임대주택 설계시 적정 주차 수요 예측

사조 - 김현, 백지은, 양희진, 정태현

목차

Ⅰ. 분석 배경 및 목적

문제점

과거 법정 주차대수가 세대당 0.5대 수준에 불과해 심각한 주차공간 부족으로 몸살을 앓고 있는 아파트들이 많다.

이에 요즘 아파트들은 오히려 법정 주차대수보다 대폭 상향해 세대당 1.5대 내외의 주차공간을 확보하여 주차면수가 남는 문제가 발생하고 있다.

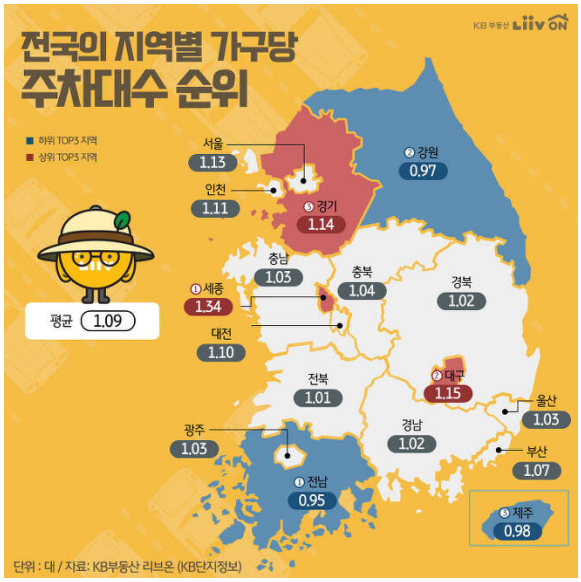
=> 과거 구축 단지에서는 과소평가 문제, 최근 신축 단지에서는 편의성 향상시켰으나 과대평가 하여 자원 과다 소비 우려. 이를 데이터를 이용하여 적정 수준의 주차 대수를 예측하고자 한다.

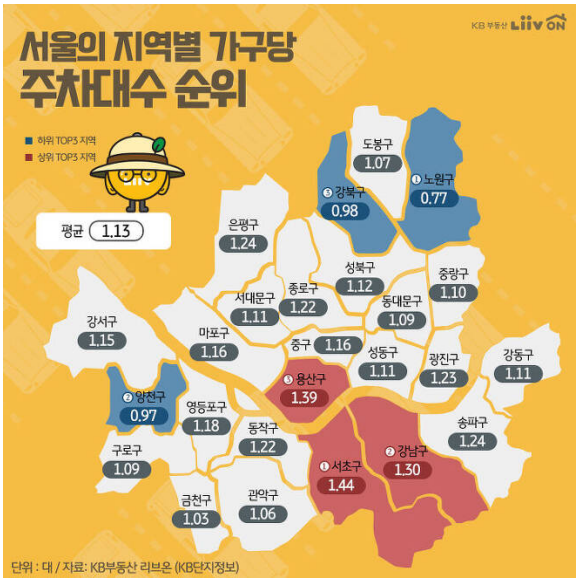
참고자료

아파트 주차대수 기준 22년만에 변경된다

아파트 등 공동주택의 법정 주차대수를 22년 만에 개선하는 방안이 추진된다. 국토교통부는 차량이 늘어나는 현실을 반영해 '주택 건설기준 등에 관한 규정'에 제시된 주차장 설치 기준을 손질하는 방안을 검토 중이라고 3일 밝혔다. 국토부 관

https://www.hani.co.kr/arti/economy/property/826126.html





https://content.v.kakao.com/v/5ebe49663faf2d0a61336e7d https://content.v.kakao.com/v/5ebe49663faf2d0a6133

분석 배경

자동차는 생애의 95%를 도로가 아닌 주차장에서 머무르며 운행되는 시간보다 주차장에 머물러 있는 시간이 많아 대부분의 주차면은 차량이 이미 점유하고 있는 상태 (Clive Thompson, 2016)

현재의 주차 대수 산출은 '주차원단위'와 '건축연면적'을 기초로 산출되고있어서 오차 문제, 시차 문제 등으로 과대/과소 산정 가능성이 높다. 이를 해결할 방안으로 보다 정확한 주차수요 예측 모델을 개발하고자 함

분석 목적

오차/ 시차문제로 인한 과대/과소산정을 최소화하고 보다 정확한 주차수요를 예측하는 모델 개발

Ⅱ. 분석 데이터

- 주택데이터 및 등록대수 데이터 (한국토지주택공사에서 제공)
- 지역별 성별/연령 비율 데이터 (한국토지주택공사에서 제공)

- 2019 시도별/연령별 자동차 등록대수 데이터 (통계청)
- 2019 지역별 세대당 세대원 수 데이터 (통계청)
- 2019 1인당 자동차 등록대수 데이터 (통계청)
- 2019 소득수준 데이터 (통계청)
- 2019 주택 연면적 데이터 (공공데이터포털)
- 2019 지역별 아파트 평균 분양가 데이터 (통계청)
- 2019 자전거도로 현황 (통계청)
- 2019 운전면허소지자현황 (통계청)

III. EDA, 데이터 전처리

1) 데이터 탐색

주택데이터 및 등록대수 데이터

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2952 entries, 0 to 2951
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   단지코드                          2952 non-null   object
 1   총세대수                          2952 non-null   int64
 2   임대건물구분                        2952 non-null   object
 3   지역                            2952 non-null   object
 4   공급유형                          2952 non-null   object
 5   전용면적                          2952 non-null   float64
 6   전용면적별세대수                      2952 non-null   int64
 7   공가수                           2952 non-null   float64
 8   자격유형                          2952 non-null   object
 9   임대보증금                         2383 non-null   object
 10  임대료                           2383 non-null   object
 11  도보 10분거리 내 지하철역 수(환승노선 수 반영)  2741 non-null   float64
 12  도보 10분거리 내 버스정류장 수           2948 non-null   float64
 13  단지내주차면수                       2952 non-null   float64
 14  등록차량수                         2952 non-null   float64
dtypes: float64(6), int64(2), object(7)
memory usage: 346.1+ KB
```

단지코드

총세대수

임대건물구분 - 아파트 / 상가

지역 - 16개 시도

공급유형 - 공공분양, 공공임대, 등 10가지 분류

전용면적

전용면적별세대수

공가수 - 빈 세대 (총세대 중 빈 세대고, 해당 전용면적에서 빈 세대가 아님)

자격유형 - 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O'

임대보증금 - NaN / '-' 로 인해 object 상태

임대료 - NaN / '-' 로 인해 object 상태

도보 10분 거리 내 지하철역 수(환승노선 수 반영)

도보 10분거리 내 버스정류장 수

단지내주차면수

등록차량수 - **Target 변수**

	총세대수	전용면적	전용면적별세대수	공가수	도보 10분거리 내 지하철역 수(환승노선 수 반영)	도보 10분거리 내 버스정류장 수	단지내주차면수	등록차량수
count	2952.000000	2952.000000	2952.000000	2952.000000	2741.000000	2948.000000	2952.000000	2952.000000
mean	886.661247	44.757215	102.747967	12.921070	0.176578	3.695726	601.668360	559.768293
std	513.540168	31.874280	132.640159	10.778831	0.427408	2.644665	396.407072	433.375027
min	26.000000	12.620000	1.000000	0.000000	0.000000	0.000000	13.000000	13.000000
25%	513.500000	32.100000	14.000000	4.000000	0.000000	2.000000	279.250000	220.000000
50%	779.000000	39.930000	60.000000	11.000000	0.000000	3.000000	517.000000	487.000000
75%	1106.000000	51.562500	144.000000	20.000000	0.000000	4.000000	823.000000	770.000000
max	2568.000000	583.400000	1865.000000	55.000000	3.000000	20.000000	1798.000000	2550.000000

- 다만, 주최측에서 데이터 오류로 제외하길 권유한 단지들이 포함되어있음.
- 해당 데이터 제외하여 진행하였음.

2) 결측치 처리

단지코드

- 주최측 오류 데이터 제공

  'C2335', 'C1327', 'C1095', 'C2051', 'C1218', 'C1894', 'C2483', 'C1502', 'C1988', 'C2085', 'C1397', 'C2431', 'C1649', 'C1036', 'C2675'

  ⇒ 삭제

지하철역 수

- 충남, 대전에서 발생

  ⇒ 충남의 경우 지역 평균값 0으로 대체,

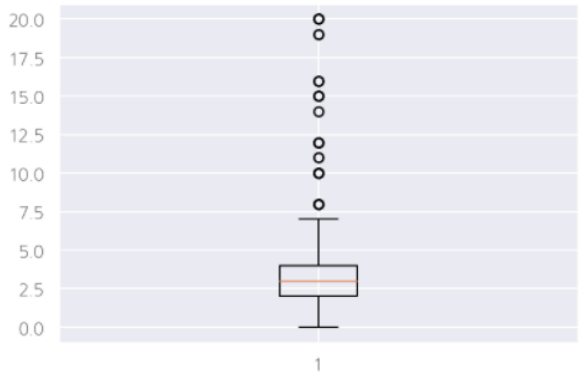
  대전의 경우 지역 평균이 0.7정도로 나타나 1로 대체.

임대보증금 / 임대료

- 임대료/임대보증금의 경우 null값이 많아 활용이 어려워 따로 결측치를 제거하고 활용하지 않고, 대신 지역별 평균 아파트 분양가격을 활용

3) 이상치 처리

버스정류장



- IQR 활용, 이상치를 극단치 경계로 수정

4) 그룹화

공급유형; 임대주택 유형

국민임대 (30년)

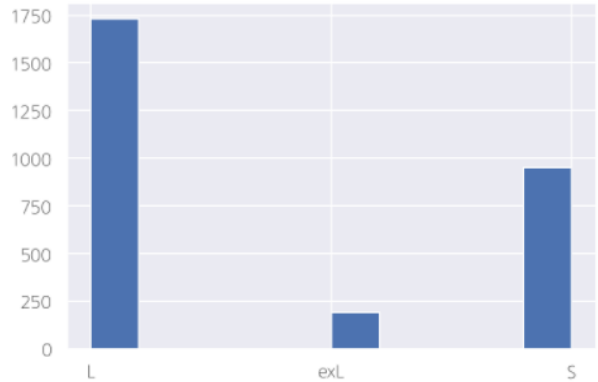
공공임대 (분납, 5년, 10년, 50년)

영구임대 (50년) ⇒ 단기/장기/초장기 기간별로 그룹화

장기전세 (20년)

공공분양

행복주택



자격유형; 임차를 할 수 있는 자격 요건

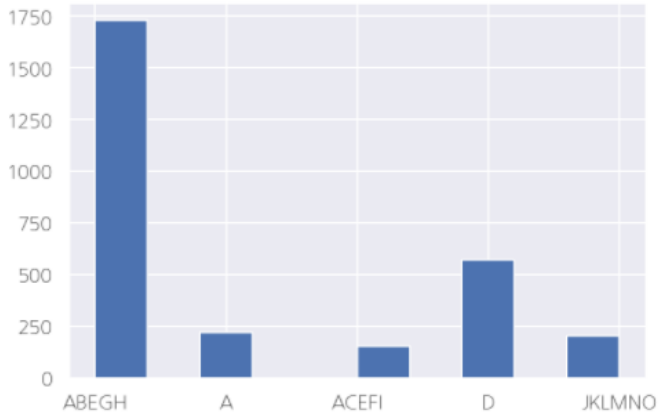
A : 공공임대(50년), 공공임대(10년), 공공임대(분납), 장기전세, 공공임대(5년)

D : 임대상가, 공공분양

A, C, E, F, I : 영구임대

J, K, L, M, N, O : 행복주택

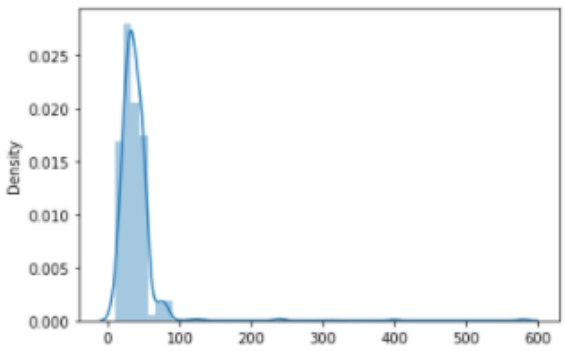
A, B, E, G, H : 국민임대



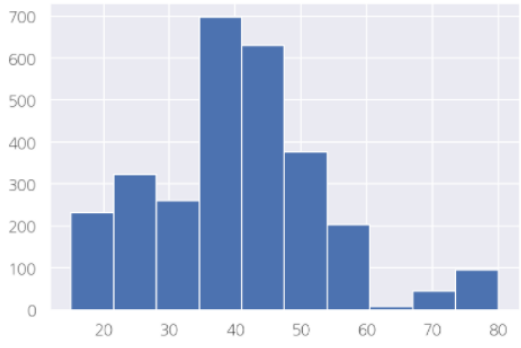
전용면적

mean	44.421394
std	32.072217
min	12.620000
25%	32.100000
50%	39.840000
75%	51.050000
max	583.400000

⇒ 최소 15, 최대 80, 5의 단위로 처리



전처리 전



전처리 후

5) 데이터 전처리

❗ 세대수/인구수 비율로 변환시켜준 이유 → 독립변수들간 관계성 때문에 과최적화 우려

임대건물구분

- 아파트/상가 세대수
- 총세대수 & 아파트/상가 비율

  총세대수 = 아파트세대수 + 상가세대수

  아파트비율 = 아파트세대수 / 총세대수

  상가비율 = 상가세대수 / 총세대수

기본정보

- 단지코드, 지역, 공가수, 도보 10분거리 내 지하철 역 수(환승노선 수 반영), 도보 10분거리 내 버스정류장 수, 단지내주차면수, 등록차량수

전용면적

- 전용면적별세대수 (5단위로 최소 15, 최대 80으로 가공한 데이터)

  전용면적비율 = 전용면적별 세대수 / 총세대수

공급유형

- 기간상 20년 초과의 초장기, 10년이하의 단기, 그리고 그 외의 장기 - 세가지 구분.

자격유형

- 각 단지의 입주 자격유형을 그룹핑, 'ABEGH', 'A', 'ACEFI', 'D', 'JKLMNO' 의 5가지로 분류.

나이/성별 인구수

- 나이/성별 인구수 → 결과에 영향 x ⇒ 그대로 붙여도 무관

1인당 자동차등록수

- 지역별 자동차 등록대수 / 주민등록인구 = 지역별 1인당 자동차등록대수

주택연면적

- 지역별 주택연면적

평당가격

- 2019 평균 분양가 데이터 → 결측치 많은 임대료/임대보증금 데이터 대체
- (2019 평균 분양가지수 데이터 → 지역별로 구분되어 활용 가능)

총면적 구하기 **

- 전용면적별세대수와 전용면적의 곱으로 확인

규모별 가격

- 규모별 가격 = 총면적 * 평균분양가격

자전거도로

- 자전거도로 데이터

  자전거도로비율 = 자전거도로 / 전국도로

운전면허소지자 현황

- 지역별 운전면허소지자수 비율

IV. 가설 검정

> ❗ **주택의 유형(지역, 전용면적, 공급유형, 자격유형 등)으로 해당 주택의 주차 수요 예측이 가능하다.**

1) ~~**대중교통이 불편할수록 주차 등록대수가 늘어날 것이다.**~~

- 전처리 후 OLS 모델을 통한 coefficient로 추정 결과, 지하철역은 - / 버스정류장은 + 로 추정
- 두 변수를 각기 두지 않고 합쳐진 칼럼으로 대체해서 재검증 예정

	coef	std err	t	P>|t|	[0.025	0.975]
총세대수	-0.1334	0.080	-1.669	0.096	-0.291	0.024
아파트비율	-24.2670	1309.277	-0.019	0.985	-2598.801	2550.267
상가비율	138.3023	1438.855	0.096	0.923	-2691.031	2967.636
공가수	-4.9897	0.952	-5.242	0.000	-6.861	-3.118
지하철역	-24.5044	23.414	-1.047	0.296	-70.545	21.536
버스정류장	7.2745	4.495	1.618	0.106	-1.565	16.114

2) ~~**주요도시(수도권 및 광역도시) 이외 지역은 교통 수단으로서 자가용 수요가 더 커서, 주차등록대수가 늘어날 것이다.**~~

- 예상과 달리 지역별로 추정 coefficient가 일관성이 없게 나옴.
- 지역 인코딩 제거하고 새로 모델링할 것.

강원도	21.6179
경기도	-137.9661
경상남도	-19.4117
경상북도	122.0461
광주광역시	-40.6628
대구광역시	-38.3621
대전광역시	-95.2634
부산광역시	-105.6717
세종특별자치시	359.6335
울산광역시	-188.9020
전라남도	-21.1743
전라북도	4.7557
제주특별자치도	-109.5566
충청남도	-32.2503
충청북도	63.8609

3) ***소득수준(지역별)은 주차 등록대수에 영향을 미칠 것이다.***

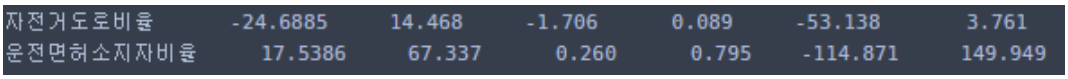
- 금일 저녁시간대에 업데이트, 테스트 후 금요일 제출 마감기한까지 완료 예정

4) ~~**지역별 연령인구별 자동차 등록대수가, 해당 지역 단지의 주차 등록대수에 영향을 미칠 것이다.**~~

- 1인당자동차 등록대수의 coefficient가 -21.7844로 영향은 끼치지만, 예상과 반대의 영향을 미치므로 해당 부분 재논의가 필요.

5) **지역별 자전거도로 현황, 운전면허소지현황이 주차 등록대수에 각기 음과 양의 방향으로 영향을 미칠 것이다.**

- 자전거도로비율은 coefficient의 부호, P-value의 크기로 볼 때 음의 방향으로 영향을 끼치는 것을 쉽게 확인할 수 있었고, 운전면허소지자 비율의 경우에는 P-value 수치가 높긴 하지만, 추정 coefficient 자체는 가설과 같음을 알 수 있음.



자전거도로비율	-24.6885	14.468	-1.706	0.089	-53.138	3.761
운전면허소지자비율	17.5386	67.337	0.260	0.795	-114.871	149.949

Ⅴ. 모델링

다중공선성 확인 및 OLS 통계량 확인

- 다중공선성이 아예 없도록 (모든 변수 10 이하로) 수정하려했으나 드랍되는 변수가 너무 많아지고, 해당 모델의 결과가 좋게 나타나지 않아, 관계를 확인할 수 있는 몇몇의 자료만 체크하고 수정하며 진행함.

ex)

```
model = sm.OLS(y, X)
result = model.fit()

print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                등록차량수   R-squared:                       0.864
Model:                            OLS   Adj. R-squared:                  0.848
Method:                 Least Squares   F-statistic:                     55.67
Date:                Tue, 27 Jul 2021   Prob (F-statistic):          4.92e-134
Time:                        09:33:30   Log-Likelihood:                -2621.4
No. Observations:                 411   AIC:                             5329.
Df Residuals:                     368   BIC:                             5502.
Df Model:                          42
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
총세대수          2.5539      1.555      1.643      0.101      -0.503       5.611
아파트세대수       -1.4888      1.533     -0.971      0.332      -4.503       1.525
상가세대수        -11.4741      6.896     -1.664      0.097     -25.036       2.087
공가수           -5.0329      1.022     -4.926      0.000      -7.042      -3.024
지하철          -28.6293     30.059     -0.952      0.342     -87.739      30.480
버스             4.9796      4.593      1.084      0.279      -4.052      14.011
단지내주차면수       0.2760      0.098      2.820      0.005       0.084       0.469
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(
    X.values, i) for i in range(X.shape[1])]
vif["features"] = X.columns
vif
```

```
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193: RuntimeWarning: divide by zero encountered in double_scalars
  vif = 1. / (1. - r_squared_i)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1715: RuntimeWarning: divide by zero encountered in double_scalars
  return 1 - self.ssr/self.centered_tss
```

	VIF Factor	features
0	inf	총세대수
1	inf	아파트세대수
2	26.925446	상가세대수
3	2.013108	공가수
4	1.810327	지하철
5	1.221231	버스
6	22.477092	단지내주차면수

```
s = corr_df.unstack()
s
```

```
총세대수  총세대수          1.00
          아파트비율        -0.13
          상가비율          0.13
          공가수            0.30
          지하철역          0.11
                           ...
규모별가격  연면적            0.29
          층별분양가격        0.14
          자전거도로          0.25
          층갯수            0.06
          규모별가격         1.00
Length: 784, dtype: float64
```

```
df = pd.DataFrame(s[s < 1].sort_values(ascending=False), columns=['corr'])
df.style.background_gradient(cmap='viridis')
```

		corr
연면적	자전거도로	0.970000
자전거도로	연면적	0.970000
규모별가격	층면적	0.860000
단지내주차면수	층면적	0.860000
총세대수	층면적	0.860000
	단지내주차면수	0.860000
층면적	규모별가격	0.860000
	총세대수	0.860000
등록차량수	단지내주차면수	0.850000
단지내주차면수	등록차량수	0.850000
등록차량수	층면적	0.800000
층면적	등록차량수	0.800000
규모별가격	단지내주차면수	0.760000
단지내주차면수	총세대수	0.760000
	규모별가격	0.760000
총세대수	단지내주차면수	0.760000
	규모별가격	0.740000
규모별가격	총세대수	0.740000
	등록차량수	0.730000
등록차량수	규모별가격	0.730000
전용면적_80_세대수	전용면적_70_세대수	0.590000
전용면적_70_세대수	전용면적_80_세대수	0.590000
총세대수	전용면적_25_세대수	0.580000
등록차량수	총세대수	0.580000
총세대수	등록차량수	0.580000
전용면적_25_세대수	총세대수	0.580000
전용면적_30_세대수	전용면적_25_세대수	0.510000
전용면적_25_세대수	전용면적_30_세대수	0.510000
상가비율	전용면적_25_세대수	0.490000
전용면적_25_세대수	상가비율	0.490000
단지내주차면수	전용면적_45_세대수	0.470000
전용면적_45_세대수	단지내주차면수	0.470000
	전용면적_35_세대수	0.440000
전용면적_35_세대수	전용면적_45_세대수	0.440000
전용면적_30_세대수	상가비율	0.430000
상가비율	전용면적_30_세대수	0.430000
전용면적_35_세대수	단지내주차면수	0.400000
공가수	단지내주차면수	0.400000
단지내주차면수	공가수	0.400000
	전용면적_35_세대수	0.400000

- 총세대수 - 아파트세대수와 상가세대수의 합으로 이루어지는데, 상가세대의 비율이 평균 0.015%로 대다수 아파트세대수로 구성되어있음

  ⇒ 따라서 총세대수와 아파트세대수의 다중공선성이 크게 나타나는 것으로 보였고, 이에 따라 총세대수를 남기고 아파트, 상가세대수는 비율만 남기는 식으로 활용

변수중요도 확인 및 칼럼 정리

]:

	Catboost	변수중요도	RandomForest	변수중요도.1	XGB	변수중요도.2	ExtraTrees	변수중요도.3	Gradient Boost	변수중요도.4
0	단지내주차면수	27.974759	단지내주차면수	0.563962	단지내주차면수	2.690195e-01	단지내주차면수	0.339318	단지내주차면수	0.552158
1	층면적	21.012296	층면적	0.222827	대전광역시	1.253529e-01	층면적	0.253400	층면적	0.200053
2	규모별가격	12.812154	규모별가격	0.027713	층면적	1.103567e-01	규모별가격	0.106637	규모별가격	0.049478
3	전용면적_25_비율	3.577833	층세대수	0.015322	A비율	6.710236e-02	A비율	0.023033	전용면적_25_비율	0.016041
4	층세대수	2.683418	전용면적_25_비율	0.015105	70대(남자)	6.105294e-02	층세대수	0.022811	10대미만(여자)	0.013885
5	ABEGH비율	2.585608	공가수	0.014930	아파트비율	5.304211e-02	초장기비율	0.022421	공가수	0.013667
6	80대(남자)	1.899604	전용면적_35_비율	0.012750	JKLMNO비율	4.899129e-02	ACEFI비율	0.017630	전용면적_35_비율	0.013562
7	전용면적_35_비율	1.804313	전용면적_45_비율	0.007967	50대(남자)	2.806111e-02	전용면적_25_비율	0.015707	50대(남자)	0.013046
8	70대(남자)	1.744006	전용면적_50_비율	0.007805	장기비율	2.275227e-02	공가수	0.011459	A비율	0.012721
9	공가수	1.537104	40대(남자)	0.007094	10대미만(여자)	2.271154e-02	전용면적_75_비율	0.008898	전용면적_50_비율	0.010546
10	40대(여자)	1.437015	80대(남자)	0.005471	40대(남자)	1.955921e-02	버스정류장	0.007523	층세대수	0.010426
11	A비율	1.303885	버스정류장	0.005327	20대(남자)	1.843438e-02	전용면적_35_비율	0.007452	초장기비율	0.009783
12	50대(여자)	1.283817	전용면적_55_비율	0.004749	초장기비율	1.535145e-02	전용면적_45_비율	0.006822	전용면적_80_비율	0.006374
13	20대(남자)	1.174396	전용면적_70_비율	0.004343	전용면적_50_비율	1.059298e-02	전용면적_50_비율	0.006344	전용면적_75_비율	0.005340
14	ACEFI비율	1.025079	A비율	0.003950	전용면적_25_비율	1.046280e-02	상가비율	0.006117	ACEFI비율	0.005067
15	은전면허소지자비율	0.981280	20대(남자)	0.003837	연면적	9.461919e-03	전용면적_80_비율	0.005960	80대(남자)	0.004903
16	전용면적_55_비율	0.926011	은전면허소지자비율	0.003717	전용면적_30_비율	8.079645e-03	전용면적_55_비율	0.005815	30대(남자)	0.004743
17	전용면적_50_비율	0.922697	10대미만(여자)	0.003602	10대(남자)	7.300888e-03	40대(남자)	0.005506	70대(남자)	0.004732
18	100대(여자)	0.918270	30대(남자)	0.003537	지하철역	7.246882e-03	아파트비율	0.005206	90대(남자)	0.004622
19	초장기비율	0.894540	전용면적_80_비율	0.003403	80대(남자)	7.118584e-03	충청남도	0.004945	40대(남자)	0.004401
20	전용면적_80_비율	0.885237	50대(여자)	0.003290	자전거도로비율	6.057231e-03	90대(남자)	0.004870	전용면적_45_비율	0.004313

- 모델별 변수 중요도를 확인해보고, 유사한 결과를 가진 모델을 그룹핑
- 유사한 모델들로 VotingRegression / 다른 그룹 모델들로 VotingRegression 해서 결과치 확인

Train / Test

① Train 데이터를 스플릿 후 학습시키고 MAE를 확인

② Test 데이터로 예측해 제출했을 시 MAE와, Train 데이터 전체를 학습시키고 MAE 확인

③ Test 데이터로 예측해 제출했을 시 MAE가 큰 차이가 없음.

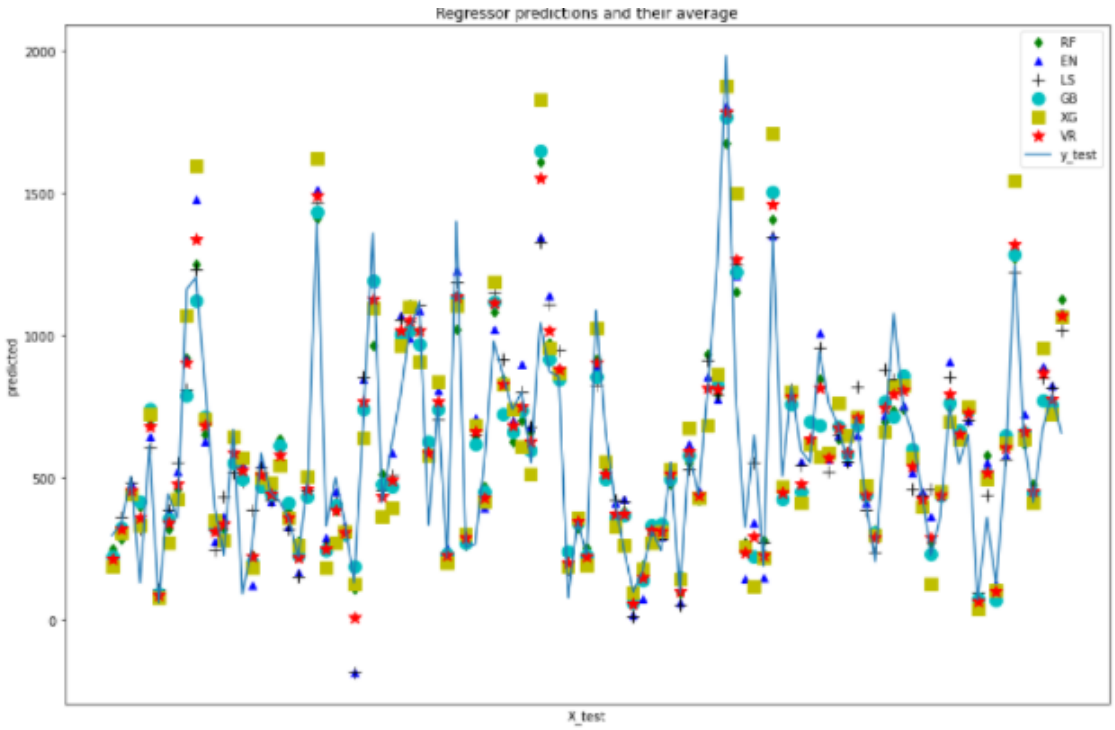
```
# lr.fit(X_train, y_train)
# print('LinearRegression Test MAE : {}'.format(mean_absolute_error(y_test, lr.predict(X_test))),
#       'LinearRegression Train MAE : {}'.format(mean_absolute_error(y_train, lr.predict(X_train))), sep='\n')

rfr.fit(X_train,y_train)
print('RandomForest Test MAE : {}'.format(mean_absolute_error(y_test, rfr.predict(X_test))),
      'RandomForest Train MAE : {}'.format(mean_absolute_error(y_train, rfr.predict(X_train))), sep='\n')

ela.fit(X_train, y_train)
print('ElasticNet Test MAE : {}'.format(mean_absolute_error(y_test, ela.predict(X_test))),
      'ElasticNet Train MAE : {}'.format(mean_absolute_error(y_train, ela.predict(X_train))), sep='\n')

rg.fit(X_train,y_train)
print('Ridge Test MAE : {}'.format(mean_absolute_error(y_test, rg.predict(X_test))),
      'Ridge Train MAE : {}'.format(mean_absolute_error(y_train, rg.predict(X_train))), sep='\n')

las.fit(X_train,y_train)
print('Lasso Test MAE : {}'.format(mean_absolute_error(y_test, las.predict(X_test))),
      'Lasso Train MAE : {}'.format(mean_absolute_error(y_train, las.predict(X_train))), sep='\n')

xg.fit(X_train,y_train)
print('Xgboost Test MAE : {}'.format(mean_absolute_error(y_test, xg.predict(X_test))),
      'Xgboost Train MAE : {}'.format(mean_absolute_error(y_train, xg.predict(X_train))), sep='\n')
```

```
C:\Users\admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: DataConversionWarning: A column-vector y was p
a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  """
RandomForest Test MAE : 136.63694117647057
RandomForest Train MAE : 46.10139053254438
ElasticNet Test MAE : 133.96658477043772
ElasticNet Train MAE : 106.15240318900389
Ridge Test MAE : 139.51963003322697
Ridge Train MAE : 99.17125578294554
Lasso Test MAE : 138.59858864709358
Lasso Train MAE : 100.15676876352259
C:\Users\admin\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:532: ConvergenceWarning: Ob
```

모델 리스트 ⇒ 모델별 MAE 확인 ⇒ VotingRegressor로 모델별 묶음 테스트, MAE 확인

```
result_test
```

	Linear	ExtraTrees	Gradient	RandomForest	ElasticNet	Ridge	Lasso	XGB	Catboost	Voting1	Voting2	Voting3	Voting4	Voting5
0	106.978	115.321	119.987	116.142	101.58	104.659	99.874	126.424	118.826	100.481	100.112	101.751	97.532	103.53




```
VotingRegressor MAE         :  105.43213155853978
RandomForestRegressor MAE   :  112.19922330097084
ElasticNet MAE              :  109.4435950638574
Lasso MAE                   :  108.34440545184215
GradientBoost MAE           :  113.17601655846391
XGBoost MAE                 :  127.1307803403984
```

- 모델링 후 결과를 시각화 해 본 결과, **Voting Regression의 결과가 실제값과 가장 유사**하게 나타남

활용 모델 ⇒ VotingRegressor([('XGBRegressor', xg), ('ElasticNet', ela), ('Lasso', las)])로 결정

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4cb61c5d-1968-4c26-8696-25ba4d9e2817/_2.TRAIN_비율_데이터_생성.ipynb

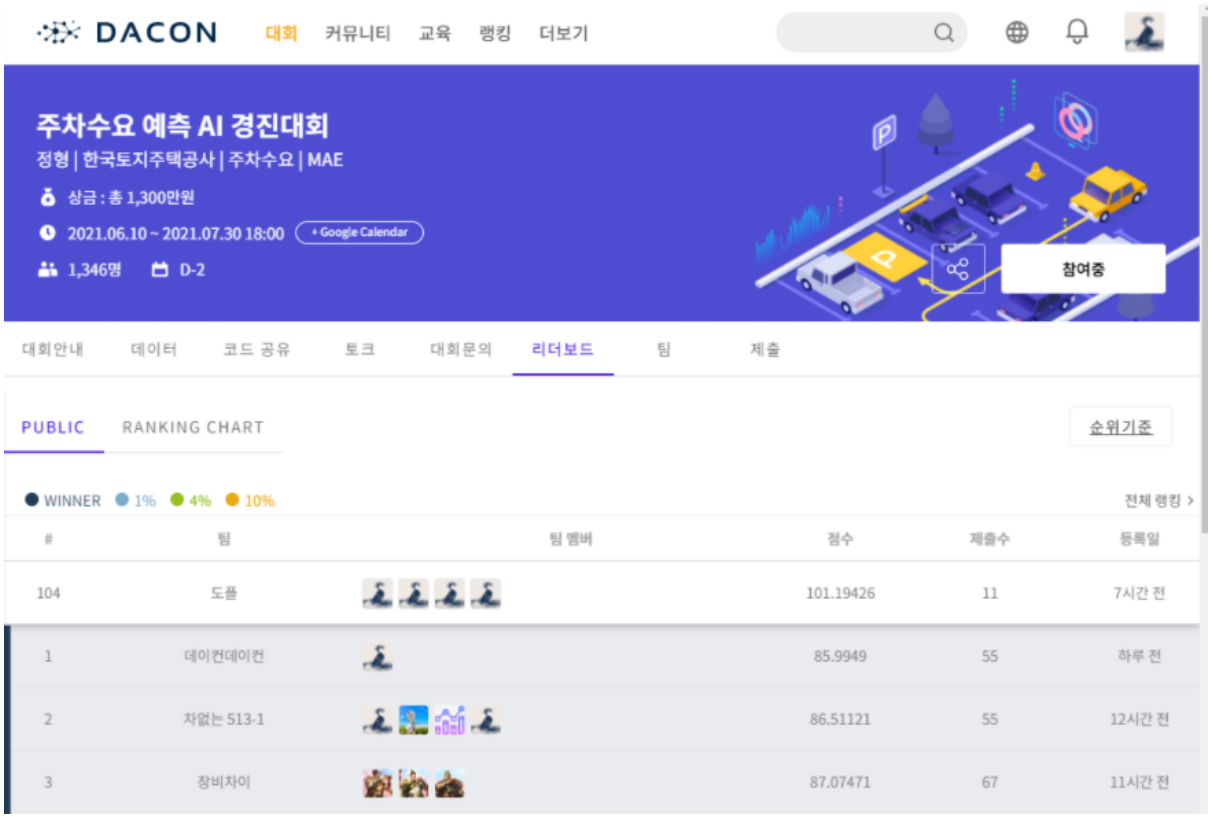
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/9b8951ca-89ee-414a-9bcd-79cbffe858f1/_2.TEST_비율_데이터_생성.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/418e91b9-898c-467e-8987-9963a2b4cfa9/_2.모델_생성_및_제출_생성.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/652f0687-2a5e-4f48-a4c1-df9fe8e98a09/model_ereg1_ratio.pickle

- 피클 파일로 만들어 추후 활용 가능성 높임

(이 외에도 all subset regression을 활용해보려 했으나, 컴퓨팅 시간이 너무 오래걸려 포기, 추후 대회 종료 후 추가로 시행 예정)



- 현재 Dacon 1346명 참여 중 104등 기록

  MAE기준 120→110→101 : 하루 10씩 꾸준히 줄이는 중.

- 제출 팀 수 기준 약 480여 팀 참여중, 20% 이내 목표

VI. 서비스 구현 및 활용 방안

1. 대시보드 구축

- 단지 정보를 입력하면 해당 단지의 예상 주차대수를 산출하는 페이지를 구축
- 현재 혹은 앞으로 들어설 아파트단지들의 정보를 취합해 주차면 과대/과소 평가된 부분들을 체크 가능

대시 실행 화면

[4]:
```
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_bootstrap_components as dbc
import dash_table
import pandas as pd
from dash.dependencies import Input, Output, State
```

[5]:
```
input_code = \
dcc.Input(
    id = 'input-code',
    type="text",
    placeholder = "단지코드"
)


input_total_household = \
dcc.Input(
    id = 'input-total-household',
    type="number",
    placeholder = "총 세대수"
)


rent_type_dropdown = \
dcc.Dropdown(
    id = 'rent-type-dropdown',
    options = [{'label':'아파트', 'value': '아파트'},
               {'label':'상가', 'value': '상가'}
              ],
    placeholder = '임대건물 구분',
    style={'width': '160px'}
)


city_dropdown = \
dcc.Dropdown(
    id = 'city-dropdown',
    options = [{'label' : '강원도', 'value' : '강원도'},
               {'label' : '경기도', 'value' : '경기도'},
               {'label' : '경상남도',  'value' : '경상남도'},
               {'label' : '경상북도',  'value' : '경상북도'},
               {'label' : '광주광역시',  'value' : '광주광역시'},
               {'label' : '대구광역시',  'value' : '대구광역시'},
               {'label' : '대전광역시',  'value' : '대전광역시'},
               {'label' : '부산광역시', 'value' : '부산광역시'},
               {'label' : '서울특별시',  'value' : '서울특별시'},
               {'label' : '세종특별자치시',  'value' : '세종특별자치시'},
               {'label' : '울산광역시',  'value' : '울산광역시'},
               {'label' : '전라남도',  'value' : '전라남도'},
               {'label' : '전라북도',  'value' : '전라북도'},
               {'label' : '제주특별자치도',  'value' : '제주특별자치도'},
               {'label' : '충청남도', 'value' : '충청남도'},
               {'label' : '충청북도', 'value' : '충청북도'}
              ],
    placeholder = '지역',
    style={'width': '160px'}
)

supply_type_dropdown = \
dcc.Dropdown(
    id = 'supply-type-dropdown',
    options = [{'label': '공공분양', 'value' : '공공분양'},
               {'label': '공공임대(10년)', 'value' : '공공임대(10년)'},
               {'label': '공공임대(50년)', 'value' : '공공임대(50년)'},
               {'label': '공공임대(5년)', 'value' : '공공임대(5년)'},
               {'label': '공공임대(분납)', 'value' : '공공임대(분납)'},
               {'label': '국민임대', 'value' : '국민임대'},
               {'label': '영구임대', 'value' : '영구임대'},
               {'label': '임대상가', 'value' : '임대상가'},
               {'label': '장기전세', 'value' : '장기전세'},
               {'label': '행복주택', 'value' : '행복주택'}
              ],
    placeholder = '공급유형',
    style={'width': '160px'}
)


input_area = \
dcc.Input(
    id = 'input-area',
    type="number",
    placeholder = "전용면적"
)

input_area_count = \
dcc.Input(
    id = 'input-area-count',
    type="number",
    placeholder = "전용면적별 세대수")

input_empty_area = \
dcc.Input(
    id = 'input-empty-area',
    type="number",
    placeholder = "공가수"
)

qualification_type_dropdown = \
dcc.Dropdown(
    id = 'qulification-type-dropdown',
    options = [{'label': 'A', 'value' : 'A'},
               {'label': 'B', 'value' : 'B'},
               {'label': 'C', 'value' : 'C'},
               {'label': 'D', 'value' : 'D'},
               {'label': 'E', 'value' : 'E'},
               {'label': 'F', 'value' : 'F'},
               {'label': 'G', 'value' : 'G'},
               {'label': 'H', 'value' : 'H'},
               {'label': 'I', 'value' : 'I'},
               {'label': 'J', 'value' : 'J'},
               {'label': 'K', 'value' : 'K'},
               {'label': 'L', 'value' : 'L'},
               {'label': 'M', 'value' : 'M'},
               {'label': 'N', 'value' : 'N'},
               {'label': 'O', 'value' : 'O'}
              ],
    placeholder = '자격유형',
    style={'width': '160px'}
)

input_rent_deposit = \
```

대시 코드 작업

- 현재 대시 작업 중

2. 활용 방안

- 정교한 수요예측 모델로 발전시켜 공공 아파트 뿐 아니라, 민간 아파트에도 수요 예측 프로그램으로 활용 가능하다.

  1) 신규 아파트 설계시, 주차면수 과대 평가로 인한 자원 낭비 최소화

  2) 기존 아파트단지의 경우 과대 평가된 주차면수를 유휴 공간으로 활용 가능

  이처럼 주차수요 예측 프로그램은 신규 단지의 주차면수 책정 뿐 아니라, 기존 아파트들에게도 좋은 기회가 될 것이다.

- 시간대별, 요일별 모델로 추가 개선 및 향상

  도심 상업지구만큼은 아니지만 아파트의 경우에도 주차대수의 요일별/시간대별 편차가 존재할 것이기 때문에, 이를 활용해 최적 주차면수를 도출할 수 있도록 모델을 개선할 수 있을 것

  또한 도심 상업지구의 경우 주차 수요가 요일별/시간별로 편차가 크기 때문에 위의 경험을 바탕으로 추가적인 데이터를 활용해 도심 주차수요 예측 모델링으로 진행할 경우 현재의 도심 주차난 등을 해소하는 데 기여할 수 있을 것

3. 유의점

1. 주어진 정량적 수치로 모델링한 경우보다, 대다수를 비율화시켜서 모델링한 케이스에서 MAE가 감소
2. 주어진 라벨링 데이터(카테고리)의 경우에도 주어진 대로 인코딩해 활용하는 경우보다 가설을 세우고 이에 맞추어서 그룹핑하여 활용할 경우 예측력이 높아지는 경향
3. 아웃라이어의 경우 모델링 시도 중 타겟 및 인코딩된 칼럼을 제외하고 모두 IQR을 활용, 이상치경계로 수정한 케이스에서 MAE가 유의미하게 감소하지 않았다. 모델링 자체의 문제인지, 데이터 자체의 문제인지 좀 더 깊이있는 논의 필요
4. 주어진 데이터 외에, 추가적인 데이터를 변수로 넣고 모델링한 케이스에서 MAE가 감소
    1. 다만 이 경우엔, 효과적인 데이터가 아닌 큰 의미 없는 데이터를 추가한 경우 MAE 감소 X
    2. 결국 적절한 데이터를 탐색하고, 알맞게 추가하는 것이 중요
5. all subset regression 의 경우 칼럼 수가 증가함에 따라 확인해야할 subset 갯수가 크게 급증하므로 현재의 하드웨어적인 상황에서 적용 어려움 → 대안으로 stepwise regression(backward) 활용

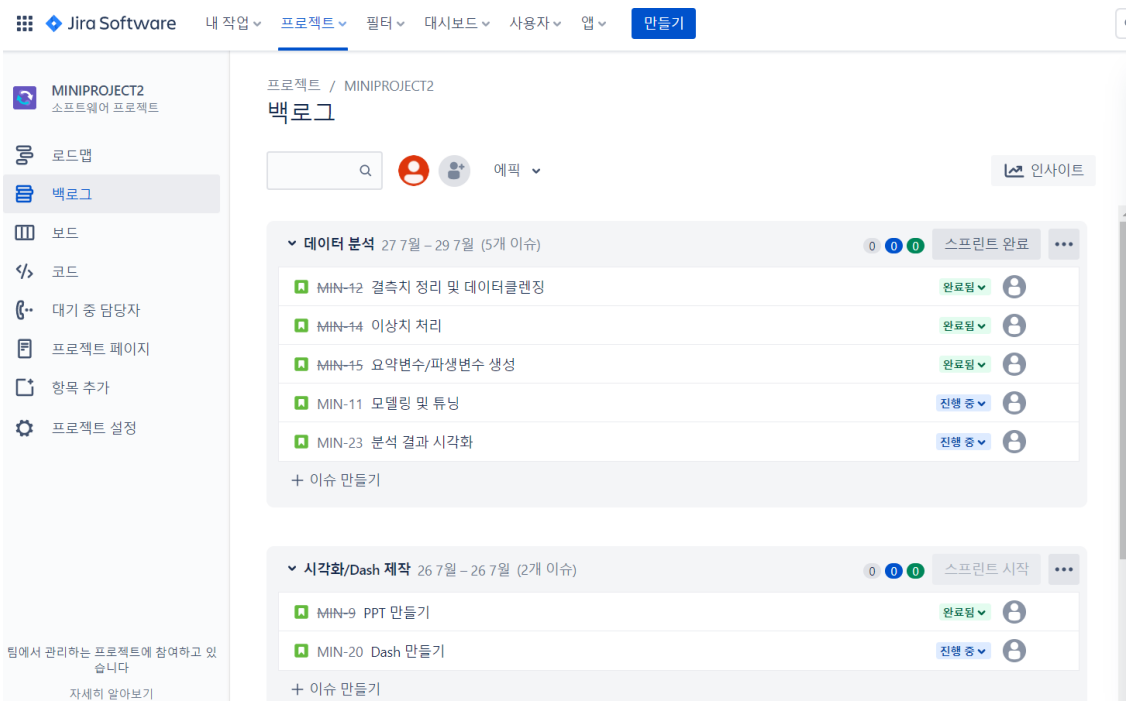
```
[*]: k = 64
MAE_list, R_squared_list, feature_list = [], [], []
numb_features = []
history = []


for k in tqdm_notebook(range(64, 54, -1), desc = 'Loop...'):
    counter = k
    counter1 = 1
    for combo in itertools.combinations(X.columns, k):
        a = list(combo)
        a.sort()
        if a not in history:
            history.append(a)
            tmp_result = fit_rfr_reg(X_train[list(combo)],X_test[list(combo)], y_train, y_test)
            MAE_list.append(tmp_result[0])
            R_squared_list.append(tmp_result[1])
            feature_list.append(combo)
            numb_features.append(len(combo))
            print(counter, counter1)
            counter1 += 1
        else:
            pass


df = pd.DataFrame({'numb_features':numb_features,
                   'MAE':MAE_list,
                   'R_squared':R_squared_list,
                   'features':feature_list})
```

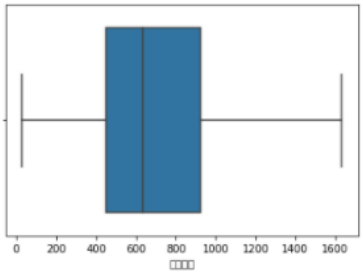
```
C:\Users\admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: TqdmDeprecationWarning: This function
will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  import sys
```

Loop...: 40% 4/10 [4:33:59<10:40:32, 6405.38s/it]

```
64 1
63 1
63 2
63 3
63 4
```

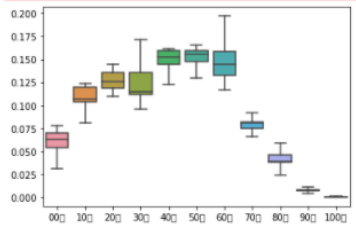
6. 최근 배운 협업 툴을 활용해볼 수 있는 기회가 되었고, 더불어 ML 라이브러리들을 활용해보고, 어떤 경우에 예측치에 문제가 생기고, 또 어떤 경우에 예측이 조금 더 향상되는지 여러 시행착오를 직접 겪어볼 수 있는 기회가 되었음






```
iqr = (X['총세대수'].describe()[6] - X['총세대수'].describe()[4])* 1.5

X['총세대수'] = np.where(X['총세대수'] < (X['총세대수'].describe()[4] - iqr),
                    (X['총세대수'].describe()[4] - iqr), X['총세대수'])
X['총세대수'] = np.where(X['총세대수'] > (X['총세대수'].describe()[6] + iqr),
                    (X['총세대수'].describe()[6] + iqr),
                    X['총세대수'])
```

총세대수 iqr 활용 outlier 처리




```
iqr = (X['10대'].describe()[6] - X['10대'].describe()[4])* 1.5

X['10대'] = np.where(X['10대'] < (X['10대'].describe()[4] - iqr),
                    (X['10대'].describe()[4] - iqr), X['10대'])
X['10대'] = np.where(X['10대'] > (X['10대'].describe()[6] + iqr),
                    (X['10대'].describe()[6] + iqr),
                    X['10대'])

iqr = (X['30대'].describe()[6] - X['30대'].describe()[4])* 1.5

X['30대'] = np.where(X['30대'] < (X['30대'].describe()[4] - iqr),
                    (X['30대'].describe()[4] - iqr), X['30대'])
X['30대'] = np.where(X['30대'] > (X['30대'].describe()[6] + iqr),
                    (X['30대'].describe()[6] + iqr),
                    X['30대'])

iqr = (X['50대'].describe()[6] - X['50대'].describe()[4])* 1.5

X['50대'] = np.where(X['50대'] < (X['50대'].describe()[4] - iqr),
                    (X['50대'].describe()[4] - iqr), X['50대'])
X['50대'] = np.where(X['50대'] > (X['50대'].describe()[6] + iqr),
                    (X['50대'].describe()[6] + iqr),
                    X['50대'])

iqr = (X['60대'].describe()[6] - X['60대'].describe()[4])* 1.5

X['60대'] = np.where(X['60대'] < (X['60대'].describe()[4] - iqr),
                    (X['60대'].describe()[4] - iqr), X['60대'])
X['60대'] = np.where(X['60대'] > (X['60대'].describe()[6] + iqr),
                    (X['60대'].describe()[6] + iqr),
                    X['60대'])

iqr = (X['70대'].describe()[6] - X['70대'].describe()[4])* 1.5

X['70대'] = np.where(X['70대'] < (X['70대'].describe()[4] - iqr),
                    (X['70대'].describe()[4] - iqr), X['70대'])
X['70대'] = np.where(X['70대'] > (X['70대'].describe()[6] + iqr),
                    (X['70대'].describe()[6] + iqr),
                    X['70대'])
```

연령별 인구비율데이터 iqr활용 outlier 처리