51189.603726392
P1000060	A1	0.4274	17.689	10.6112	Group2	148313.339805297
P1000064	A1	-1.5066	19.845	9.7027	Group2	61074.026560969
P1000083	A1	1.1936	3.574	7.8773	Group1	50632.088794274
P1000084	A2	-1.1741	4.27	10.4408	Group2	3065.22776267054
P1000087	A1	-0.3164	3.105	13.7127	Group1	6159.11438564402
P1000129	A2	-0.4274	-0.509	null	Group2	159393.020826127

```python
part3.show()
```

ID	X1	X2	X3	X4	X5	Y
P5296770	A1	-0.1934	26.697	NA	Group2	83223.5089882021
P5296771	A2	0.068	10.071	6.1157	Group1	43143.3482288991
P5296777	A2	-1.7123	3.661	9.493	Group2	1550.53998250517
P5296784	A4	0.1998	19.85	9.3692	Group2	7572.07919410697
P5296787	A2	-1.4737	18.933	12.1682	Group2	5895.67818712711
P5296792	A3	-0.7172	23.318	5.7581	Group2	591.379519815538
P5296797	A3	0.9106	9.803	9.5171	Group1	84390.077175249
P5296801	A4	-0.6138	9.279	6.969	Group3	22383.329163738
P5296804	A4	0.9997	13.654	7.841	Group2	387864.997228424
P5296806	A2	0.1604	0.877	10.7047	Group2	314654.491913031

2. COMBINE & toPandas

```
Sdata = part1.union(part2)
```

```
Sdata = Sdata.union(part3)
```

```
Sdata = Sdata.join(part4, ['ID'], 'inner')
```

```
Sdata = Sdata.unionByName(part5)
```

```
Sdata.show(5)
```

```
+--------+---+-------+------+-------+------+------------------+----+----+--------+---+------+
|      ID| X1|     X2|    X3|     X4|    X5|                 Y|  X6|  X7|      X8| X9|   X10|
+--------+---+-------+------+-------+------+------------------+----+----+--------+---+------+
|P1000342| A4|-0.4945|24.935|10.6905|Group2|8680.83289773887| 5.0|Age4|  12.075|102|  Male|
|P1000864| A4| 1.7582| 8.017| 9.9748|Group2|815078.562414822|null|Age5| 9.96618| 88|Female|
|P1005777| A3|-0.9239|11.739| 7.2338|Group1|106371.230243878| 6.0|Age6|  9.5947| 85|Female|
|P1006134| A1|-0.6068|-7.772|13.1339|Group2|10852.2018059725| 5.0|Age3| 9.29497|114|  Male|
|P1006412| A4| 0.7677| 2.398|10.2349|Group2| 34034.552569579| 5.0|Age5|12.10124| 91|Female|
+--------+---+-------+------+-------+------+------------------+----+----+--------+---+------+
only showing top 5 rows
```

```
data = Sdata.toPandas()
```

```
data = data[['ID', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9', 'X10', 'Y']]
```

```
data.head()
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	Y
0	P1000342	A4	-0.4945	24.935	10.6905	Group2	5.0	Age4	12.075	102	Male	8680.83289773887
1	P1000864	A4	1.7582	8.017	9.9748	Group2	None	Age5	9.96618	88	Female	815078.562414822
2	P1005777	A3	-0.9239	11.739	7.2338	Group1	6.0	Age6	9.5947	85	Female	106371.230243878
3	P1006134	A1	-0.6068	-7.772	13.1339	Group2	5.0	Age3	9.29497	114	Male	10852.2018059725
4	P1006412	A4	0.7677	2.398	10.2349	Group2	5.0	Age5	12.10124	91	Female	34034.552569579

```
data2 = data.copy()
```

```
test2 = test.copy()
```

- DATA1
  - 범주형 변수 결측치 제거
  - MinMaxScaler 이용

- DATA2
  - 범주형 변수 최빈값 대체
  - StandartdScaler 이용

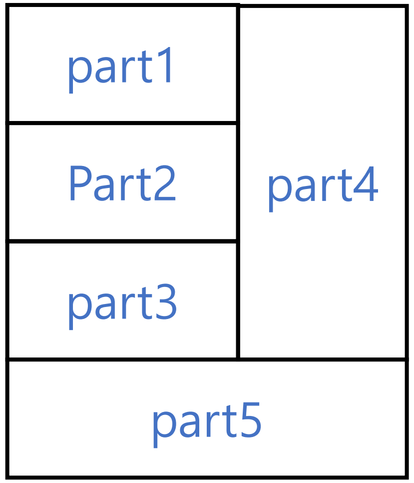
3. DATA TYPE

```
part1.printSchema()
```

```
root
 |-- ID: string (nullable = true)
 |-- X1: string (nullable = true)
 |-- X2: double (nullable = true)
 |-- X3: double (nullable = true)
 |-- X4: double (nullable = true)
 |-- X5: string (nullable = true)
 |-- Y: double (nullable = true)
```

```
part4.printSchema()
```

```
root
 |-- ID: string (nullable = true)
 |-- X6: double (nullable = true)
 |-- X7: string (nullable = true)
 |-- X8: double (nullable = true)
 |-- X9: short (nullable = true)
 |-- X10: string (nullable = true)
```




```
lst = ['X2', 'X3', 'X4', 'X6', 'X8', 'Y']

for col in lst:
    data[col] = data[col].replace('NA', -999)
    data[col] = data[col].astype('double')
    data[col] = data[col].replace(-999, np.nan)

data['X9'] = data['X9'].astype('short')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1100000 entries, 0 to 1099999
Data columns (total 12 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   ID      1100000 non-null  object
 1   X1      1095186 non-null  object
 2   X2      1100000 non-null  float64
 3   X3      1100000 non-null  float64
 4   X4      1059405 non-null  float64
 5   X5      1099062 non-null  object
 6   X6      1015249 non-null  float64
 7   X7      1100000 non-null  object
 8   X8      1100000 non-null  float64
 9   X9      1100000 non-null  int16
 10  X10     1078024 non-null  object
 11  Y       1100000 non-null  float64
dtypes: float64(6), int16(1), object(5)
memory usage: 94.4+ MB
```

4. DROP DUPLICATES

ID 기준 중복되는 행 제거

```
data.shape
```

```
(1100000, 12)
```

```
data = data.drop_duplicates(['ID'], keep='first', ignore_index=True)
```

```
df_id = data[['ID']]
```

```
data.shape
```

```
(1000000, 12)
```

5. TEST DATA

```
testX = spark.read.option('header', 'true').csv("hdfs://localhost:9000/finalDB/TestX.csv")
testY = spark.read.option('header', 'true').csv("hdfs://localhost:9000/finalDB/TestY.csv")
```

```
testX = testX.toPandas()
testY = testY.toPandas()
```

```
test = pd.merge(testX, testY, on=['ID'], how='right')
test
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	Y
0	P9920380	A2	0.2921	17.636	10.6735	Group2	6	Age2	10.81764	113	Female	13441.9471005922
1	P9619726	A2	1.0492	23.758	13.1427	Group2	5	Age2	11.0113	96	Male	37465.0559510013
2	P9851405	A4	0.6842	26.687	6.7143	Group2	6	Age3	8.65085	109	Female	345725.74044955
3	P9764914	A3	-0.6416	12.955	11.5501	Group2	6	Age4	10.52249	125	Male	23663.0498937355
4	P9615948	A2	1.1395	20.243	10.3386	Group2	5	Age4	8.39666	86	Male	1608375.72917404
...	...	...	...	...	...	...	...	...	...	...	...	...
48595	P9599067	A1	1.6197	-4.713	11.1224	Group2	5	Age3	7.56934	88	Female	887727.955459428
48596	P9851235	A3	0.1437	8.867	12.6313	Group3	5	Age1	9.84248	121	NA	9297.56648520998
48597	P9658431	A1	-0.9883	14.446	NA	Group1	5	Age4	9.43371	139	Male	31485.7594803602
48598	P9661211	A1	0.358	21.694	8.4582	Group3	5	Age2	10.90353	123	Male	21720.4420341918
48599	P9609076	A2	1.3599	-4.464	11.3768	Group1	6	Age3	7.71662	88	Female	548630.745246185

48600 rows × 12 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48600 entries, 0 to 48599
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      48600 non-null  object
 1   X1      48600 non-null  object
 2   X2      48600 non-null  float64
 3   X3      48600 non-null  float64
 4   X4      46742 non-null  float64
 5   X5      48600 non-null  object
 6   X6      44818 non-null  float64
 7   X7      48600 non-null  object
 8   X8      48600 non-null  float64
 9   X9      48600 non-null  int16
 10  X10     48600 non-null  object
 11  Y       48600 non-null  object
dtypes: float64(5), int16(1), object(6)
memory usage: 4.2+ MB
```

6. MISSING VALUES

```
data.isna().sum()
```

```
ID          0
X1       4814
X2          0
X3          0
X4      36868
X5        938
X6      76994
X7          0
X8          0
X9          0
X10     21976
Y           0
dtype: int64
```

```
test.isna().sum()
```

```
ID         0
X1         0
X2         0
X3         0
X4      1858
X5         0
X6      3782
X7         0
X8         0
X9         0
X10        0
Y          0
dtype: int64
```

6-1. CATEGORICAL

```
numeric = data.select_dtypes(include = np.number).columns.tolist()
categoric = data.select_dtypes(exclude = np.number).columns.tolist()
```

DATA1) dropna( )

```
for col in categoric:
    data[col] = data[col].replace('NA', np.nan)
    print(col, ": ", data[col].isna().sum())
```

```
ID :  0
X1 :  9620
X5 :  1899
X7 :  0
X10 :  27427
```

```
data.dropna(subset=categoric, inplace=True)
```

```
for col in categoric:
    test[col] = test[col].replace('NA', np.nan)
    print(col, ": ", test[col].isna().sum())
```

```
ID :  0
X1 :  477
X5 :  104
X7 :  0
X10 :  1303
```

```
test.dropna(subset=categoric, inplace=True)
```

6-1. CATEGORICAL

DATA2) fillna( )

```
for col in categoric:
    data2[col] = data2[col].replace('NA', np.nan)
    print(col, ": ", data[col].isna().sum())
```

```
ID :  0
X1 :  4814
X5 :  938
X7 :  0
X10 :  21976
```

```
data2['X1'].fillna(data2["X1"].mode()[0], inplace=True)
data2['X5'].fillna(data2["X5"].mode()[0], inplace=True)
data2['X10'].fillna(data2["X10"].mode()[0], inplace=True)
```

```
for col in categoric:
    test2[col] = test2[col].replace('NA', np.nan)
    print(col, ": ", test2[col].isna().sum())
```

```
ID :  0
X1 :  477
X5 :  104
X7 :  0
X10 :  1303
```

```
test2['X1'].fillna(test2["X1"].mode()[0], inplace=True)
test2['X5'].fillna(test2["X5"].mode()[0], inplace=True)
test2['X10'].fillna(test2["X10"].mode()[0], inplace=True)
```

6-2. NUMERIC

- 딱히 이상치는 없어 보이고,
- 행 사이의 연관이 없어 확률 대체로 결측치 처리
- + KNNImputer 무한로딩 오류

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
```

```
imputer = IterativeImputer(random_state = 0)
imputer.set_output(transform = 'pandas')
```

```
▾ IterativeImputer
IterativeImputer(random_state=0)
```

```
data[numeric] = imputer.fit_transform(data[numeric])
```

```
data2[numeric] = imputer.fit_transform(data[numeric])
```

```
test[numeric] = imputer.fit_transform(test[numeric])
```

```
test2[numeric] = imputer.fit_transform(test2[numeric])
```

```
data.isna().sum()
```

```
ID     0
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
Y      0
dtype: int64
```

```
test.isna().sum()
```

```
ID     0
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
Y      0
dtype: int64
```

```
data2.isna().sum()
```

```
ID     0
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
Y      0
dtype: int64
```

```
test2.isna().sum()
```

```
ID     0
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
Y      0
dtype: int64
```

7. X, Y SPLIT

```
trainX = data.drop(['ID', 'Y'], axis=1)
trainY = data.Y
```

```
trainX2 = data2.drop(['ID', 'Y'], axis=1)
trainY2 = data2.Y
```

```
print(trainX.shape)
print(trainY.shape)
```

```
(961376, 10)
(961376,)
```

```
testX = test.drop(['ID', 'Y'], axis=1)
testY = test.Y
```

```
testX2 = test2.drop(['ID', 'Y'], axis=1)
testY2 = test2.Y
```

```
print(testX.shape)
print(testY.shape)
```

```
(46730, 10)
(46730,)
```

8. SCALING

```
tnumeric = ['X2', 'X3', 'X4', 'X6', 'X8', 'X9']
tcategoric = ['X1', 'X5', 'X7', 'X10']
```

DATA1) MinMaxScaler( )

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(trainX[tnumeric])
```

```
trainX[tnumeric] = scaler.transform(trainX[tnumeric])

testX[tnumeric] = scaler.fit_transform(testX[tnumeric])
```

```
trainX.describe()
```

	X2	X3	X4	X6	X8	X9
count	961376.000	961376.000	961376.000	961376.000	961376.000	961376.000
mean	0.520	0.492	0.469	0.500	0.474	0.507
std	0.102	0.107	0.103	0.100	0.095	0.104
min	0.000	0.000	0.000	0.000	0.000	0.000
25%	0.451	0.420	0.402	0.400	0.410	0.438
50%	0.520	0.492	0.469	0.500	0.474	0.505
75%	0.589	0.564	0.536	0.600	0.539	0.578
max	1.000	1.000	1.000	1.000	1.000	1.000

```
testX.describe()
```

	X2	X3	X4	X6	X8	X9
count	46730.000	46730.000	46730.000	46730.000	46730.000	46730.000
mean	0.506	0.503	0.493	0.556	0.499	0.510
std	0.120	0.125	0.123	0.111	0.118	0.120
min	0.000	0.000	0.000	0.000	0.000	0.000
25%	0.425	0.418	0.412	0.444	0.420	0.431
50%	0.506	0.503	0.493	0.556	0.499	0.509
75%	0.587	0.588	0.574	0.667	0.578	0.593
max	1.000	1.000	1.000	1.000	1.000	1.000

8. SCALING

DATA2) StandardScaler( )

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
```

```
trainX2[tnumeric] = ss.fit_transform(trainX2[tnumeric])
testX2[tnumeric] = ss.fit_transform(testX2[tnumeric])
```

```
trainX2.describe()
```

	X2	X3	X4	X6	X8	X9
count	1000000.000	1000000.000	1000000.000	1000000.000	1000000.000	1000000.000
mean	0.000	0.000	0.000	-0.000	-0.000	0.000
std	1.000	1.000	1.000	1.000	1.000	1.000
min	-5.088	-4.619	-4.569	-4.999	-4.973	-4.866
25%	-0.674	-0.674	-0.657	-1.001	-0.676	-0.665
50%	0.000	-0.000	0.000	-0.002	0.000	-0.015
75%	0.675	0.675	0.658	0.998	0.675	0.686
max	5.126	4.763	5.174	4.996	5.513	4.737

```
testX2.describe()
```

	X2	X3	X4	X6	X8	X9
count	48600.000	48600.000	48600.000	48600.000	48600.000	48600.000
mean	-0.000	0.000	0.000	0.000	-0.000	0.000
std	1.000	1.000	1.000	1.000	1.000	1.000
min	-4.222	-4.018	-3.990	-5.009	-4.222	-4.258
25%	-0.676	-0.680	-0.658	-1.005	-0.670	-0.661
50%	-0.000	-0.001	-0.001	-0.004	0.004	-0.011
75%	0.674	0.675	0.656	0.997	0.670	0.688
max	4.118	3.995	4.095	3.999	4.243	4.085

9. DUMMIES

DATA1) X1, X5: 가변수처리, X7: label encoding, X10: 0과 1로

- 카테고리가 너무 많은 변수를 가변수 처리하면 데이터의 cardinality가 증가하여 성능이 떨어질 수 있음

```
trainX['X1'].value_counts()
```

```
X1
A4    240440
A1    240378
A3    240311
A2    240247
Name: count, dtype: int64
```

```
trainX['X1'] = trainX['X1'].astype('category')

x1 = pd.get_dummies(trainX['X1'])
x1 = x1*1
x1.
```

	A1	A2	A3	A4
0	0	0	0	1
1	0	0	0	1
2	0	0	1	0
3	1	0	0	0
4	0	0	0	1

```
trainX['X5'].value_counts()
```

```
X5
Group2    657257
Group1    152200
Group3    151919
Name: count, dtype: int64
```

```
trainX['X5'] = trainX['X5'].astype('category')

x5 = pd.get_dummies(trainX['X5'])
x5 = x5*1
x5
```

	Group1	Group2	Group3
0	0	1	0
1	0	1	0
2	1	0	0
3	0	1	0
4	0	1	0

```
trainX[["X10"]] = trainX[["X10"]].replace({"Male":0,"Female":1})
```

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
encoder.fit(trainX['X7'])
trainX['X7'] = encoder.transform(trainX['X7'])
trainX[['X7']]
```

	X2	X3	X4	X6	X7	X8	X9	X10	A1	A2	A3	A4	Group1	Group2	Group3
0	-0.495	1.494	0.351	-0.002	3	2.075	0.235	0	0	0	0	1	0	1	0
1	1.756	-0.198	-0.014	0.000	4	-0.034	-0.465	1	0	0	0	1	0	1	0
2	-0.924	0.174	-1.409	0.998	5	-0.405	-0.615	1	0	0	1	0	1	0	0
3	-0.607	-1.777	1.595	-0.002	2	-0.705	0.836	0	1	0	0	0	0	1	0
4	0.766	-0.760	0.119	-0.002	4	2.101	-0.315	1	0	0	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
999995	0.200	0.563	-0.840	1.997	1	0.362	-0.815	1	0	0	1	0	0	1	0
999996	-0.640	-0.611	-0.014	-2.000	5	-0.859	-1.065	0	0	0	1	0	0	1	0
999997	-1.558	-0.503	0.972	-1.001	2	-1.179	1.086	1	0	0	0	1	0	0	1
999998	-0.636	-1.640	1.147	-0.002	2	0.195	0.235	0	0	0	0	1	0	1	0
999999	-1.378	2.026	1.109	-0.002	2	0.181	0.686	0	0	1	0	0	0	1	0

1000000 rows × 15 columns

9. DUMMIES

DATA2) X1, X5: 가변수처리, X7: label encoding, X10: 0과 1로

- DATA1과 동일하게 처리

```
trainX2['X1'] = trainX2['X1'].astype('category')
x1 = pd.get_dummies(trainX2['X1'])
x1 = x1*1

trainX2['X5'] = trainX2['X5'].astype('category')
x5 = pd.get_dummies(trainX2['X5'])
x5 = x5*1

from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
encoder.fit(trainX2['X7'])
trainX2['X7'] = encoder.transform(trainX2['X7'])
```

```
trainX2 = trainX2.drop(['X1', 'X5'], axis=1)
```

```
trainX2 = pd.concat([trainX2, x1], axis=1)
```

```
trainX2 = pd.concat([trainX2, x5], axis=1)
```

	X2	X3	X4	X6	X7	X8	X9	X10	A1	A2	A3	A4	Group1	Group2	Group3
0	-0.495	1.494	0.351	-0.002	3	2.075	0.235	0	0	0	0	1	0	1	0
1	1.756	-0.198	-0.014	0.000	4	-0.034	-0.465	1	0	0	0	1	0	1	0
2	-0.924	0.174	-1.409	0.998	5	-0.405	-0.615	1	0	0	1	0	1	0	0
3	-0.607	-1.777	1.595	-0.002	2	-0.705	0.836	0	1	0	0	0	0	1	0
4	0.766	-0.760	0.119	-0.002	4	2.101	-0.315	1	0	0	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
999995	0.200	0.563	-0.840	1.997	1	0.362	-0.815	1	0	0	1	0	0	1	0
999996	-0.640	-0.611	-0.014	-2.000	5	-0.859	-1.065	0	0	0	1	0	0	1	0
999997	-1.558	-0.503	0.972	-1.001	2	-1.179	1.086	1	0	0	0	1	0	0	1
999998	-0.636	-1.640	1.147	-0.002	2	0.195	0.235	0	0	0	0	1	0	1	0
999999	-1.378	2.026	1.109	-0.002	2	0.181	0.686	0	0	1	0	0	0	1	0

1000000 rows × 15 columns

9. DUMMIES

DATA3) X1, X5 , X7: 가변수처리, X10: 0과 1로

```python
trainX['X7'] = trainX['X7'].astype('category')
```

```python
x7 = pd.get_dummies(trainX['X7'])
x7 = x7*1
x7
```

	0	1	2	3	4	5
0	0	0	0	1	0	0
1	0	0	0	0	1	0
2	0	0	0	0	0	1
3	0	0	1	0	0	0
4	0	0	0	0	1	0
...	...	...	...	...	...	...
999995	0	1	0	0	0	0
999996	0	0	0	0	0	1
999997	0	0	1	0	0	0
999998	0	0	1	0	0	0
999999	0	0	1	0	0	0

961376 rows × 6 columns

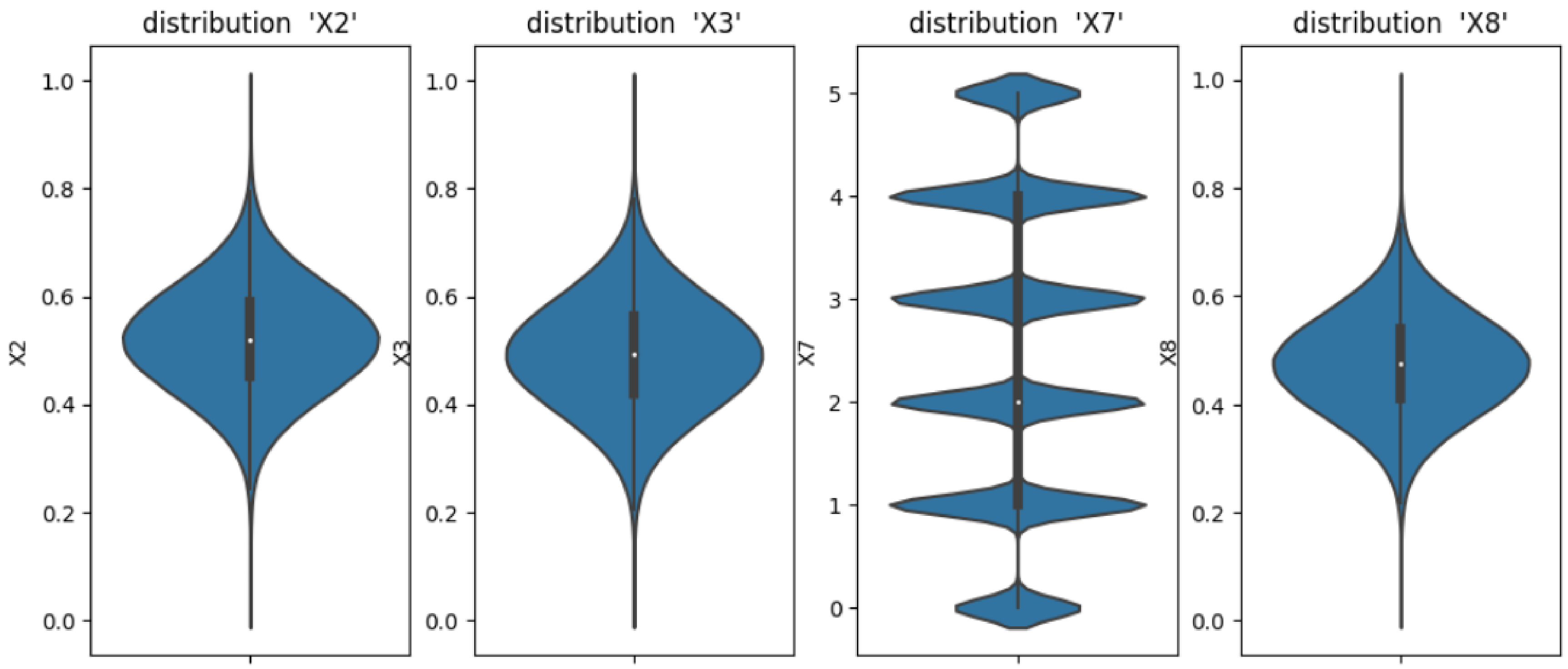
```python
trainX = trainX.drop('X7', axis=1)
trainX = pd.concat([trainX, x7], axis=1)
```

	X2	X3	X4	X6	X8	X9	X10	A1	A2	A3	A4	Group1	Group2	Group3	0	1	2	3	4	5
0	0.470	0.652	0.505	0.500	0.672	0.531	0	0	0	0	1	0	1	0	0	0	0	1	0	0
1	0.700	0.471	0.468	0.500	0.471	0.458	1	0	0	0	1	0	1	0	0	0	0	0	1	0
2	0.426	0.511	0.324	0.600	0.436	0.443	1	0	0	1	0	1	0	0	0	0	0	0	0	1
3	0.458	0.303	0.633	0.500	0.407	0.594	0	1	0	0	0	0	1	0	0	0	1	0	0	0
4	0.599	0.411	0.481	0.500	0.675	0.474	1	0	0	0	1	0	1	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
999995	0.541	0.552	0.383	0.700	0.509	0.422	1	0	0	1	0	0	1	0	0	1	0	0	0	0
999996	0.455	0.427	0.468	0.300	0.392	0.396	0	0	0	1	0	0	1	0	0	0	0	0	0	1
999997	0.361	0.439	0.569	0.400	0.362	0.620	1	0	0	0	1	0	0	1	0	0	1	0	0	0
999998	0.455	0.318	0.587	0.500	0.493	0.531	0	0	0	0	1	0	1	0	0	0	1	0	0	0
999999	0.379	0.708	0.583	0.500	0.492	0.578	0	0	1	0	0	0	1	0	0	0	1	0	0	0

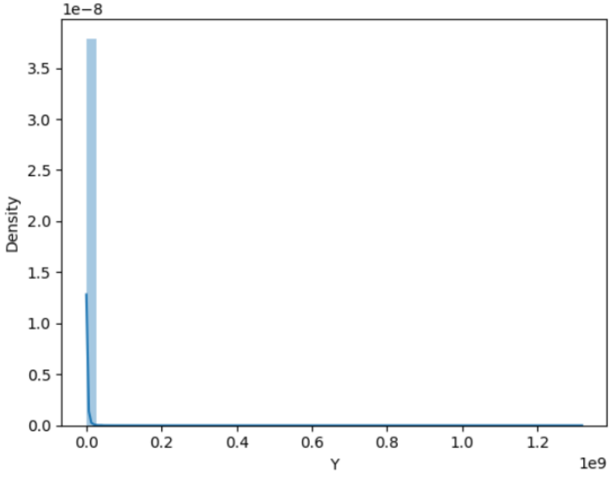
961376 rows × 20 columns

10. VISUALIZATION

```
figure, ax_list = plt.subplots(nrows=1, ncols=4)
figure.set_size_inches(12,5)

cols = ['X2', 'X3', 'X7', 'X8']

for i in range(4):
    col = cols[i]
    sns.violinplot(data=trainX, y=col, showfliers=True, ax=ax_list[i])
    ax_list[i].set_title(f"distribution  '{col}'")

# 분포가 고르기 때문에 로그 변환 X
```



10-1. Y-VISUALIZATION



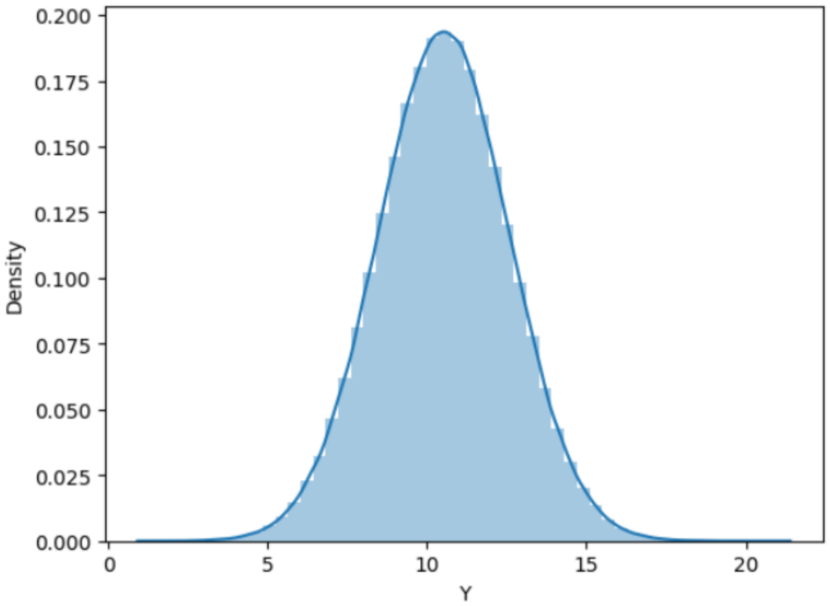

```
numeric = ['X2', 'X3', 'X4', 'X6', 'X8', 'X9', 'Y']

for col in numeric:
    print('{:15}'.format(col),
          'Skewness: {:05.2f}'.format(data[col].skew()) ,
          '   ' ,
          'Kurtosis: {:06.2f}'.format(data[col].kurt())
         )
```

```
X2              Skewness: 00.00     Kurtosis: -00.00
X3              Skewness: -0.00     Kurtosis: -00.00
X4              Skewness: -0.00     Kurtosis: -00.01
X6              Skewness: 00.00     Kurtosis: -00.00
X8              Skewness: 00.00     Kurtosis: -00.00
X9              Skewness: -0.00     Kurtosis: -00.00
Y               Skewness: 182.92     Kurtosis: 78437.10
```

```
trainY_log = np.log1p(trainY)
```

```
trainY_log
```

```
0          9.068988
1         13.611041
2         11.574700
3          9.292215
4         10.435161
            ...
999995     8.863625
999996    11.985615
999997     9.485059
999998     8.499563
999999     9.187542
Name: Y, Length: 961376, dtype: float64
```



Y(target)의 왜도와 첨도가 매우 큼

(한 쪽으로 몰려있음)

-> 로그 변환

11. RESIDUAL CHECK

```python
model = sm.OLS(list(trainY), sm.add_constant(trainX))
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.056
Model:                            OLS   Adj. R-squared:                  0.056
Method:                 Least Squares   F-statistic:                     4404.
Date:                Wed, 21 Jun 2023   Prob (F-statistic):               0.00
Time:                        09:17:03   Log-Likelihood:            -1.5512e+07
No. Observations:              961376   AIC:                         3.102e+07
Df Residuals:                  961362   BIC:                         3.102e+07
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.313e+16   2.74e+17     -0.048      0.962    -5.5e+17    5.23e+17
X2          3.835e+06   2.46e+04    156.103      0.000    3.79e+06    3.88e+06
X3          1.042e+06   2.36e+04     44.207      0.000    9.96e+05    1.09e+06
X4         -3.118e+04   2.45e+04     -1.274      0.203   -7.91e+04    1.68e+04
X6          3957.9485   2.51e+04      0.158      0.875   -4.52e+04    5.32e+04
X7          2.261e+05   1674.157    135.074      0.000    2.23e+05    2.29e+05
X8         -2.897e+06   2.63e+04   -109.988      0.000   -2.95e+06   -2.85e+06
X9          4.972e+05   2.41e+04     20.613      0.000     4.5e+05    5.44e+05
X10         3519.4242   5022.953      0.701      0.484   -6325.395    1.34e+04
A1         -1.276e+14   2.66e+15     -0.048      0.962   -5.34e+15    5.09e+15
A2         -1.276e+14   2.66e+15     -0.048      0.962   -5.34e+15    5.09e+15
A3         -1.276e+14   2.66e+15     -0.048      0.962   -5.34e+15    5.09e+15
A4         -1.276e+14   2.66e+15     -0.048      0.962   -5.34e+15    5.09e+15
Group1      1.326e+16   2.76e+17      0.048      0.962   -5.28e+17    5.55e+17
Group2      1.326e+16   2.76e+17      0.048      0.962   -5.28e+17    5.55e+17
Group3      1.326e+16   2.76e+17      0.048      0.962   -5.28e+17    5.55e+17
==============================================================================
Omnibus:                  4719140.970   Durbin-Watson:                     1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   325571321424171.062
Skew:                         201.613   Prob(JB):                           0.00
Kurtosis:                   90155.487   Cond. No.                       1.49e+15
==============================================================================
```

|t| >= 2 & p-value <= 0.05

DATA1, 모든 컬럼으로 확인

- X2, X3, X7, X8, X9 유의한 것으로 판단

11. RESIDUAL CHECK

```python
model = sm.OLS(list(trainY2), sm.add_constant(trainX2))
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.056
Model:                            OLS   Adj. R-squared:                  0.056
Method:                 Least Squares   F-statistic:                     4594.
Date:                Wed, 21 Jun 2023   Prob (F-statistic):               0.00
Time:                        09:17:35   Log-Likelihood:            -1.6134e+07
No. Observations:             1000000   AIC:                         3.227e+07
Df Residuals:                  999986   BIC:                         3.227e+07
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.082e+16   3.05e+17      0.101      0.919   -5.66e+17    6.28e+17
X2           3.93e+05   2464.926    159.438      0.000    3.88e+05    3.98e+05
X3          1.108e+05   2460.896     45.018      0.000    1.06e+05    1.16e+05
X4         -3394.8684   2490.684     -1.363      0.173   -8276.525    1486.788
X6         -1187.3799   2496.240     -0.476      0.634   -6079.927    3705.167
X7           2.26e+05   1640.595    137.784      0.000    2.23e+05    2.29e+05
X8         -2.758e+05   2463.541   -111.947      0.000   -2.81e+05   -2.71e+05
X9          5.207e+04   2463.901     21.133      0.000    4.72e+04    5.69e+04
X10         4318.4234   4922.701      0.877      0.380   -5329.905     1.4e+04
A1         -3.301e+16   3.26e+17     -0.101      0.919   -6.73e+17    6.07e+17
A2         -3.301e+16   3.26e+17     -0.101      0.919   -6.73e+17    6.07e+17
A3         -3.301e+16   3.26e+17     -0.101      0.919   -6.73e+17    6.07e+17
A4         -3.301e+16   3.26e+17     -0.101      0.919   -6.73e+17    6.07e+17
Group1      2.182e+15   2.16e+16      0.101      0.919   -4.01e+16    4.45e+16
Group2      2.182e+15   2.16e+16      0.101      0.919   -4.01e+16    4.45e+16
Group3      2.182e+15   2.16e+16      0.101      0.919   -4.01e+16    4.45e+16
==============================================================================
Omnibus:                  4875922.739   Durbin-Watson:                       1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   317546214955778.688
Skew:                         196.912   Prob(JB):                             0.00
Kurtosis:                   87301.076   Cond. No.                         1.96e+15
==============================================================================
```

$|t| >= 2$ & p-value $<= 0.05$

DATA2, 모든 컬럼으로 확인

- X2, X3, X7, X8, X9 유의한 것으로 판단

11. RESIDUAL CHECK

```
model = sm.OLS(trainY_log, sm.add_constant(trainX))
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.976
Model:                            OLS   Adj. R-squared:                  0.976
Method:                 Least Squares   F-statistic:                 2.264e+06
Date:                Thu, 22 Jun 2023   Prob (F-statistic):               0.00
Time:                        20:17:21   Log-Likelihood:            -2.7244e+05
No. Observations:              961376   AIC:                         5.449e+05
Df Residuals:                  961358   BIC:                         5.451e+05
Df Model:                          17
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       6.067e+10      9e+10      0.674      0.500   -1.16e+11    2.37e+11
X2             1.2669      0.000   3860.359      0.000       1.266       1.268
X3             0.3516      0.000   1070.292      0.000       0.351       0.352
X4         -5.678e-05      0.000     -0.173      0.863      -0.001       0.001
X6             0.0006      0.000      1.777      0.076   -6.01e-05       0.001
X8            -0.8910      0.000  -2717.717      0.000      -0.892      -0.890
X9             0.1695      0.000    517.260      0.000       0.169       0.170
X10            0.0011      0.001      1.650      0.099      -0.000       0.002
0          -1.845e+10   5.17e+10     -0.357      0.721    -1.2e+11    8.29e+10
1          -1.845e+10   5.17e+10     -0.357      0.721    -1.2e+11    8.29e+10
2          -1.845e+10   5.17e+10     -0.357      0.721    -1.2e+11    8.29e+10
3          -1.845e+10   5.17e+10     -0.357      0.721    -1.2e+11    8.29e+10
4          -1.845e+10   5.17e+10     -0.357      0.721    -1.2e+11    8.29e+10
5          -1.845e+10   5.17e+10     -0.357      0.721    -1.2e+11    8.29e+10
A1          6.659e+09   7.12e+10      0.093      0.926   -1.33e+11    1.46e+11
A2          6.659e+09   7.12e+10      0.093      0.926   -1.33e+11    1.46e+11
A3          6.659e+09   7.12e+10      0.093      0.926   -1.33e+11    1.46e+11
A4          6.659e+09   7.12e+10      0.093      0.926   -1.33e+11    1.46e+11
Group1     -4.889e+10   6.84e+10     -0.714      0.475   -1.83e+11    8.53e+10
Group2     -4.889e+10   6.84e+10     -0.714      0.475   -1.83e+11    8.53e+10
Group3     -4.889e+10   6.84e+10     -0.714      0.475   -1.83e+11    8.53e+10
==============================================================================
Omnibus:                   113122.575   Durbin-Watson:                   2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1213426.547
Skew:                          -0.004   Prob(JB):                         0.00
Kurtosis:                       8.504   Cond. No.                     7.12e+14
==============================================================================
```

|t| >= 2 & p-value <= 0.05

DATA3, 모든 컬럼으로 확인

- X2, X3, X8, X9 유의한 것으로 판단

12. RESIDUAL CHECK ITERATION

DATA1

```
for i in range(10):
    print(f'Iteration {i+1}')

    sample = np.random.choice(train.index, size=1000, replace=False)
    sub = train.loc[sample]

    x = sub.drop('Y', axis=1)
    y = sub.Y

    model = sm.OLS(y, sm.add_constant(x))
    results = model.fit()

    print('summary')
    print(results.summary())
    print(' ')
    print('------------------')
```

```
Iteration 1
summary
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.189
Model:                            OLS   Adj. R-squared:                  0.178
Method:                 Least Squares   F-statistic:                     17.63
Date:                Wed, 21 Jun 2023   Prob (F-statistic):           4.58e-37
Time:                        09:18:13   Log-Likelihood:                -15474.
No. Observations:                1000   AIC:                         3.098e+04
Df Residuals:                     986   BIC:                         3.104e+04
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -7.884e+05   3.16e+05     -2.495      0.013   -1.41e+06   -1.68e+05
X2          3.963e+06      4e+05      9.904      0.000    3.18e+06    4.75e+06
X3          1.023e+06   3.65e+05      2.801      0.005    3.06e+05    1.74e+06
```

- t-value, p-value 고려한 변수 선택
  - X2, X3, X7, X8, A1-A4, G1-G3

```
col = ['X2', 'X3', 'X7', 'X8', 'A1', 'A2', 'A3', 'A4', 'Group1', 'Group2', 'Group3']
```

col = ['X2', 'X3', 'X7', 'X8',

'A1', 'A2', 'A3', 'A4',

'Group1', 'Group2', 'Group3']

21. RESIDUAL CHECK ITERATION

DATA2

```python
for i in range(10):
    print(f'Iteration {i+1}')

    sample = np.random.choice(train2.index, size=1000, replace=False)
    sub = train2.loc[sample]

    x = sub.drop('Y', axis=1)
    y = sub.Y

    model = sm.OLS(y, sm.add_constant(x))
    results = model.fit()

    print('summary')
    print(results.summary())
    print(' ')
    print('--------------------')
```

col2 = ['X2', 'X3', 'X7', 'X8']

```
--------------------
Iteration 10

summary
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.122
Model:                            OLS   Adj. R-squared:                  0.111
Method:                 Least Squares   F-statistic:                     10.57
Date:                Wed, 21 Jun 2023   Prob (F-statistic):           2.81e-21
Time:                        09:18:15   Log-Likelihood:                -15795.
No. Observations:                1000   AIC:                         3.162e+04
Df Residuals:                     986   BIC:                         3.169e+04
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.502e+05   8.19e+04     -1.834      0.067    -3.11e+05    1.05e+04
X2          3.919e+05   5.51e+04      7.112      0.000     2.84e+05       5e+05
```

- t-value, p-value 고려한 변수 선택
  - X2, X3, X7, X8

```python
col2 = ['X2', 'X3', 'X7', 'X8']
```

21. RESIDUAL CHECK ITERATION

DATA3

```python
for i in range(10):
    print(f'Iteration {i+1}')

    sample = np.random.choice(train.index, size=1000, replace=False)
    sub = train.loc[sample]

    x = sub.drop('Y', axis=1)
    y = sub.Y

    model = sm.OLS(y, sm.add_constant(x))
    results = model.fit()

    print('summary')
    print(results.summary())
    print(' ')
    print('-------------------')
```

```
[2] The smallest eigenvalue is 1.14e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

-------------------
Iteration 3
summary
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.127
Model:                            OLS   Adj. R-squared:                  0.112
Method:                 Least Squares   F-statistic:                     8.424
Date:                Wed, 21 Jun 2023   Prob (F-statistic):           2.33e-20
Time:                        11:26:14   Log-Likelihood:                -15902.
No. Observations:                1000   AIC:                         3.184e+04
Df Residuals:                     982   BIC:                         3.193e+04
Df Model:                          17
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

- DATA3) t-value, p-value 고려한 변수 선택
  - X2, X3, X8, A1-A4, G1-G3, 0-5

```python
col3 = ['X2', 'X3', 'X8', 'A1', 'A2', 'A3', 'A4', 'Group1', 'Group2', 'Group3', '0', '1', '2', '3', '4', '5']
```

col3 = ['X2', 'X3', 'X8', 'X9'
'A1', 'A2', 'A3', 'A4',
'Group1', 'Group2', 'Group3', '0', '1', '2', '3', '4', '5']

22. VIF

DATA2

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()

list = []

for i in range(trainX2.shape[1]):
    v = variance_inflation_factor(trainX2.values, i)
    list.append(v)

vif["VIF Factor"] = list
vif["features"] = trainX2.columns
vif
```

	VIF Factor	features
0	1.000	X2
1	1.000	X3
2	1.000	X4
3	1.000	X6
4	1.000	X7
5	1.000	X8
6	1.000	X9
7	1.000	X10
8	inf	A1
9	inf	A2
10	inf	A3
11	inf	A4
12	inf	Group1
13	inf	Group2
14	inf	Group3

'A1'-'A4'

'Group1'-'Group3'

다중공선성 높음 -> 제외

DATA3

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()

list = []

for i in range(trainX.shape[1]):
    v = variance_inflation_factor(trainX.values, i)
    list.append(v)

vif["VIF Factor"] = list
vif["features"] = trainX.columns
vif
```

	VIF Factor	features
0	1.000	X2
1	1.000	X3
2	1.000	X4
3	1.000	X6
4	1.000	X8
5	1.000	X9
6	1.000	X10
7	1283719225.125	0
8	56092534.597	1
9	2491589103.181	2
10	1856198.008	3
11	2178238520.606	4
12	29324106.362	5
13	18474961.035	A1
14	12658561246.210	A2
15	26144189.282	A3
16	28149867.802	A4
17	144684666924.872	Group1
18	8366738.026	Group2
19	850230677.172	Group3

'A1'-'A4'

'Group1'-'Group3'

'0'-'5'

다중공선성 높음

but, 유의한 변수로 판단했기 때문에 우선 유지

23. OLS

DATA1

```python
model = sm.OLS(trainY, sm.add_constant(trainX[col]))
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.056
Model:                            OLS   Adj. R-squared:                  0.056
Method:                 Least Squares   F-statistic:                     6312.
Date:                Wed, 21 Jun 2023   Prob (F-statistic):               0.00
Time:                        11:01:00   Log-Likelihood:            -1.5513e+07
No. Observations:              961376   AIC:                         3.103e+07
Df Residuals:                  961366   BIC:                         3.103e+07
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -2.029e+16   3.23e+17     -0.063      0.950   -6.54e+17    6.13e+17
X2          3.835e+06   2.46e+04    156.130      0.000    3.79e+06    3.88e+06
X3          1.041e+06   2.36e+04     44.161      0.000    9.95e+05    1.09e+06
X7          2.262e+05   1674.609    135.062      0.000    2.23e+05    2.29e+05
X8         -2.897e+06   2.63e+04   -109.976      0.000   -2.95e+06   -2.85e+06
A1          1.097e+15   1.75e+16      0.063      0.950   -3.32e+16    3.53e+16
A2          1.097e+15   1.75e+16      0.063      0.950   -3.32e+16    3.53e+16
A3          1.097e+15   1.75e+16      0.063      0.950   -3.32e+16    3.53e+16
A4          1.097e+15   1.75e+16      0.063      0.950   -3.32e+16    3.53e+16
Group1      1.919e+16   3.06e+17      0.063      0.950    -5.8e+17    6.18e+17
Group2      1.919e+16   3.06e+17      0.063      0.950    -5.8e+17    6.18e+17
Group3      1.919e+16   3.06e+17      0.063      0.950    -5.8e+17    6.18e+17
==============================================================================
Omnibus:                  4718284.201   Durbin-Watson:                     1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   324977040899480.875
Skew:                         201.485   Prob(JB):                           0.00
Kurtosis:                   90073.169   Cond. No.                       2.17e+15
==============================================================================
```

23. OLS

DATA2

```
model = sm.OLS(trainY2, sm.add_constant(trainX2[col2]))
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.056
Model:                            OLS   Adj. R-squared:                  0.056
Method:                 Least Squares   F-statistic:                 1.481e+04
Date:                Wed, 21 Jun 2023   Prob (F-statistic):               0.00
Time:                        11:46:23   Log-Likelihood:            -1.6135e+07
No. Observations:             1000000   AIC:                         3.227e+07
Df Residuals:                  999995   BIC:                         3.227e+07
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -2.559e+05   4781.364    -53.519      0.000   -2.65e+05   -2.47e+05
X2          3.931e+05   2460.261    159.780      0.000    3.88e+05    3.98e+05
X3          1.107e+05   2460.266     44.998      0.000    1.06e+05    1.16e+05
X7          2.261e+05   1640.058    137.843      0.000    2.23e+05    2.29e+05
X8         -2.758e+05   2460.264   -112.118      0.000   -2.81e+05   -2.71e+05
==============================================================================
Omnibus:                  4875031.665   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):   316953428054667.562
Skew:                         196.786   Prob(JB):                         0.00
Kurtosis:                   87219.554   Cond. No.                         6.17
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

23. OLS

DATA3

-

```python
model = sm.OLS(trainY_log, sm.add_constant(trainX[col3]))
result_log = model.fit()
# results = np.expm1(result_log)
print(result_log.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.976
Model:                            OLS   Adj. R-squared:                  0.976
Method:                 Least Squares   F-statistic:                 2.749e+06
Date:                Thu, 22 Jun 2023   Prob (F-statistic):               0.00
Time:                        20:26:08   Log-Likelihood:            -2.7242e+05
No. Observations:              961376   AIC:                         5.449e+05
Df Residuals:                  961361   BIC:                         5.451e+05
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.111e+11   1.16e+11     -0.955      0.339   -3.39e+11    1.17e+11
X2             1.2669      0.000   3865.422      0.000       1.266       1.268
X3             0.3516      0.000   1073.031      0.000       0.351       0.352
X8            -0.8910      0.000  -2717.196      0.000      -0.892      -0.890
X9             0.1695      0.000    516.765      0.000       0.169       0.170
A1         -3.726e+10   3.65e+10     -1.020      0.308   -1.09e+11    3.44e+10
A2         -3.726e+10   3.65e+10     -1.020      0.308   -1.09e+11    3.44e+10
A3         -3.726e+10   3.65e+10     -1.020      0.308   -1.09e+11    3.44e+10
A4         -3.726e+10   3.65e+10     -1.020      0.308   -1.09e+11    3.44e+10
Group1      9.509e+10    7.7e+10      1.235      0.217   -5.58e+10    2.46e+11
Group2      9.509e+10    7.7e+10      1.235      0.217   -5.58e+10    2.46e+11
Group3      9.509e+10    7.7e+10      1.235      0.217   -5.58e+10    2.46e+11
0           5.329e+10   4.25e+10      1.255      0.210   -2.99e+10    1.37e+11
1           5.329e+10   4.25e+10      1.255      0.210   -2.99e+10    1.37e+11
2           5.329e+10   4.25e+10      1.255      0.210   -2.99e+10    1.37e+11
3           5.329e+10   4.25e+10      1.255      0.210   -2.99e+10    1.37e+11
4           5.329e+10   4.25e+10      1.255      0.210   -2.99e+10    1.37e+11
5           5.329e+10   4.25e+10      1.255      0.210   -2.99e+10    1.37e+11
==============================================================================
Omnibus:                   113174.719   Durbin-Watson:                   2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1214913.317
Skew:                           0.005   Prob(JB):                         0.00
Kurtosis:                       8.507   Cond. No.                     1.13e+15
==============================================================================
```

24. LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression(n_jobs=-1)
```

DATA1

```
result = lr.fit(trainX[col], trainY)
print(result.coef_)
```

```
[ 3.83526380e+06  1.04089112e+06  2.26176259e+05 -2.89701327e+06
 -5.37707545e+16 -5.37707545e+16 -5.37707545e+16 -5.37707545e+16
  7.70995970e+16  7.70995970e+16  7.70995970e+16]
```

```
pred = lr.predict(testX[col])
print(pred[0:10])
```

```
[-189500.  171532. 1081572. -129164. 1512708.  -62180. -503132.   47740.
 -523492.    8052.]
```

```
accuracy = lr.score(testX[col], testY)
print(accuracy)
```

```
0.08285997452508131
```

```
print("train: ", lr.score(trainX[col], trainY))
print("test: ", lr.score(testX[col], testY))
```

```
train:  0.055791301958747375
test:  0.08285997452508131
```

24. LINEAR REGRESSION

DATA2

```
result = lr.fit(trainX2[col2], trainY2)
print(result.coef_)
```

```
[ 393101.33152143  110708.12763475  226069.66025595 -275839.18186622]
```

```
pred = lr.predict(testX2[col2])
print(pred[0:10])
```

```
[ -57459.15345344  253995.11103301 1019039.79533117   57796.13779811
 1422515.13270799  126593.87491142 -240788.90307793  157728.35355784
 -350220.57382104  150110.18596626]
```

```
accuracy = lr.score(testX2[col2], testY2)
print(accuracy)
```

```
0.08868003008670722
```

```
print("train: ", lr.score(trainX2[col2], trainY2))
print("test: ", lr.score(testX2[col2], testY2))
```

```
train:  0.05592044133893337
test:  0.08868003008670722
```

24. LINEAR REGRESSION

DATA3

- 범주형 자료 결측치 제거
- 수치자료 IterativeImputer로 대체
- StandardScaler
- X1, X5, X7은 가변수 처리,
- X10은 male:0, female:1
- Y 로그 변환
- col: X4, X6, X10 제외한 변수

```
result = lr.fit(trainX[col3], trainY_log)
print(result.coef_)
```

```
[ 1.26689112e+00  3.51596195e-01 -8.90983340e-01  1.69463948e-01
  5.30826377e+10  5.30826377e+10  5.30826377e+10  5.30826377e+10
 -1.89210318e+11 -1.89210318e+11 -1.89210318e+11  6.02355854e+10
  6.02355854e+10  6.02355854e+10  6.02355854e+10  6.02355854e+10
  6.02355854e+10]
```

```
pred3_log = lr.predict(testX[col3])
pred3 = np.expm1(pred3_log)
print(pred3[0:10])
```

```
[  12717.79604216   29944.79416213  416064.77212368   20206.07039693
 1181540.29019616   13647.60806653    5568.33266852   22570.52827246
    3747.32720737   22773.26001448]
```

```
accuracy = lr.score(testX[col3], testY_log)
print(accuracy)
```

```
0.9761505875908276
```

```
print("train: ", lr.score(trainX[col3], trainY_log))
print("test: ", lr.score(testX[col3], testY_log))
```

```
train:  0.9756314428536542
test:  0.9761505875908276
```

25. MSE

DATA1

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(testY, pred)
print(f'MSE: {mse:.3f}')
```

MSE: 3986494296931.519

DATA2

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(testY2, pred2)
print(f'MSE: {mse:.3f}')
```

MSE: 3895142425880.727

DATA3

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(testY, pred3)
print(f'MSE: {mse:.3f}')
```

MSE: 1542328389766.551