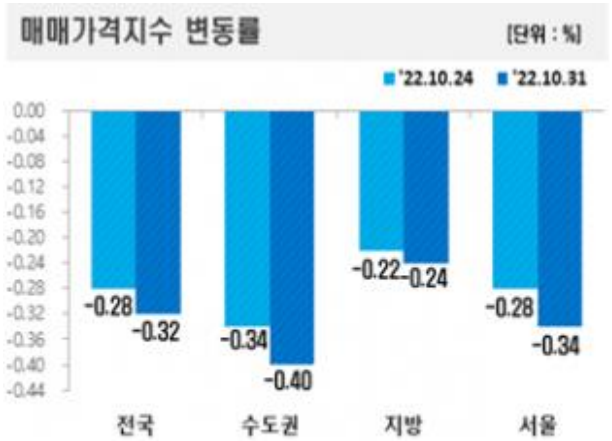
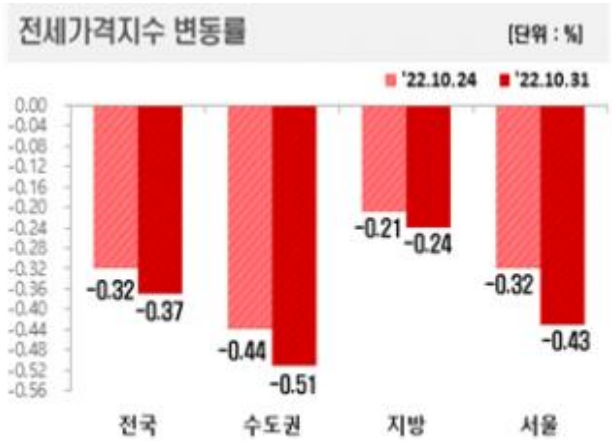
서울시 부동산 분석

정재훈, 허진욱

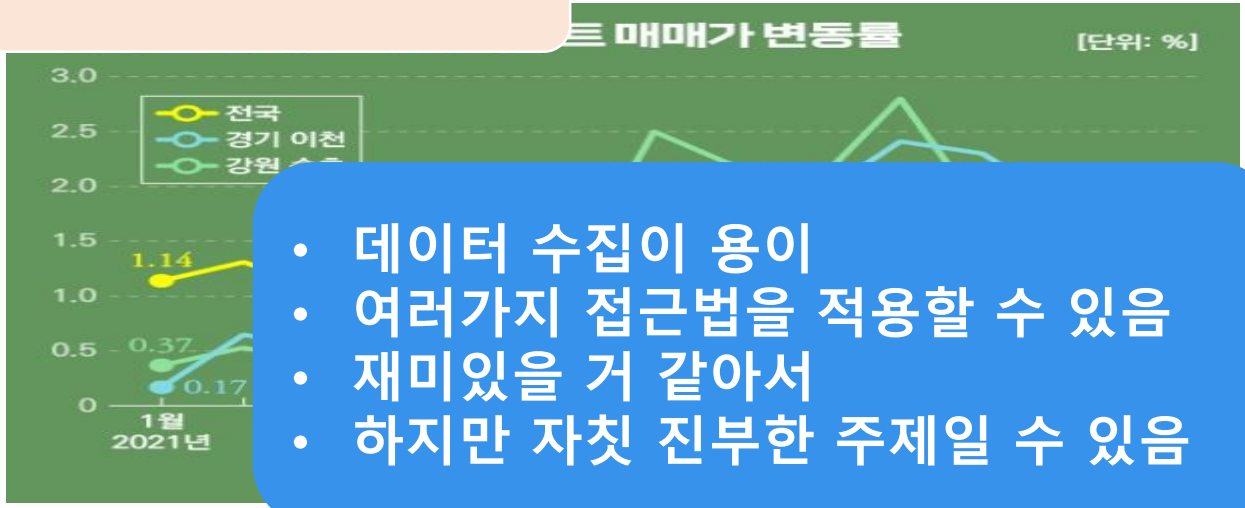
WHY





=한국부동산원

부동산은 항상 관심의 대상



- 데이터 수집이 용이
- 여러가지 접근법을 적용할 수 있음
- 재미있을 거 같아서
- 하지만 자칫 진부한 주제일 수 있음

목 차

① Feature 수집
② Sampling & Geo-Data 추출
③ Outlier 처리
④ Feature
⑤ 머신러닝

아파트 가격 예측

상권분석

① 상가 임대료
② 시각화
③ ARIMA

행정동 분석

① Feature 수집
② Feature 분석
③ Feature 시각화
④ 군집화

데이터 수집

DATA 공공데이터포털 .GO.KR

R-ONE 부동산통계정보

Google

서울 열린데이터 광장
SEOUL OPEN DATA PLAZA

우리마을가게 상권분석서비스

한국감정원 부동산연구원

아파트 가격 예측

- Feature:
  위도, 경도, 건축 년도, 인구 밀도, 학원 개수,
  거리(지하철역, 대학병원, 중심상권, 전통시장, 공원),
  투기지구, 강남3구, 마용성, 노도강 등
- Output:
  2022년 9월 기준 환산 m^2 당 가격

Feature 수집

PS) 어린적 따라했던 동요들~~~~
그 속에 부동산의 진리가 숨어있었습니다.

1) "엄마야 누나야 강변살자"에서는 결국 강/ 호수 주변에 위치한 집이 핵심 지역임을 암시.

2) "기찻길옆오막살이"에서는 역세권에 살아야 아기가 잘도 잘 수있다는 의미.

3) "두껍아 헌집줄게새집다오", 재개발 재건축을 노리는 전략이 있음을 암시.

4) "곰세마리가 한집에 있어", 최소 20평, 쓰리룸인 30평대가 주력.

5) "깊은산속옹달샘 누가와서 먹나요?" 그린벨트지역과 같은 개발특수지역을 먹기가 쉽지 않음을 암시

일찍이 동요들은 부동산 비법을 암시하고 있었....

실거래가 자료

Feature : 거래가격, 거래일, 층, 건축년도, 주소

추가 자료

좌표 기반 Feature :
지하철역, 공원, 상가, 중심상권, 전통시장, 대학교

행정동/법정동 기준 Feature:
인구밀도, 학원갯수, 투기지구/마용성/노도강 유무

Feature 수집

```
# 공공데이터 긁어오기 & 저장

url = 'http://openapi.molit.go.kr/OpenAPI_ToolInstallPackage/service/rest/RTMSOBJSvc/getRTMSDataSvcAptTradeDev'#아파트 실거래가 데이터

df_raw = None
date = None

for y in range(2017,2023) : #5년치 데이터 받기
    for m in range(1,13) :
```

공공 데이터 API로 아파트 실거래가 거래 자료 Crawling
- 거래가 많지 않으므로 5년치 데이터 합침

거래일	거래금액	전용면적	구주소	도로명주소	아파트	층	건축년도	법정동	구
2018-03-12	75000	117.19	화곡동 11	공항대로	우장산롯	14	2003	화곡동	강서구
2019-08-24	54000	23.7	잠원동 71	잠원로3길	킴스빌리지	13	1996	잠원동	서초구
2022-08-25	46300	48.93	공릉동 10	화랑로51길	비선아파트				
2017-06-13	148900	126.18	방이동 89	양재대로	올림픽선수				
2018-02-12	98000	120.41	여의도동	의사당대로	롯데캐슬임				
2020-06-25	84500	114.95	청량리동	제기로 13	한신	6	1997	청량리동	동대문구

중복, Null, 거래 취소, 주소 등 Preprocessing

지역		'17.1Q		'17.2Q		'17.3Q		'17.4Q		'18.1Q		'18.2Q	
		지수	변동률	지수	변동률	지수	변동률	지수	변동률	지수	변동률	지수	변동률
서울	종로구	94.8	-	96.2	1.49	98.6	2.51	100	1.44	103.4	3.43	105.9	2.35
	중구	92.6	-	94.2	1.72	97.6	3.60						
	용산구	88.2	-	91	3.14	97	6.59						
	성동구	86.5	-	90.1	4.12	95.5	6.09						
	광진구	88.9	-	90.7	2.11	95.1	4.82						
	동대문구	93	-	94.4	1.53	97.6	3.43	100	2.43	105.3	5.28	112.2	6.57

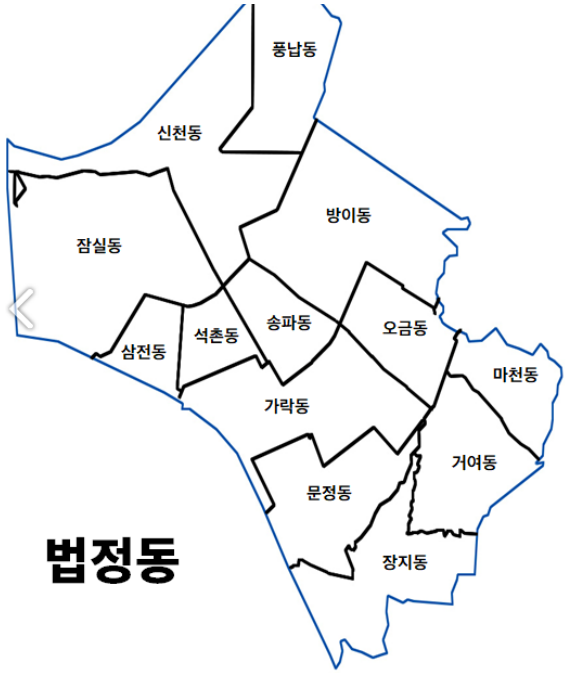
실거래가격 지수를 이용
과거 거래가 -> 현재 가격으로 환산

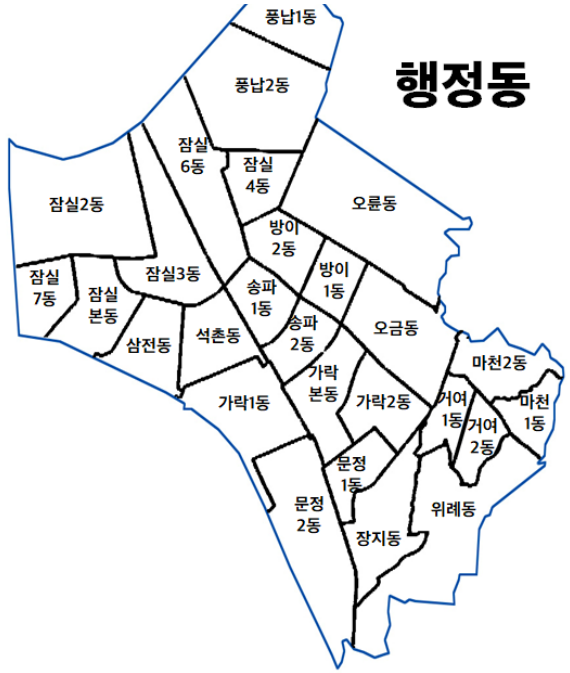
Feature 수집

법정동 : 옛부터 전래되어온 동명으로 개인의 권리·의무 및 법률행위시 주소로 사용되는 동명칭
행정동 : 주민의 편의와 행정능률을 위하여 적정한 규모와 인구를 기준으로 동주민센터를 설치운영하는 동명칭

구별현황 - 행정동 426개동,법정동 466개

법정동 : 옛부터 전래되어온 동명으로 개인의 권리·의무 및 법률행위시 주소로 사용되는 동명칭
행정동 : 주민의 편의와 행정능률을 위하여 적정한 규모와 인구를 기준으로 동주민센터를 설치운영하는 동명칭





일부 데이터 :
행정동 기준(대부분)

일부 데이터 :
법정동 기준 (부동산 관련)

주택 가격 분석에서는
법정동 기준으로 통합

Sampling & Geo-Data 추출

샘플링

```
#너무 많다. 샘플링 후 배치 테스트
batch_df = apart_raw_df[(apart_raw_df['전용면적'] < 85) & (apart_raw_df['전용면적'] > 83)] # 국평 뽑기
batch_df = batch_df.sample(frac = 0.1)  # 임의의 샘플링
batch_df.reset_index(inplace = True)
len(batch_df)
```

```
12308
```

모든 경우를 다 다룰 경우
- Geo Coding API limit에 걸림
- 평수가 작을수록 평단가 왜곡
- 아파트 가격을 대표하는 표본 추출 필요

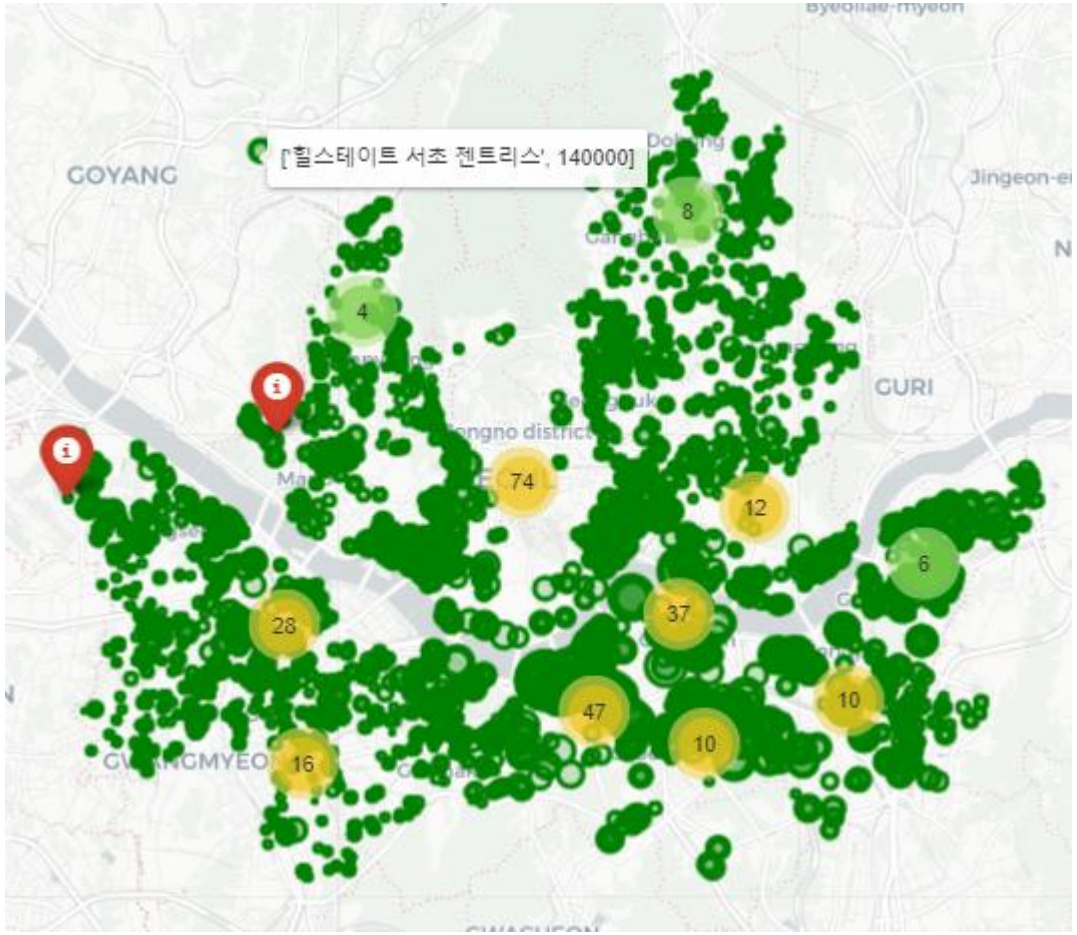
83~85m^2 (33~34평)
국민평형 데이터를 추출

DATA 각각의 위도 경도 추출

데이터 위도, 경도 찾기

```
batch_df['위경도'] = batch_df['구주소'].apply(get_location)
batch_df.dropna(inplace = True)
batch_df[['위도', '경도']] = batch_df.위경도.apply(pd.Series)

try :
    batch_df.reset_index(inplace = True)
except:
    pass

batch_df = batch_df.astype({'위도':'float'})
batch_df = batch_df.astype({'경도':'float'})

s = './batch/batch_apart'+'.csv'
batch_df.to_csv(s, index = False, encoding = 'utf-8-sig')
```

Outlier 처리



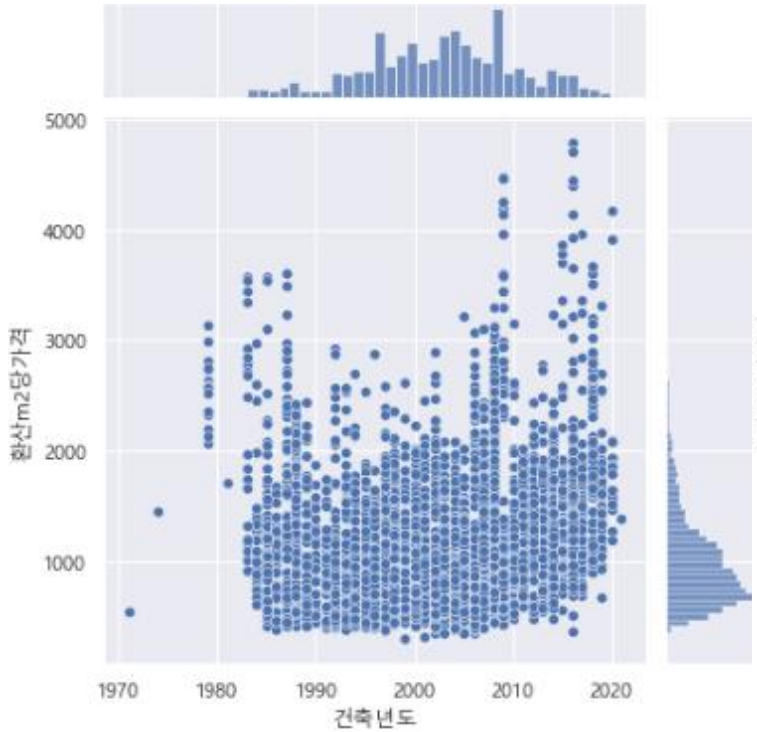
잘못 추출된 좌표

```
batch_df = pd.read_csv('./batch/batch_apart_with_coordinate.csv')

batch_df.위도[batch_df.아파트 == '힐스테이트 서초 젠트리스']= 37.45578045486
batch_df.경도[batch_df.아파트 == '힐스테이트 서초 젠트리스']= 127.0578237590
key_df = batch_df[['법정동','구', '층', '건축년도', '인구밀도', '위도', '경도
key_df.describe()
key_df.위도[batch_df.아파트 == '힐스테이트 서초 젠트리스']
```

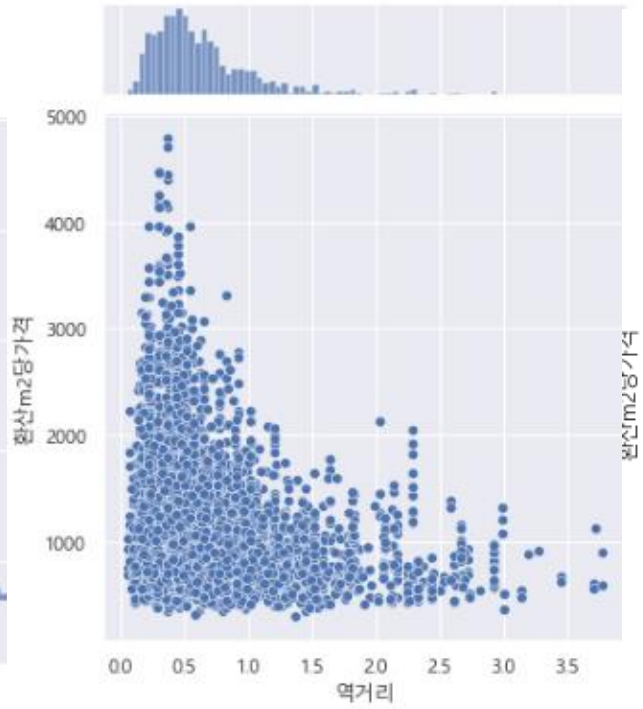
```
842     37.45578
3880    37.45578
Name: 위도, dtype: float64
```

좌표 오류 수정

Feature와 Output의 상관관계



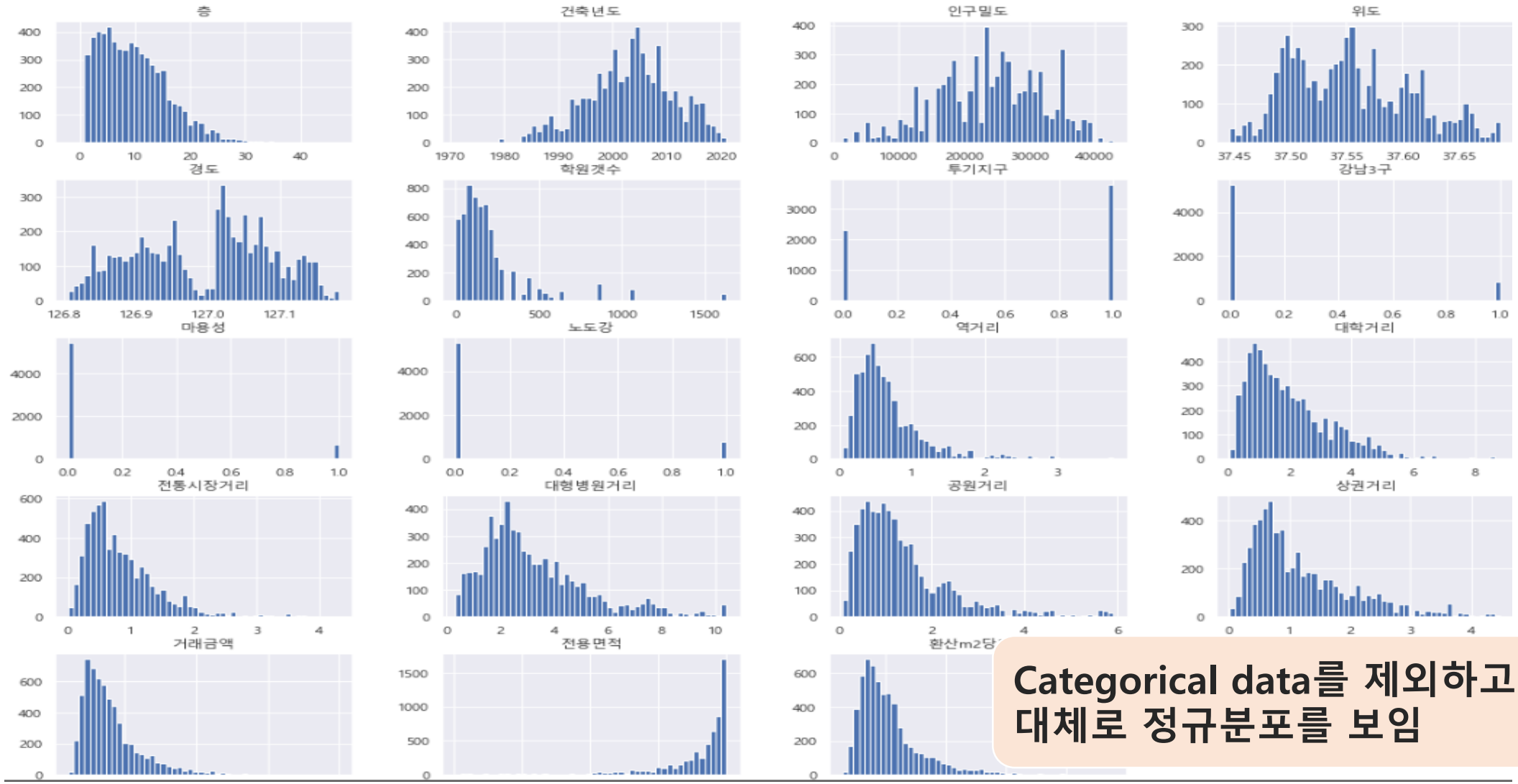
최근에 지은 아파트가 비쌈



역에 가까울수록 비쌈

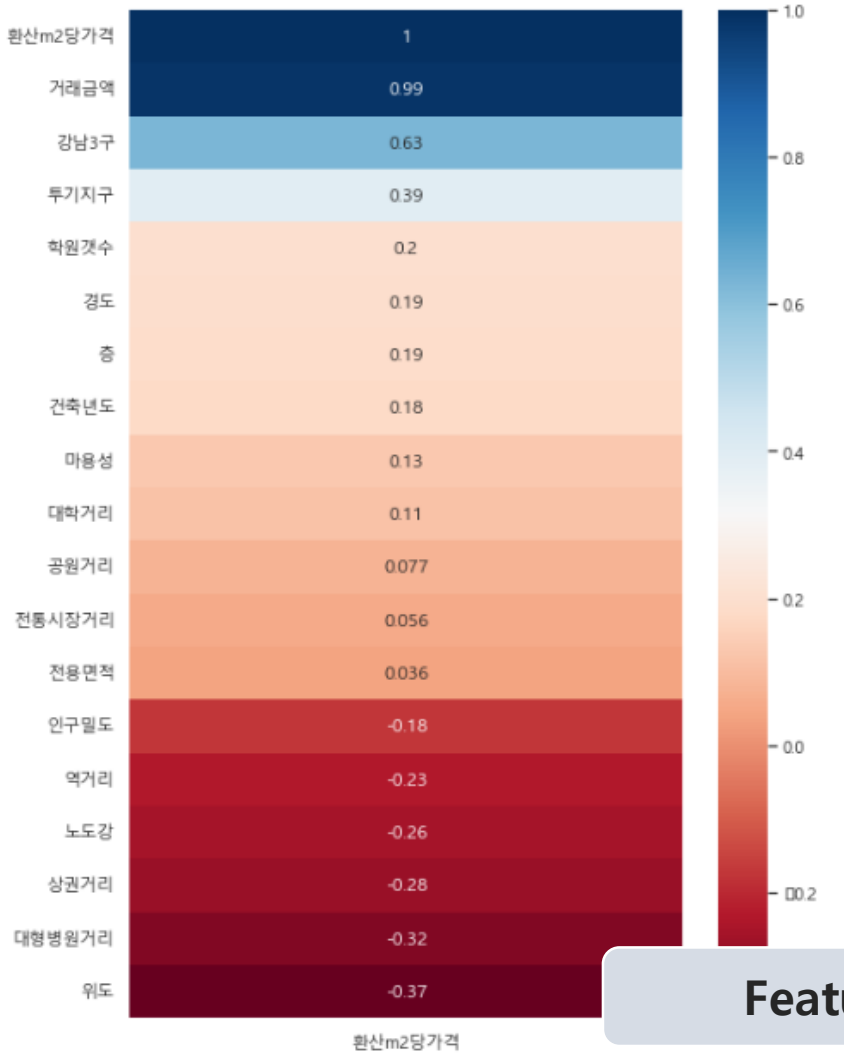
층수와 가격은 Clear cut 하지 않음

Feature Histogram 분포

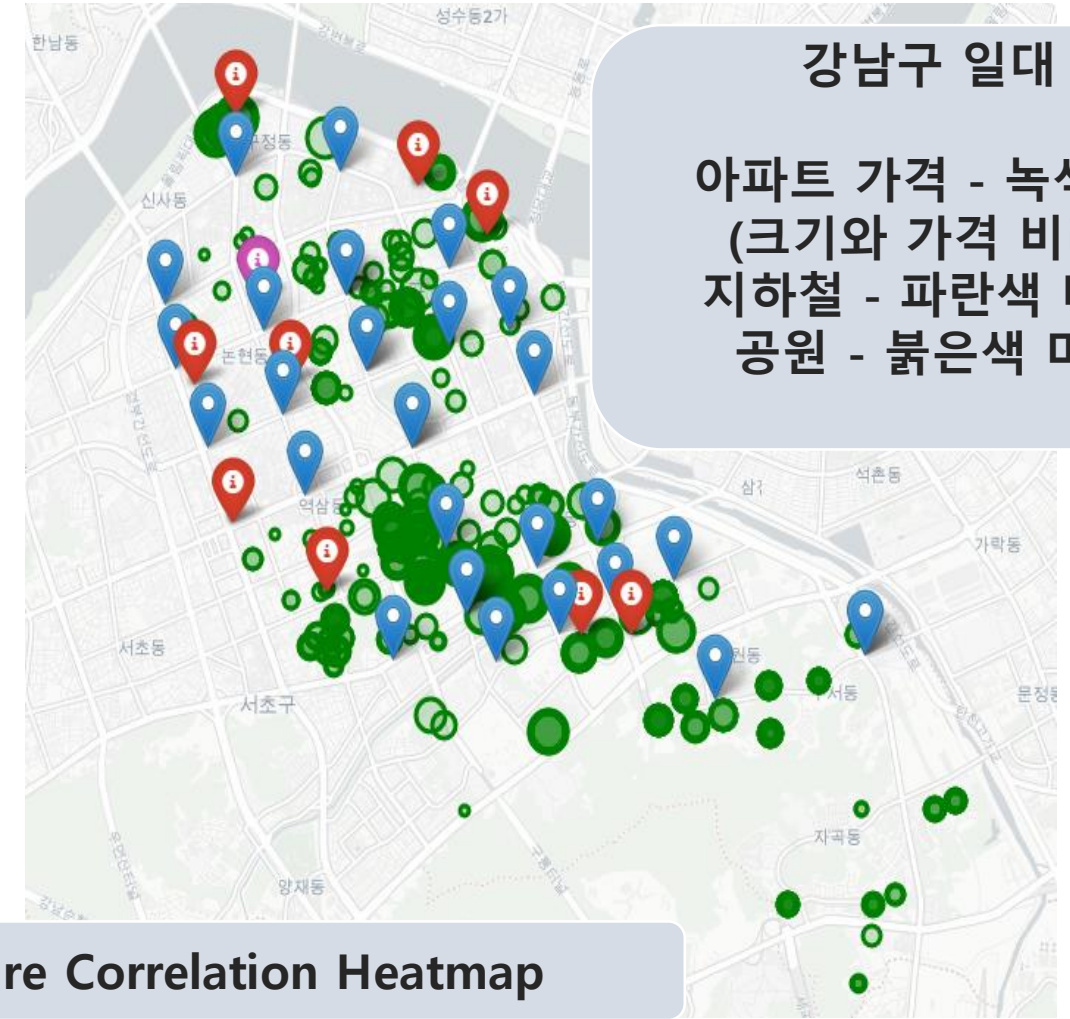


Categorical data를 제외하고 대체로 정규분포를 보임

Feature 시각화



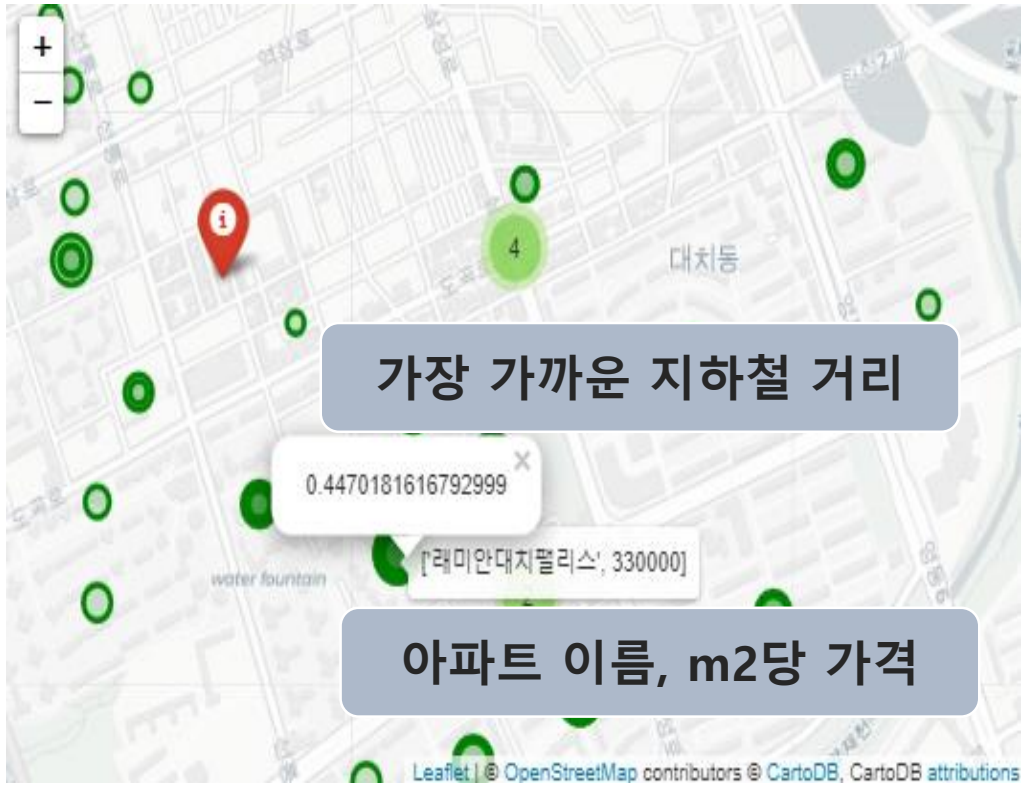
Feature Correlation Heatmap



강남구 일대

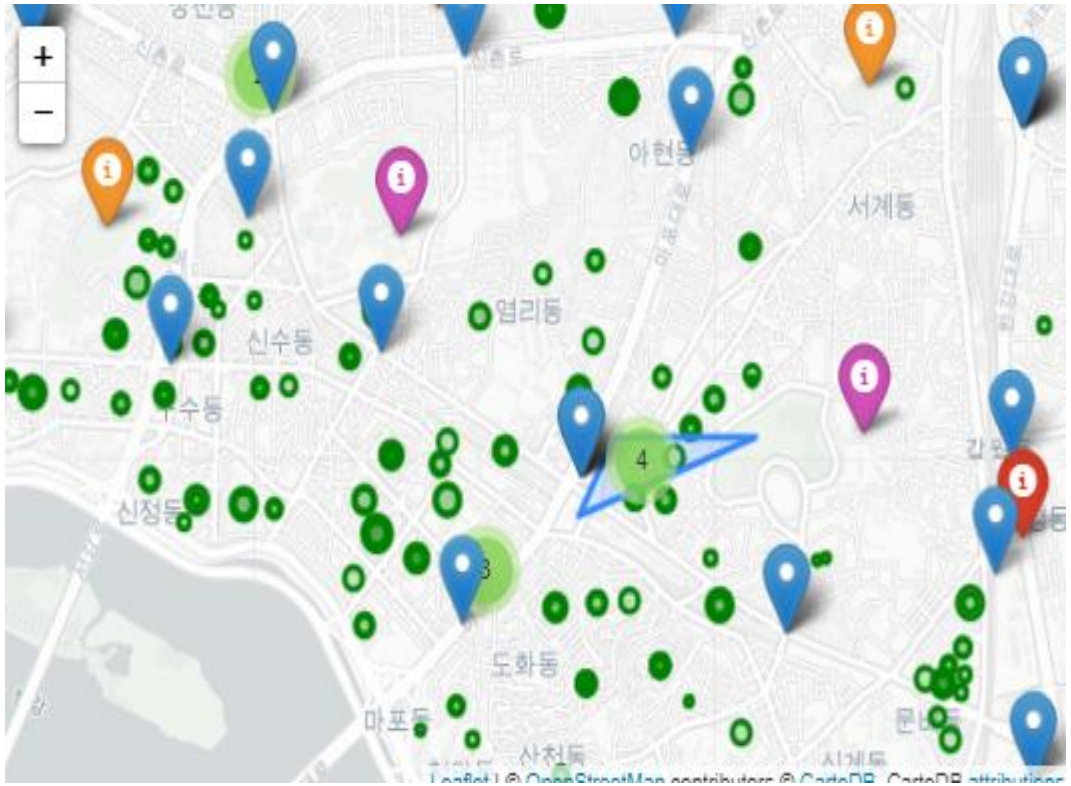
아파트 가격 - 녹색 원
(크기와 가격 비례)
지하철 - 파란색 마커
공원 - 붉은색 마커

Feature 시각화



가장 가까운 지하철 거리

아파트 이름, m2당 가격



초록 – 아파트 /보라색 – 대학교
파란색 – 지하철/ 노란색 – 공원
옅은 초록 – 상가/삼각모형- 상가 범위

Machine Learning

Linear regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

apart_price_pred = lin_reg.predict(X_test)

lin_reg_rmse = mean_squared_error(y_test, apart_price_pred, squared=False)
#lin_reg_rmse =np.sqrt(lin_reg_mse)

int(lin_reg_rmse)
```

```
6999550611590275
```

699955061

Multi layered Perceptron (layer 3개 hidden neuron 100개씩)

```
from sklearn.metrics import mean_squared_error
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

mlp_reg = MLPRegressor(hidden_layer_sizes=[100, 100, 100], random_state=42, max_iter = 1000)
pipeline = make_pipeline(StandardScaler(), mlp_reg)
pipeline.fit(X_train, y_train)
mlp_y_pred = pipeline.predict(X_test)
rmse = mean_squared_error(y_test, mlp_y_pred, squared=False)
rmse
```

```
338.1379991651182
```

338.1379991

KNN (위경도만 사용)

```
from sklearn.neighbors import KNeighborsRegressor

knn_feature = batch_df[['위도', '경도']]
knn_label = batch_df['환산m2당가격']

scaler = StandardScaler()
knn_feature_tr = scaler.fit_transform(knn_feature)

X_train, X_test, y_train, y_test = train_test_split(knn_feature_tr, knn_label, test_size=0.2, random_st

knn = KNeighborsRegressor(n_neighbors=25, weights = 'distance')
knn.fit(X_train,y_train)
apart_price_pred = knn.predict(X_test)

knn_reg_rmse = mean_squared_error(y_test, apart_price_pred, squared=False)
#knn_reg_rmse =np.sqrt(knn_reg_mse)

knn_reg_rmse
```

```
256.347221641913
```

256.3472216

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
from sklearn import tree

tree_reg = DecisionTreeRegressor()
tree_reg.fit(X_train, y_train)

apart_price_pred = tree_reg.predict(X_test)

tree_mse = mean_squared_error(y_test, apart_price_pred)
tree_rmse = np.sqrt(tree_mse)

print(tree_rmse)
```

269.8782248

Linear regression - improved

Linear regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

apart_price_pred = lin_reg.predict(X_test)

lin_reg_rmse = mean_squared_error(y_test, apart_price_pred, squared=False)
#lin_reg_rmse =np.sqrt(lin_reg_mse)

int(lin_reg_rmse)
```

```
6999550611590275
```

699955061

Linear하지 않은 categorical features:
결과에 큰 영향을 미침

도움이 되지 않는 Features 제외:
Significant Improvement!

Linear regression

```
linear_df_feature = key_df[['층','건축년도', '인구밀도
linear_df_label = key_df['환산m2당가격']
```

```
# 정규화
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
linear_df_feature_tr = scaler.fit_transform(linear_d

from sklearn.model_selection import train_test_split

X_train1, X_test1, y_train1, y_test1 = train_test_sp
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_reg = LinearRegression()
lin_reg.fit(X_train1, y_train1)

apart_price_pred = lin_reg.predict(X_test1)

lin_reg_rmse = mean_squared_error(y_test1, apart_pri
#lin_reg_rmse =np.sqrt(lin_reg_mse)

int(lin_reg_rmse)
```

```
410
```

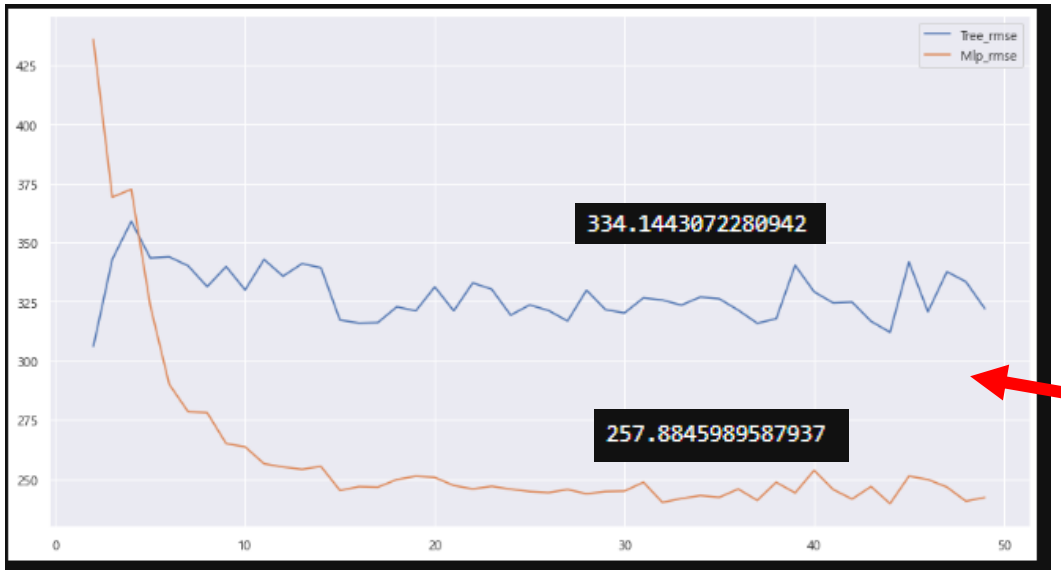
410

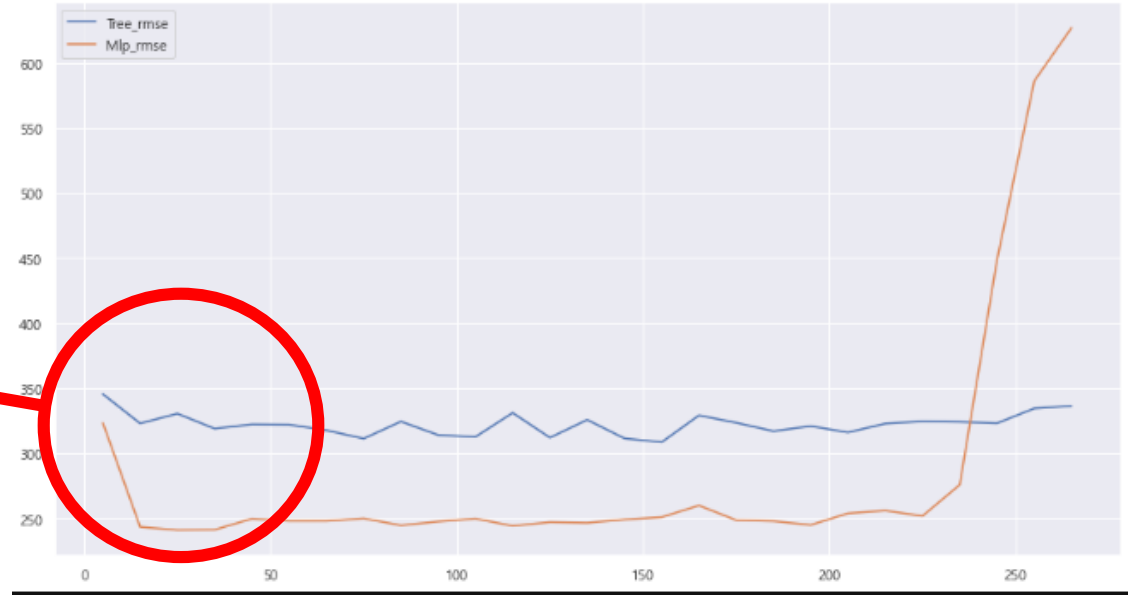
Feature 차원축소(PCA)

총 Feature 개수 276
(after one hot encoding)

Principal Component :2~275

Decision Tree와 MLP 의 RMSE 측정





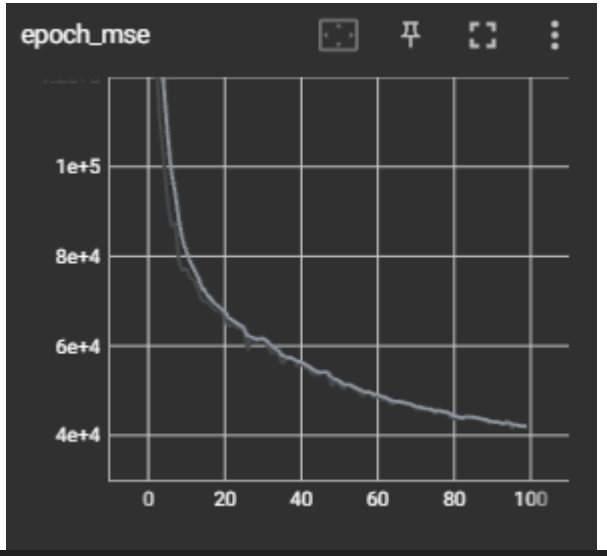
Feature가 많다고 무조건 좋은 것은 아니다.
Noise에 가까운 Feature는 제거해 줄 때 성능이 더욱 개선됨

TensorFlow 회귀

```python
import tensorflow as tf

tf.random.set_seed(42)
model = tf.keras.Sequential([
    tf.keras.layers.Dense(100, activation="relu",
                          kernel_initializer="he_normal"),
    tf.keras.layers.Dense(100, activation="relu",
                          kernel_initializer="he_normal"),
    tf.keras.layers.Dense(100, activation="relu",
                          kernel_initializer="he_normal"),
    tf.keras.layers.Dense(1)
])

model.compile(loss="mse", optimizer=tf.keras.optimizers.RMSprop(0.02), metrics=['mae', 'mse'])
history = model.fit(X_train, y_train, epochs=100)
model.evaluate(X_test, y_test)
```

```
Epoch 1/100
152/Model: "sequential_10"                                    .0849 - mse: 228961.1406
Epoc_________________________________________________________________
152/ Layer (type)                 Output Shape              Param #     .2842 - mse: 147821.1719
Epoc=================================================================
152/ dense_40 (Dense)             (None, 100)               27700       .1072 - mse: 118876.6562
Epoc
152/ dense_41 (Dense)             (None, 100)               10100       .121
Epoc
152/ dense_42 (Dense)             (None, 100)               10100       .8941
Epoc
152/ dense_43 (Dense)             (None, 1)                 101         .6215
Epoc                                                                    .8399
152/=================================================================
Total params: 48,001                                                    .5421 - mse: 85253.4922
Trainable params: 48,001
Non-trainable params: 0
_________________________________________________________________
```

TensorBoard

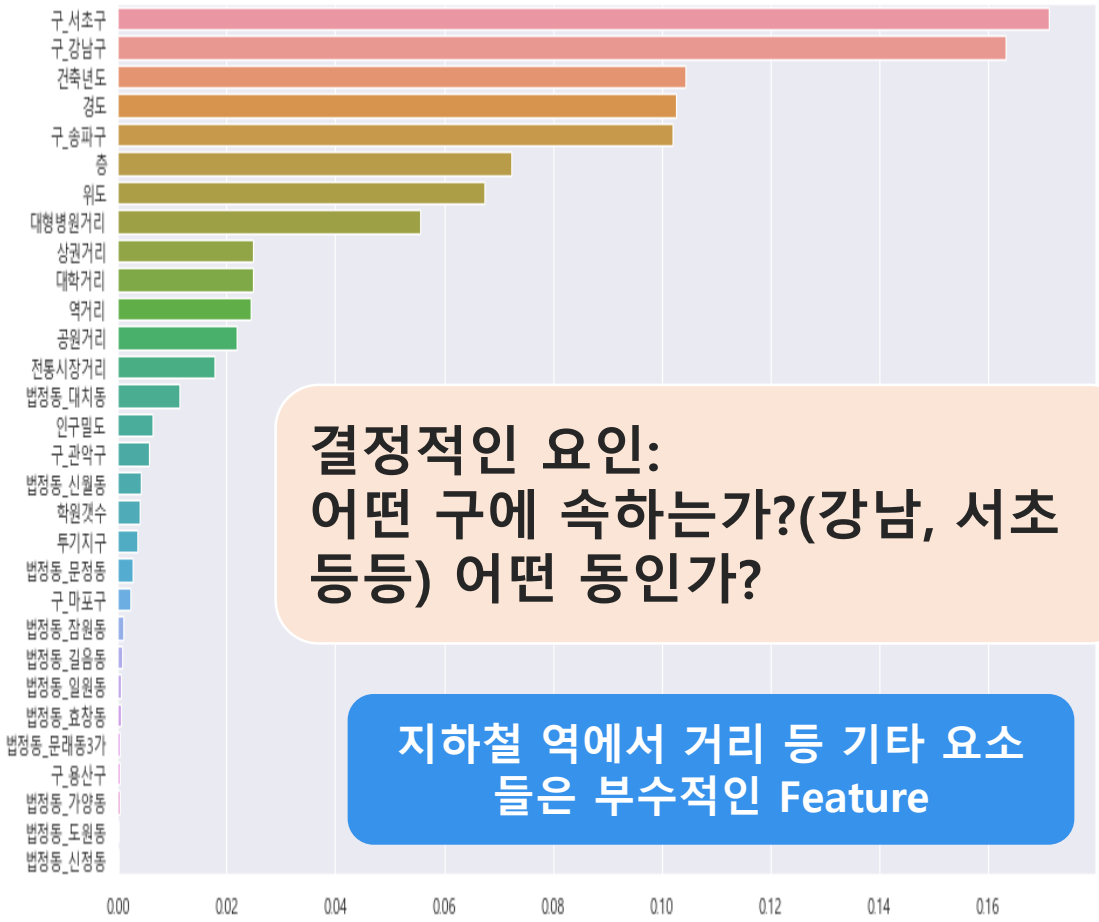



```python
loss, mae, mse = model.evaluate(X_test, y_test, verbose=2)
np.sqrt(mse)
```

```
38/38 - 0s - loss: 89370.2344 - mae: 218.4483 - mse: 89370.2344 - 69ms/epoch - 2ms/step
298.94854803962505
```

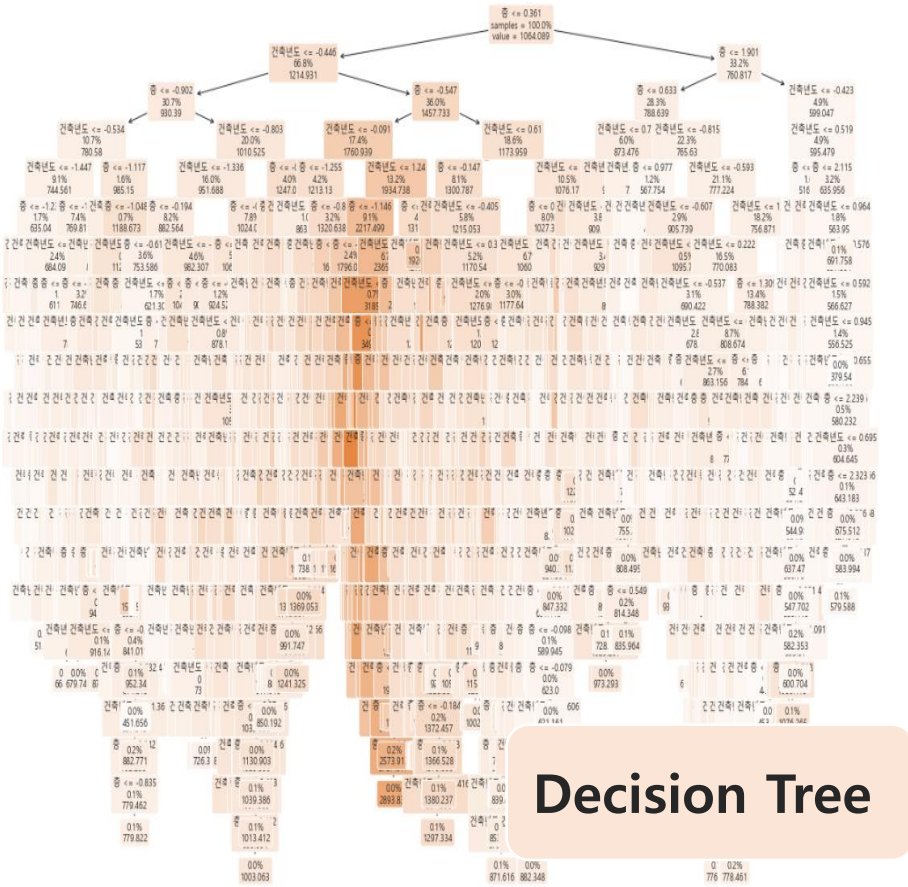
1000회 epoch결과 : RMSE 298

Machine Learning



결정적인 요인:
어떤 구에 속하는가?(강남, 서초 등등) 어떤 동인가?

지하철 역에서 거리 등 기타 요소들은 부수적인 Feature



Decision Tree

결론

KNN (위경도만 사용)

```
from sklearn.neighbors import KNeighborsRegressor

knn_feature = batch_df[['위도', '경도']]
knn_label = batch_df['환산m2당가격']

scaler = StandardScaler()
knn_feature_tr = scaler.fit_transform(knn_feature)

X_train, X_test, y_train, y_test = train_test_split(knn_feature_tr, knn_label, test_size=0.2, random_st

knn = KNeighborsRegressor(n_neighbors=25, weights = 'distance')
knn.fit(X_train,y_train)
apart_price_pred = knn.predict(X_test)

knn_reg_rmse = mean_squared_error(y_test, apart_price_pred, squared=False)
#knn_reg_rmse =np.sqrt(knn_reg_mse)

knn_reg_rmse
```

```
253.81693273513045
```

BEST

253.816

결국 가장 가격을 잘 예측하는 방법은?

- 위도, 경도 기반 Nearest Neighbor

비싼집 근처는 비싸고
싼집 근처는 싸다

426개 행정동 분석

인구비율 Feature : 인구, 인구밀도, 보육시설, 외국인주민
기초수급자, 학원갯수, 1인, 보통가구

면적비율 Feature : 면적, 상가면적, 도로비율, 임야비율
공원비율

기타 Feature : 전세가격, 중위연령(노인인구)

Output : 비슷한 행정동 끼리 Clustering

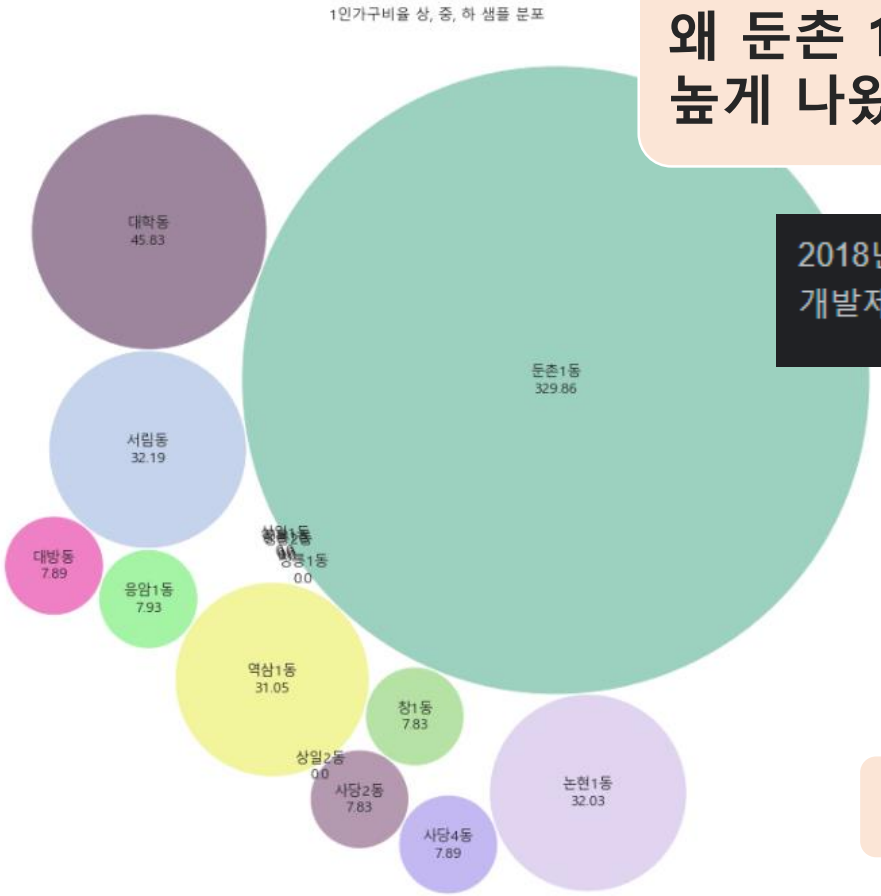
Feature 수집

공공데이터 포털

새로운 피쳐 생성

- 동별 면적
- 동별 인구밀도
- 보육시설
- 외국인 주민
- 중위 연령
- 국민기초 생활 보장 수급자
- 상가 면적(m2)
- 도로비율
- 임야 비율
- 공원 비율
- 전세 가격(20210기준)
- 학원 개수

Feature 분석



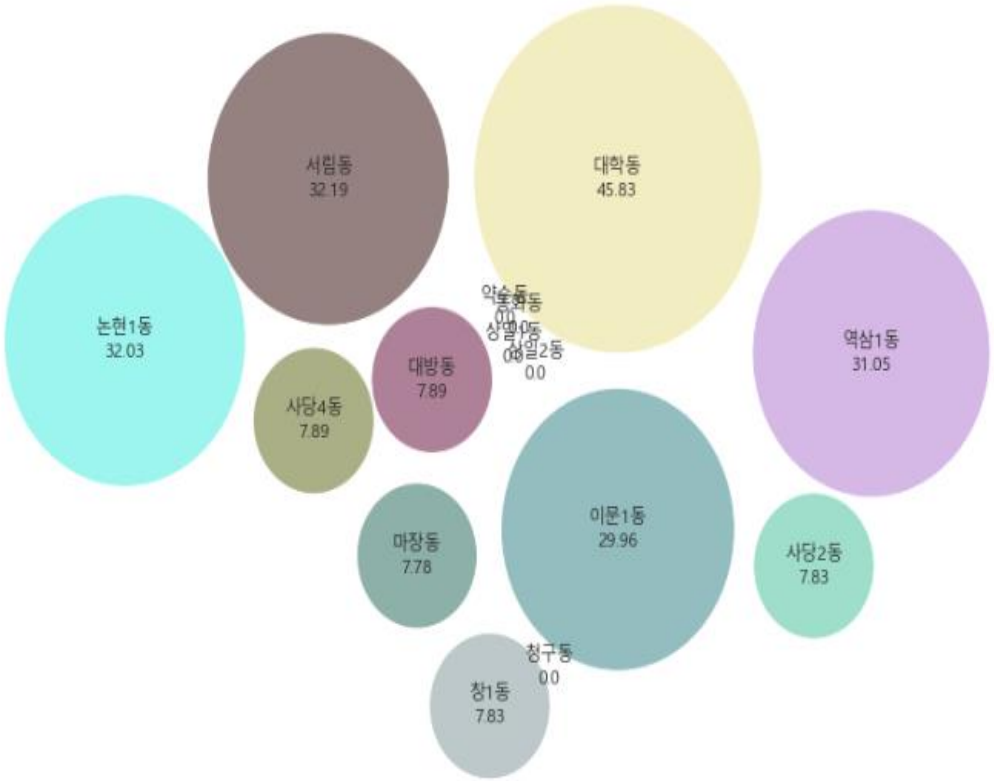
왜 둔촌 1동이 1인가구비율이 제일 높게 나왔나?

2018년 1월 19일 둔촌주공아파트의 재건축 이주가 완료되어, 둔촌1동의 거주 지역은 동부의 개발제한구역만 남게 되었고 현재 인구는 **355명**이다. Oct 24, 2022

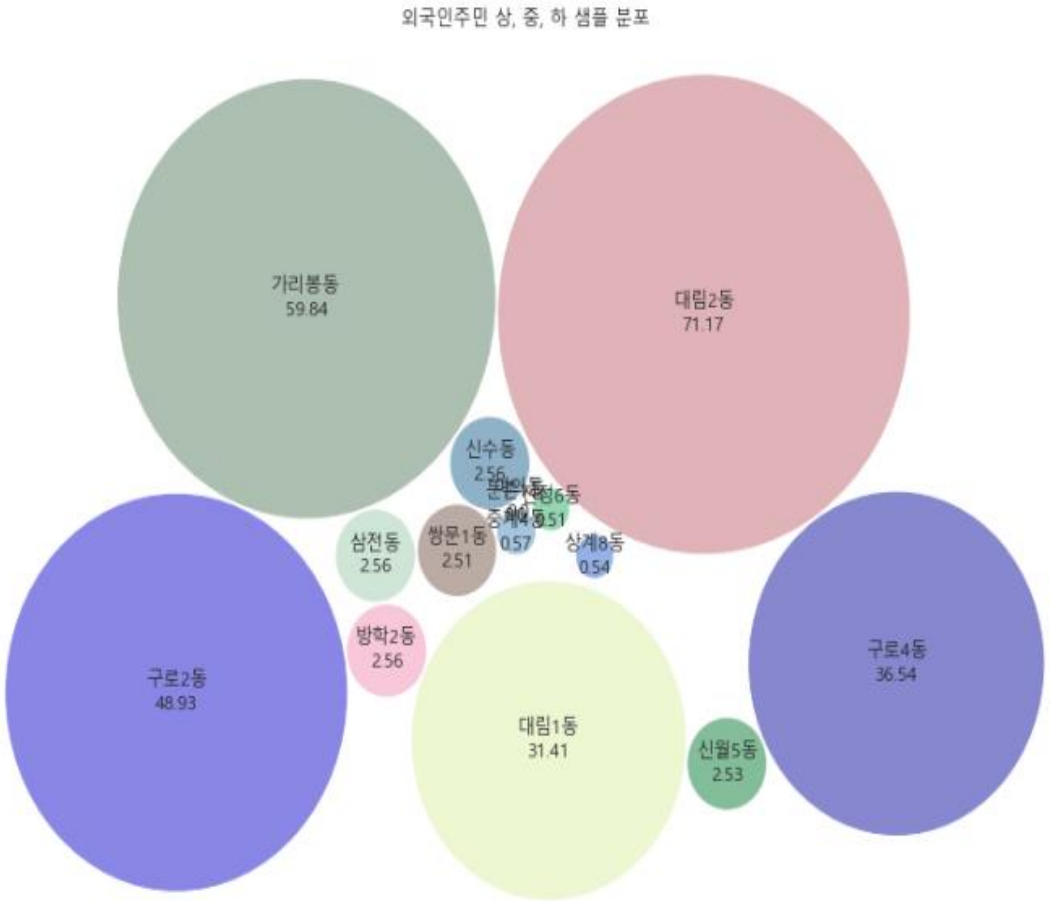
1인가구 data는 예전 기준
인구는 최근 기준
데이터의 왜곡

둔촌 1동 2017년 인구 17639명으로 update

Feature 시각화

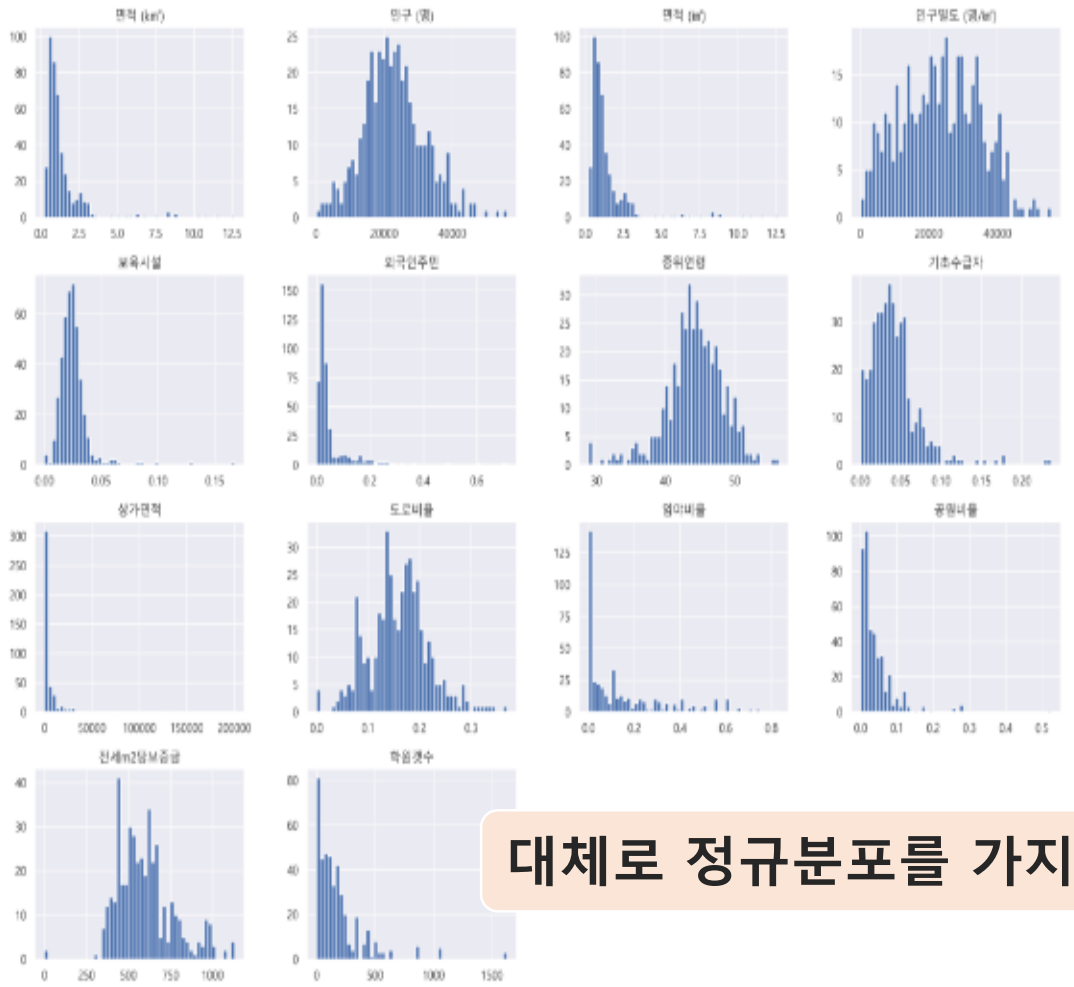


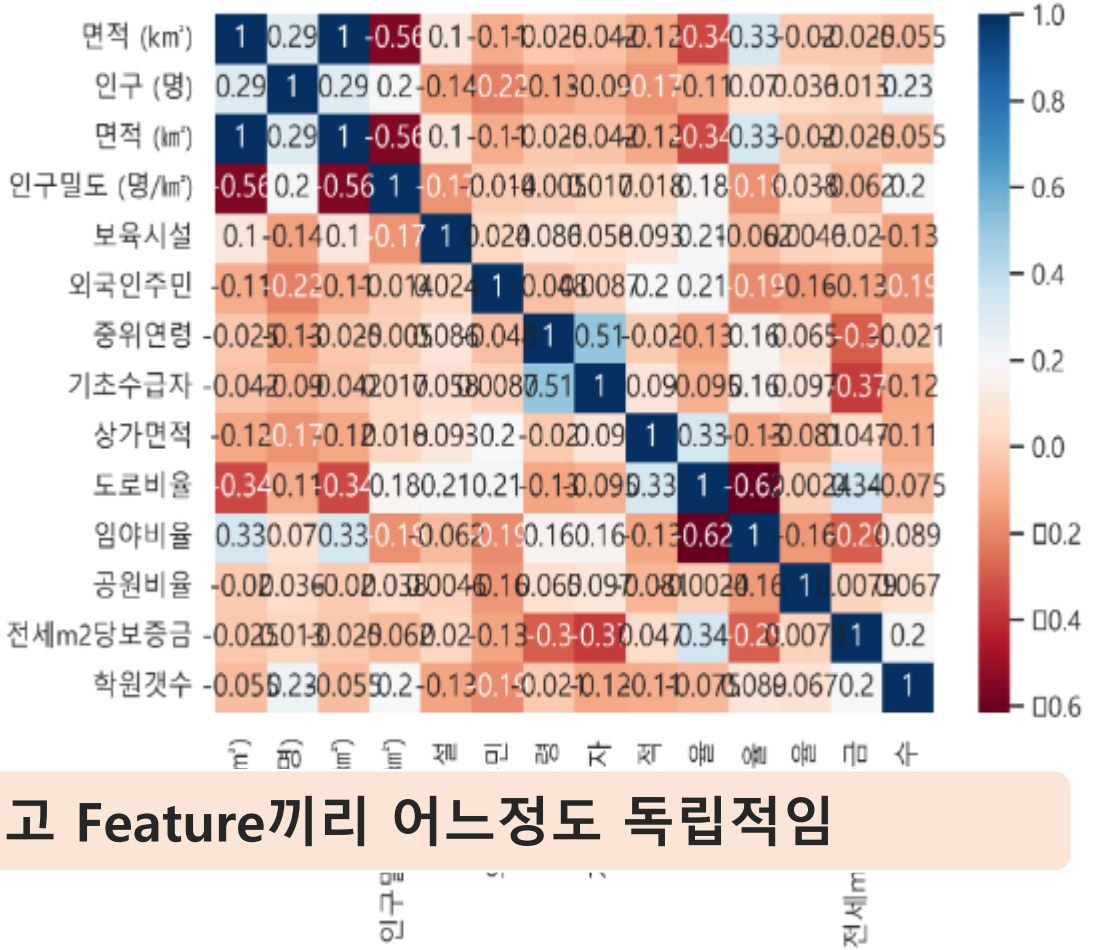
1인 가구 비율 상, 중, 하 5개씩



외국인 주민 상, 중, 하 5개씩

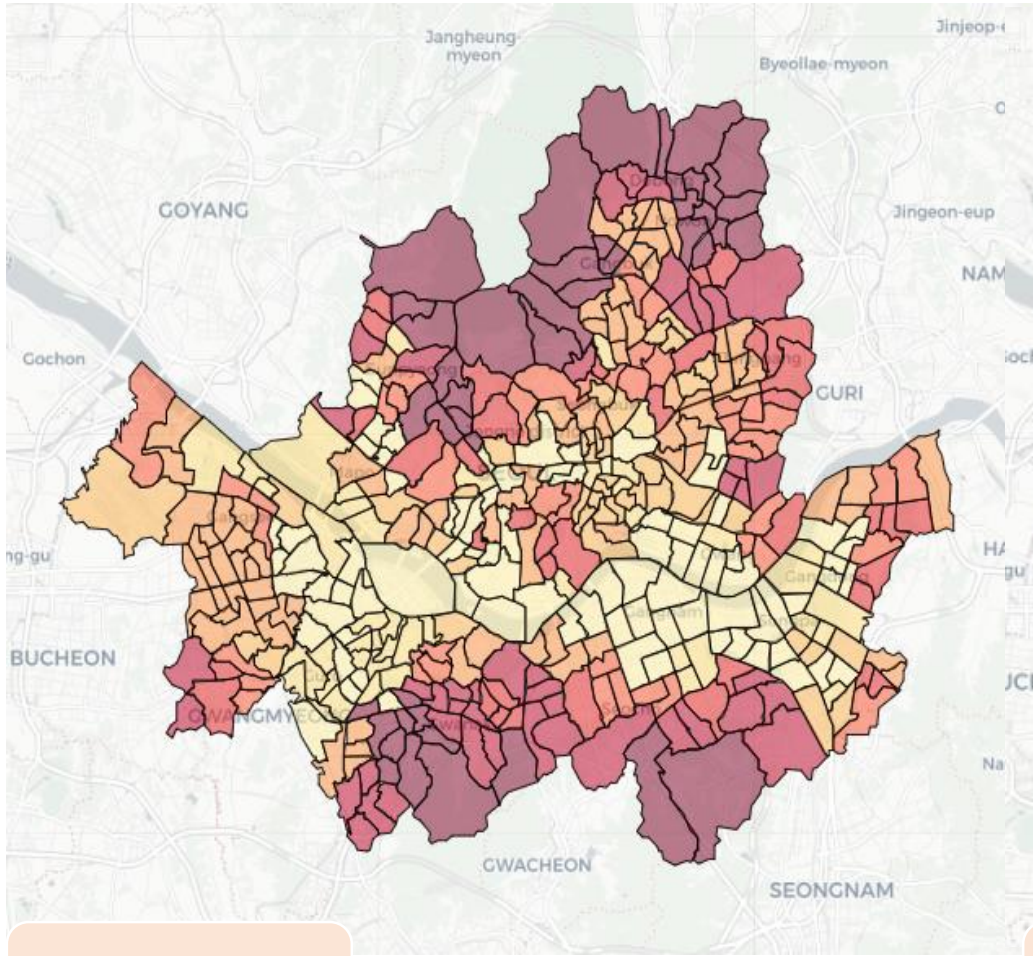
Feature 시각화



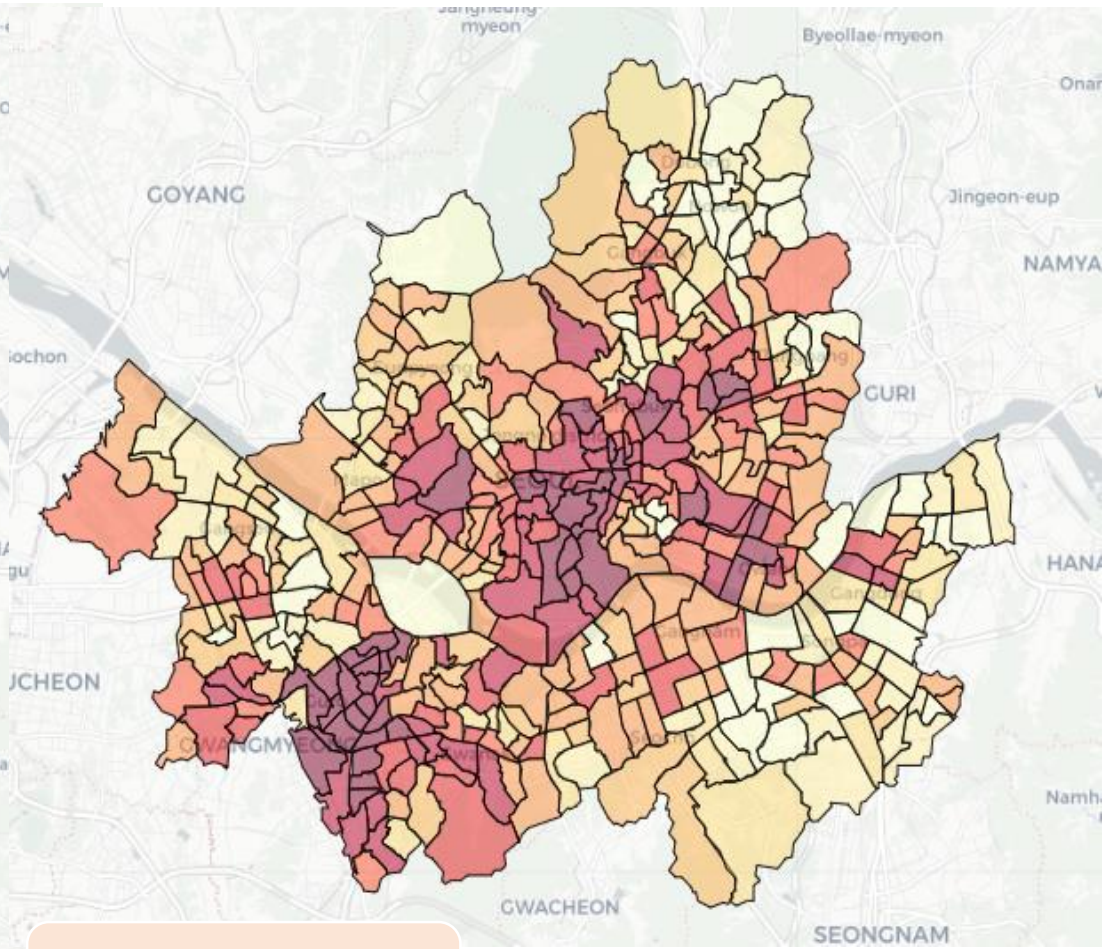


대체로 정규분포를 가지고 Feature끼리 어느정도 독립적임

Feature 시각화



임야비율



외국인주민비율

지역별 군집화

구	행정동	면적 (㎢)	인구 (명)	면적 (㎢)	인구밀도 (명/㎢)	보육시설	외국인주민	중위연령	기초수급자	상가면적	도로비율	임야비율	공원비율	전세m2당보증금	학원갯수	1인가구비율	보통가구비율
종로구	사직동	1.23	9636	1.23	7834	0.099004	0.041926	45.7	0.012972	4292.682927	0.228669	0.032864	0.021608	573.0	3.0	0.107202	0.246679
종로구	삼청동	1.49	2739	1.49	1838	0.020080	0.031763	47.9	0.016429	0.000000	0.137477	0.114735	0.000000	523.0	1.0	0.098941	0.275648
종로구	부암동	2.27	9782	2.27	4309	0.015743	0.038745	46.2	0.017788	0.000000	0.086470	0.243808	0.000061	420.0	8.0	0.070844	0.274279
종로구	평창동	8.87	18329	8.87	2066	0.009548	0.023460	47.3	0.007856	0.000000	0.031210	0.713596	0.003034	455.0	14.0	0.050085	0.260734
종로구	무악동	0.36	8297	0.36	23047	0.028926	0.011570	46.1	0.024828	0.000000	0.100584	0.456919	0.010462	787.0	22.0	0.058093	0.266843

수집된 자료를 통해
행정동 기준으로
비슷한 지역 CLUSTERING

지역별 군집화

DBSCAN으로 clustering

```
from sklearn.cluster import DBSCAN # 인싸론

dbscan =DBSCAN(eps=0.9, min_samples=3, metric='euclidean')
dbscan_labels=dbscan.fit_predict(X_features_scaled)
dong_df['dbscan_cluster'] = dbscan_labels
dong_df['dbscan_cluster'].unique()
```

DBSCAN

Spectral clustering

- 그래프 기반 클러스터링, DBSCAN이나 AGGLOMERATIVE랑 성능 비슷

```
from sklearn.cluster import SpectralClustering

sc2 = SpectralClustering(n_clusters=5, gamma=1, random_state=42)
sc2.fit(X_features_scaled)

dong_df['spectral_labels'] =  sc2.labels_
```

Spectral clustering

Agglomerative clustering (병합군집)

```
from sklearn.cluster import AgglomerativeClustering

agg = AgglomerativeClustering(n_clusters  = 8, linkage="complete").fit(X_features_sca

dong_df['agg_labels'] = agg.labels_
```

병합 군집

Bayesian Gaussian Mixture Models

```
from sklearn.mixture import BayesianGaussianMixture

bgm = BayesianGaussianMixture(n_components=10, n_init=10, random_state=42)
B_G_labels= bgm.fit_predict(X_features_scaled)
gmm_labels = gm.fit_predict(X_features_scaled)
dong_df['Bayesian_Gaussian'] = B_G_labels
```

베이지안 가우시안

Gaussian mixture clustering

```
from sklearn.mixture import GaussianMixture

gm = GaussianMixture(n_components=8, n_init=10, random_state=42)# 몇개 클러스터/ 몇번
gmm_labels = gm.fit_predict(X_features_scaled)
dong_df['gaussian_labels'] = gmm_labels

cv1_mean = np.mean(dong_df["전세m2당보증금"])    # 16.5
cv1_std = np.std(dong_df["전세m2당보증금"])     # 2.99

cv2_mean = np.mean(dong_df["중위연령"])    # 15.7
cv2_std = np.std(dong_df["중위연령"])     # 3.51

import scipy.stats as stats

cv1_pdf = stats.norm.pdf(dong_df["전세m2당보증금"].sort_values(), cv1_mean, cv1_std)
cv2_pdf = stats.norm.pdf(dong_df["중위연령"].sort_values(), cv2_mean, cv2_std)

import matplotlib.pyplot as plt
from pylab import rcParams

#plt.plot(dong_df["전세m2당보증금"].sort_values(), cv1_pdf, color="Black", label="Cult
plt.plot(dong_df["중위연령"].sort_values(), cv2_pdf, color="Orange", label="Cultivar_E
plt.legend()
plt.xlabel("Grain area (mm2)", size=15)
plt.ylabel("Frequency", size=15)
plt.grid(True, alpha=0.3, linestyle="--")
plt.rcParams["figure.figsize"] = [7,5]
plt.rcParams["figure.dpi"] = 500   # 해상
plt.show()
```

가우시안 혼합 군집

Feature 시각화

K MEANS Clustering

```
dong_df.head()
```

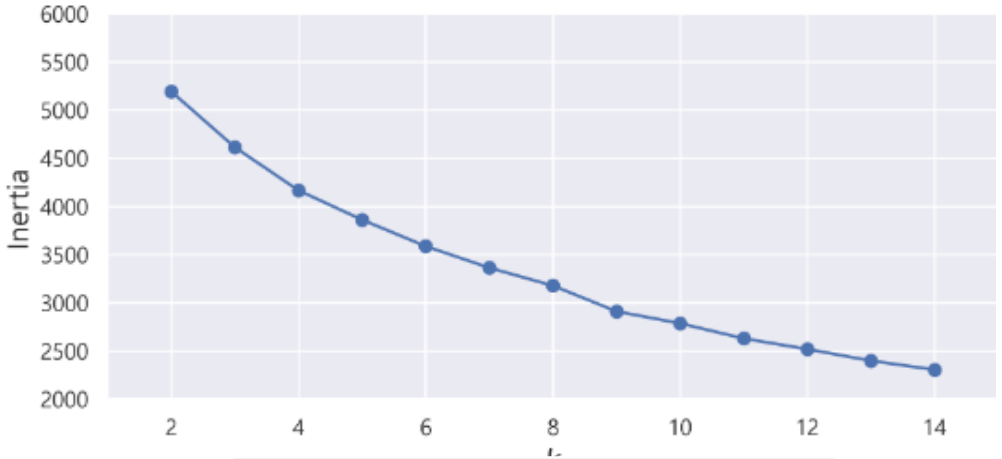
	구	행정동	면적 (㎢)	인구 (명)	면적 (㎡)	인구밀도 (명/㎡)	보육시설	외국인주
0	종로구	사직동	1.23	9636	1.23	7834	0.099004	0.041
1	종로구	삼청동	1.49	2739	1.49	1838	0.020080	0.031
2	종로구	부암동	2.27	9782	2.27	4309	0.015743	0.038
3	종로구	평창동	8.87	18329	8.87	2066	0.009548	0.023
4	종로구	무악동	0.36	8297	0.36	23047	0.028926	0.011

5 rows × 24 columns

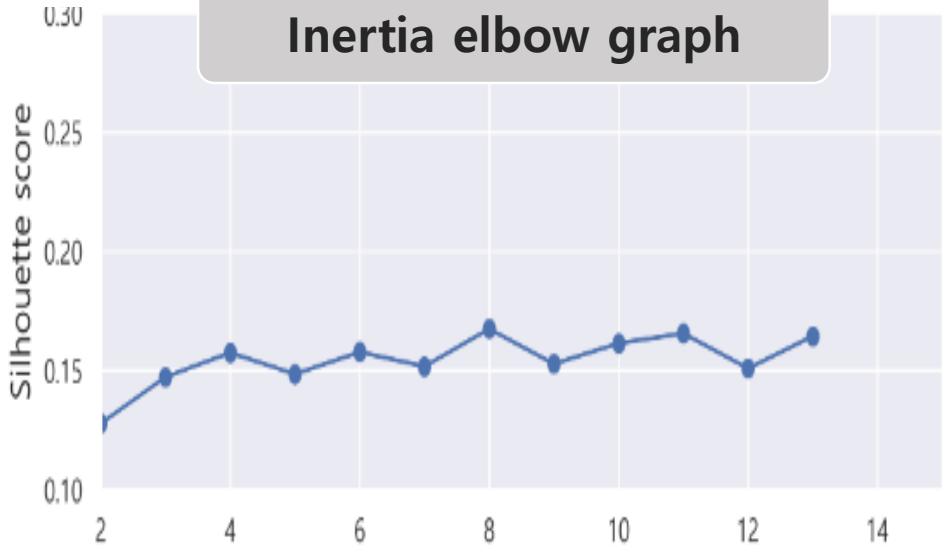
```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples

dong_k
X_feat
kmeans
labels
dong_d

print('실루엣 스코어는 : {0:.3f}'.format(silhouette_score(X_features_sca
```

실루엣 스코어는 : 0.132

전반적으로 스코어가 낮고 잘 구별되지 않음 -> 비슷한 지역이 많음

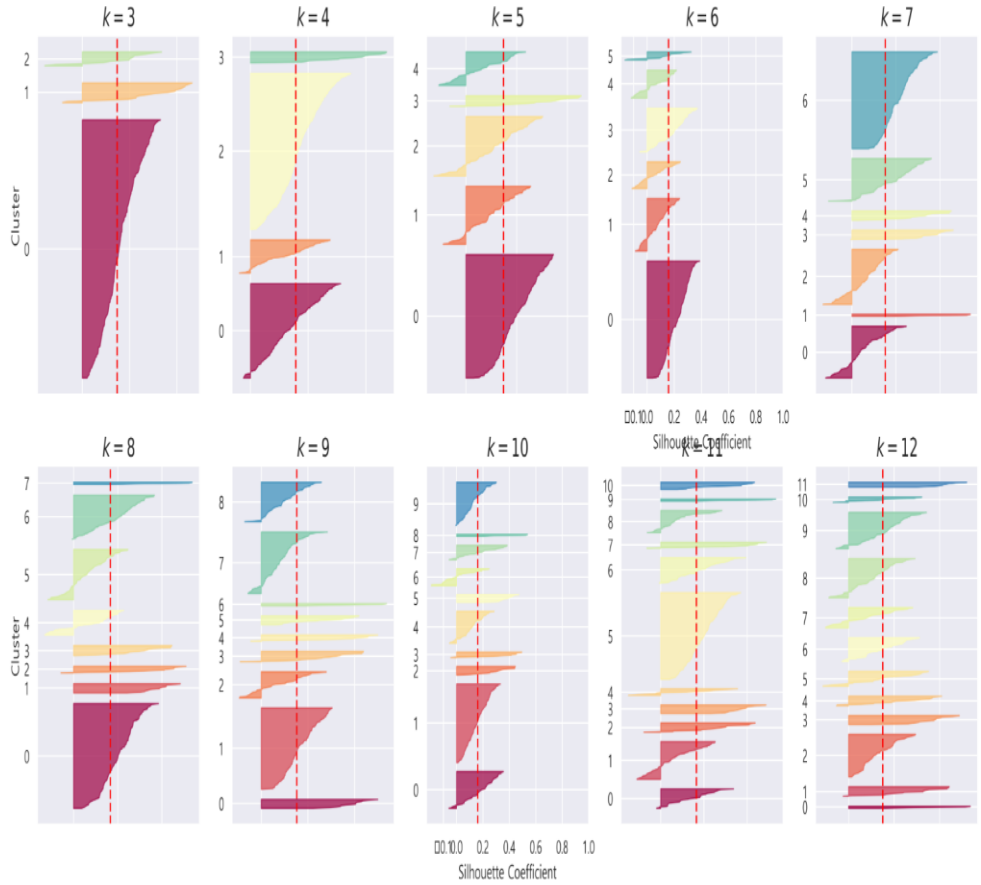


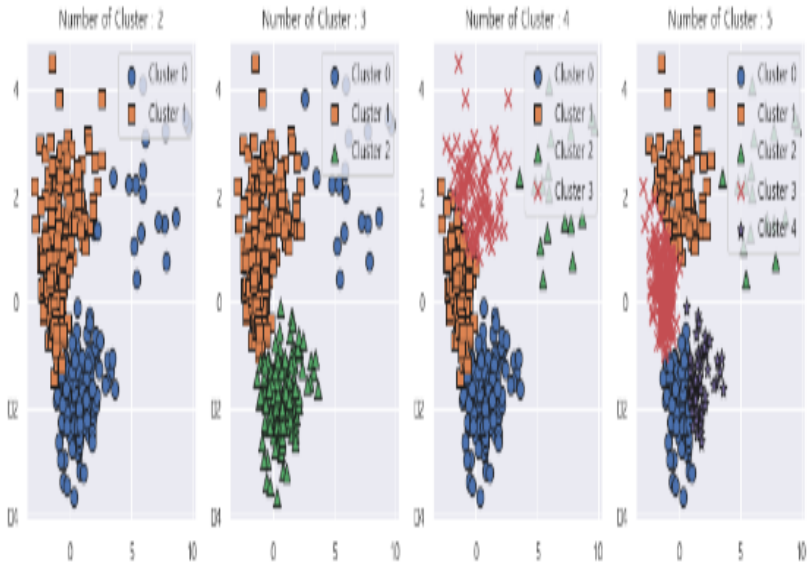
Inertia elbow graph



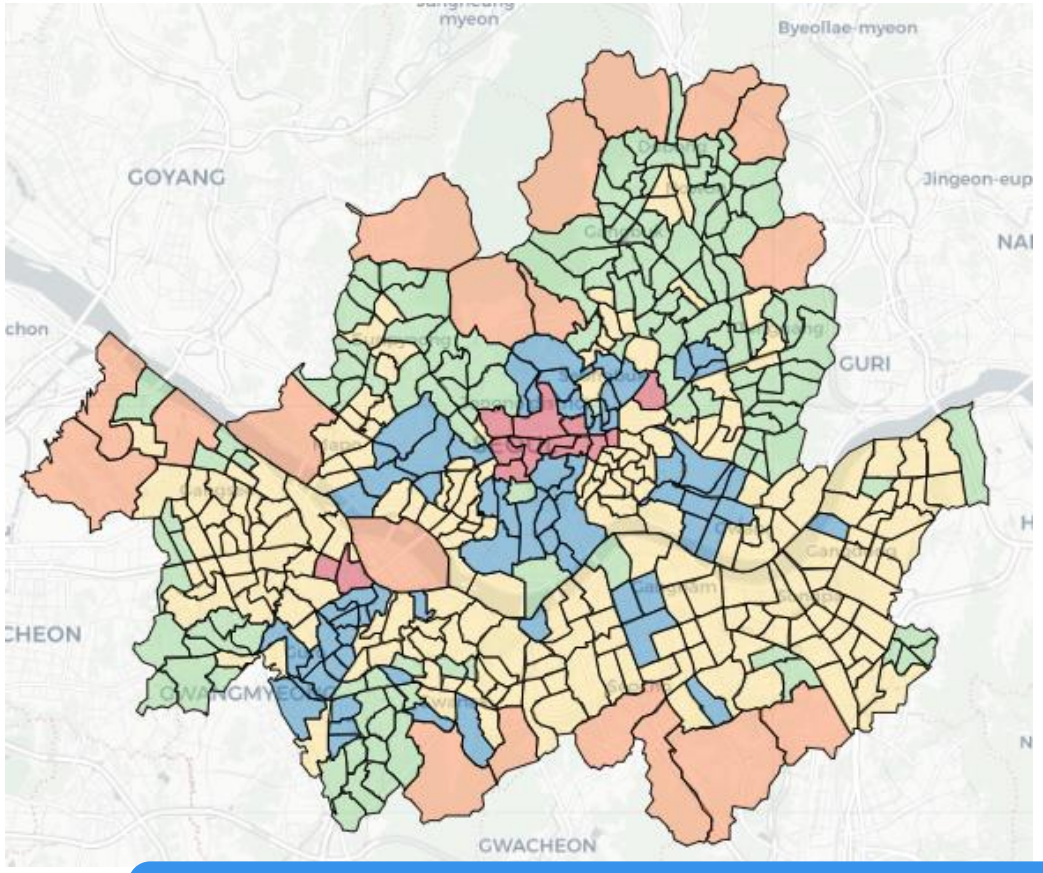
실루엣 스코어

KMEANS 시각화

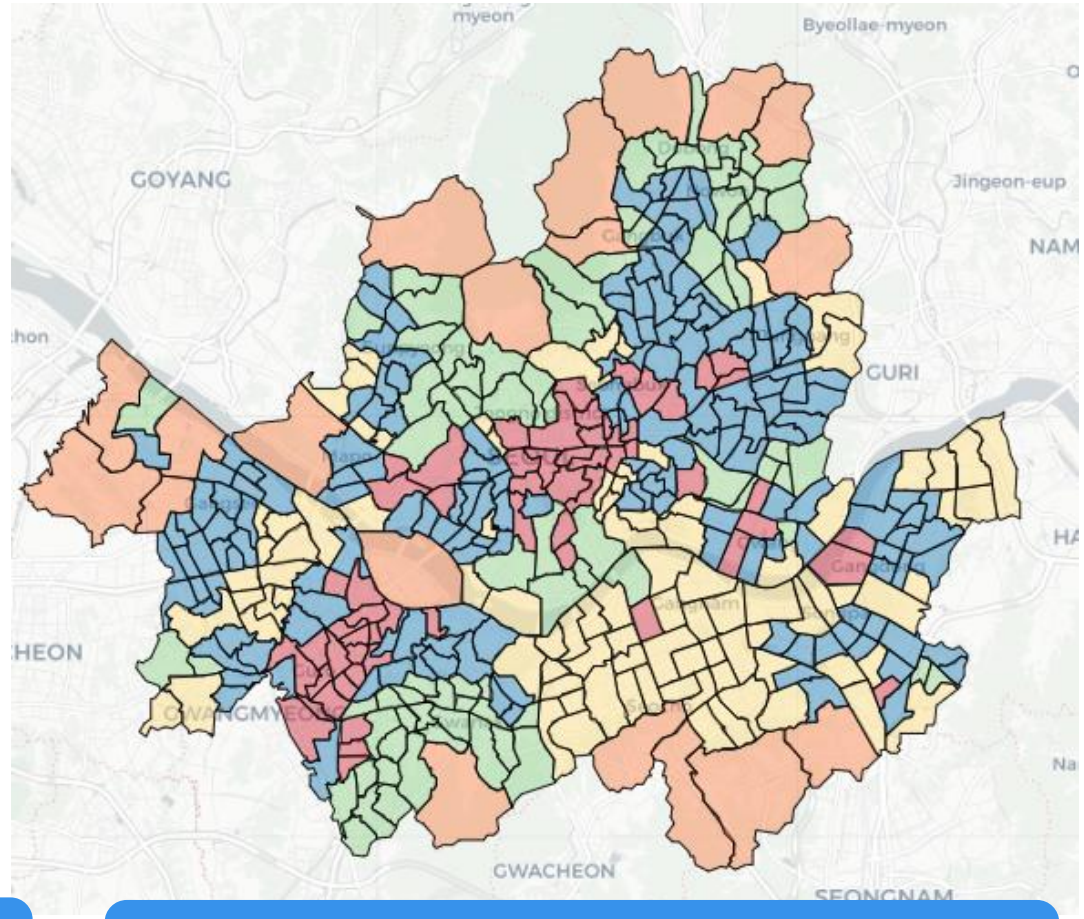

k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 9
k = 10
k = 11
k = 12
Cluster
Silhouette Coefficient


Number of Cluster : 2
Number of Cluster : 3
Number of Cluster : 4
Number of Cluster : 5
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4

Feature 시각화



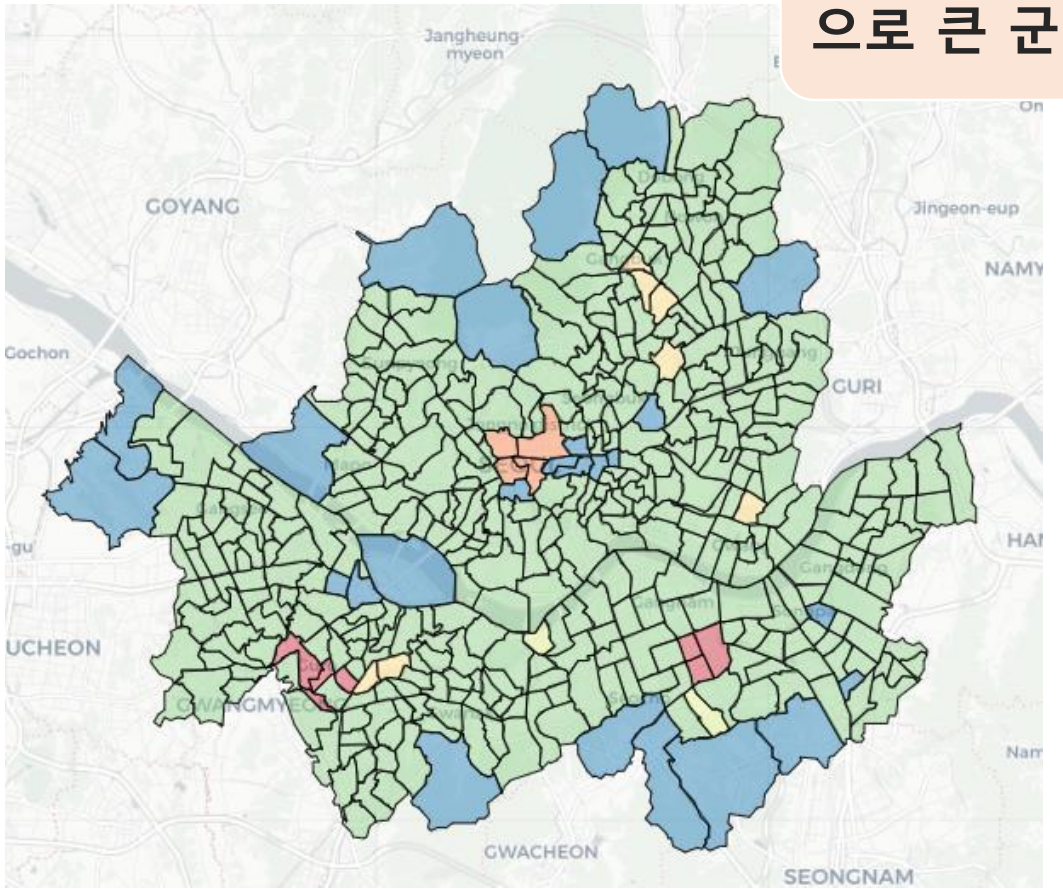
KMEANS 군집 결과

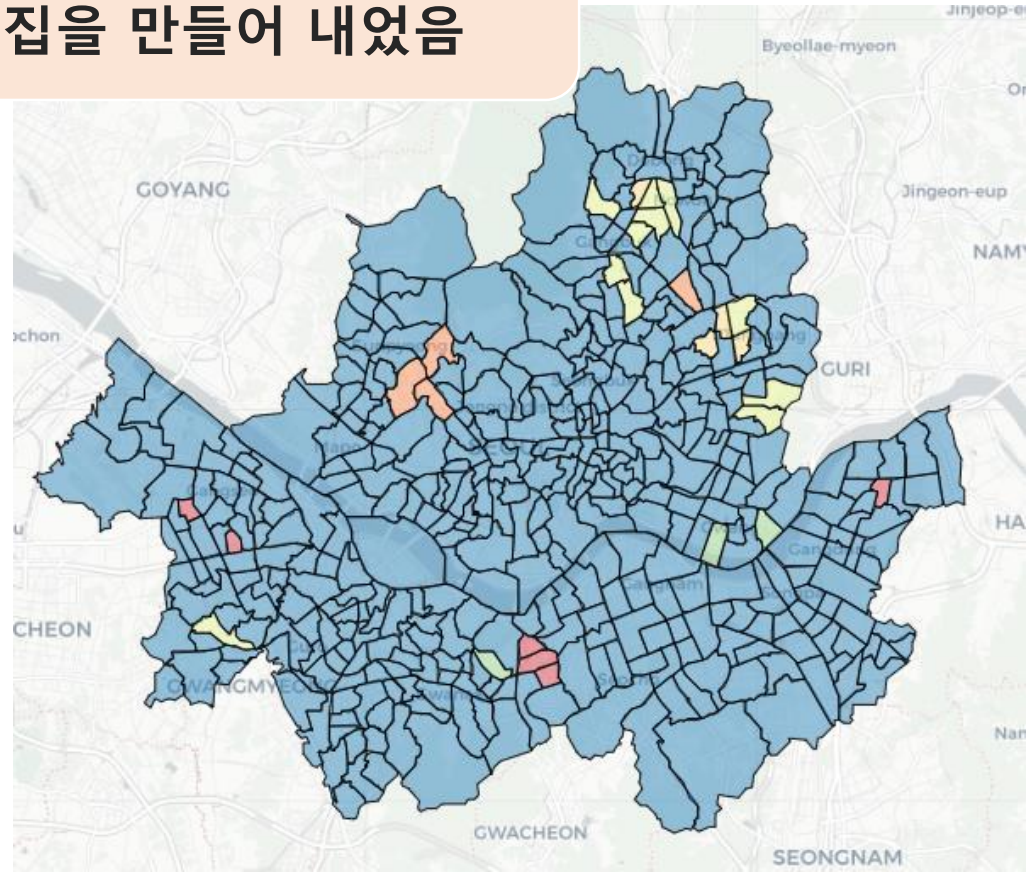


베이지언 가우시안 군집 결과

Feature 시각화

병합/그래프 기반 접근은 전반적으로 큰 군집을 만들어 내었음





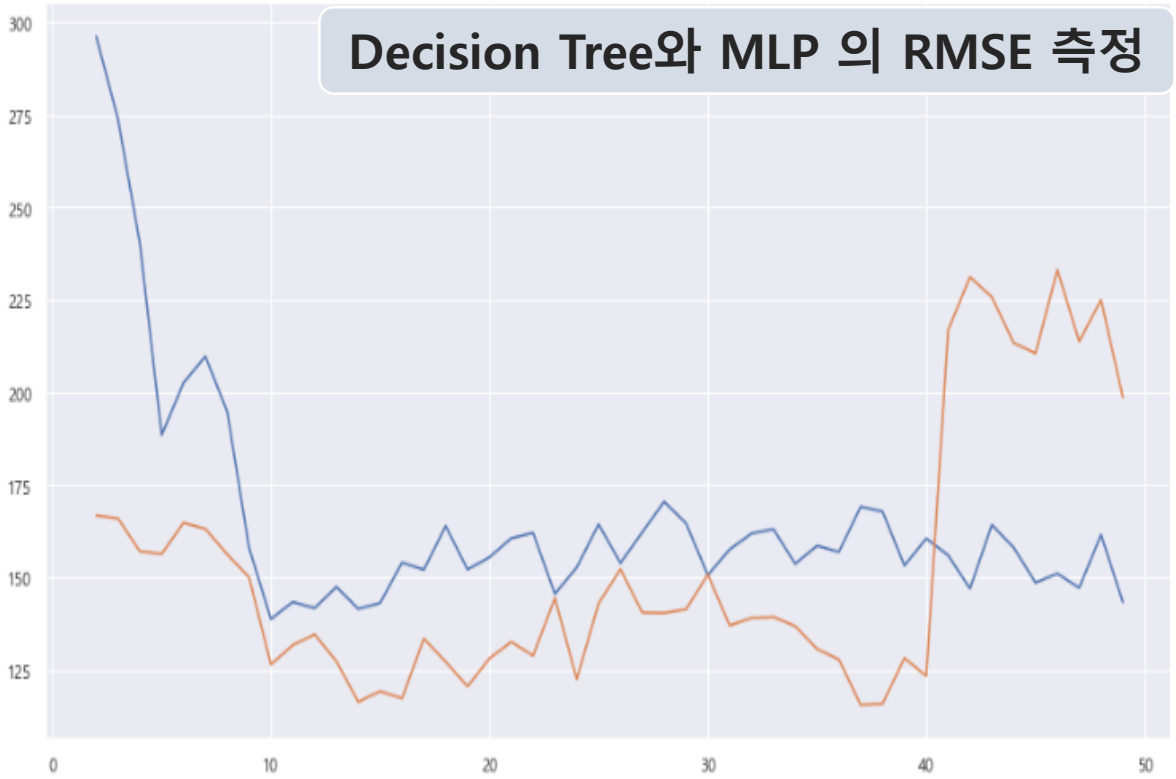
병합군집 결과

DBSCAN 군집 결과

m2당 보증금 ML with PCA

총 Feature 개수 426
(after one hot encoding)

Principal Component :2~50



tree=138.7524984

mlp=115.5200378

동별 Feature를 바탕으로 m2당 보증금을 회귀 분석

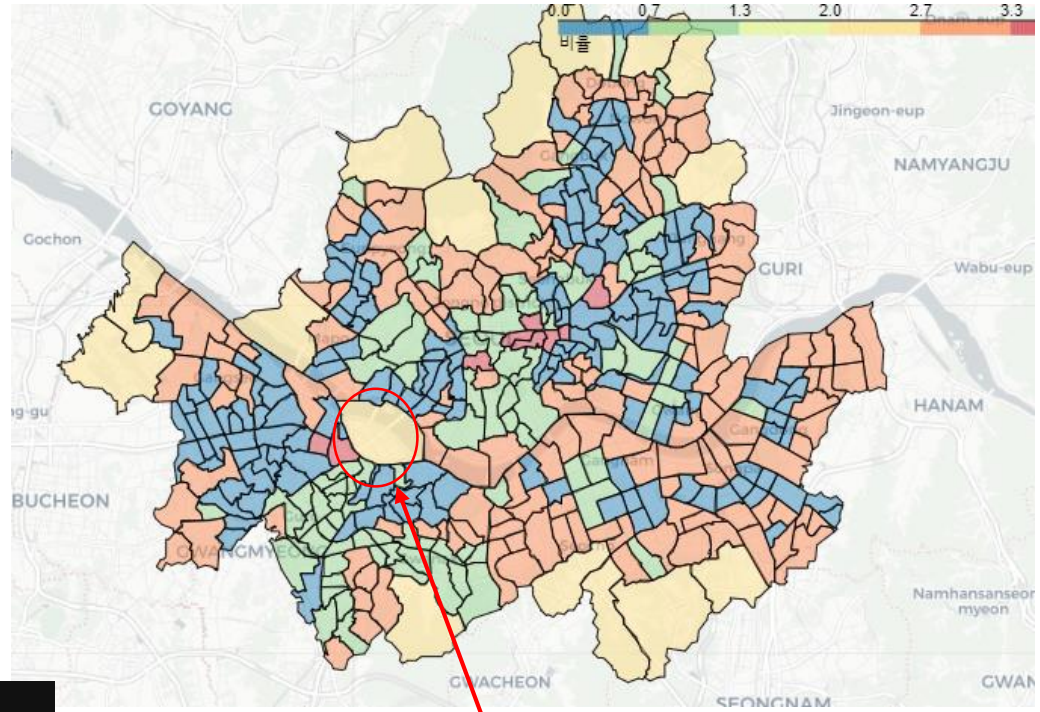
Feature의 개수가 많아진다고 성능이 높아지는 것이 아님

결론

```
pd.DataFrame(dong_df.groupby('Bayesian_Gaussian').행정동.unique())
```

Bayesian_Gaussian	행정동
0	[창신3동, 숭인1동, 신당5동, 중림동, 후암동, 원효로2동, 용문동, 청파동, ...
1	[삼청동, 부암동, 가회동, 청운효자동, 용산2가동, 서빙고동, 보광동, 한강로동,...
2	[무악동, 교남동, 다산동, 약수동, 청구동, 동화동, 효창동, 이촌1동, 이촌2동...
3	[평창동, 우이동, 인수동, 도봉1동, 공릉2동, 상계1동, 상계3.4동, 진관동,...
4	[사직동, 종로1.2.3.4가동, 종로5.6가동, 이화동, 혜화동, 창신1동, 창신...

대체로 지리상 근접한 행정동이
비슷한특징을 가지고
같은 Cluster로 묶임



	구	행정동	면적(km²)	인구(명)	면적(㎢)	인구밀도(명/㎢)	보육시설	외국인주민	중위연령	기초수급자	...	전세 m2당보증금	학원갯수	1인가구비율	보통가구비율
286	영등포구	여의동	8.4	33420	8.4	3979	0.081628	0.0	41.8	0.004039	...	662.0	115.0	0.05015	0.271095

일부 데이터에 결측 치로 인하여
비슷하지 않은 동도
같은 cluster로 묶이기도 함

상권 분석

사용한 Feature : 동별 임대료 시세, 층별 효용 비율

주변 상가 찾기 API

앞 장에서 얻어낸 동별 Feature

Output : 임대료 예측, 행정동 기준 특징 출력

주변 경쟁상가 리스트 시각화

임대료 시계열 분석, 예측

상가 조회 시스템

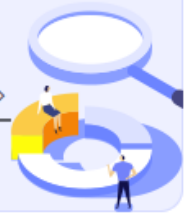
상업용부동산 임대동향조사

상업용 부동산
임대동향조사

임대정보

임대가격지수
자세히보기 >

공실률
자세히보기 >



임대료
자세히보기 >

층별효용비율
자세히보기 >

전환률
자세히보기 >

통계지도보기
임대가격지수·임대료·공실률·수익률
자세히보기 >

수익률 정보

지역별 수익별 →

상가권리금

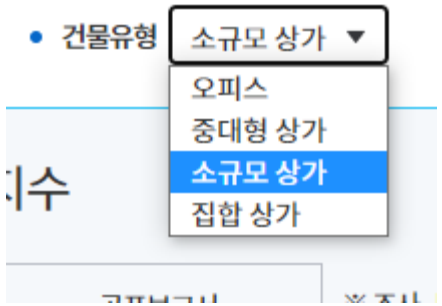
시도별/업종별
상가권리금 →

임대료 계산

동별 임대료 크롤링

행정구역	환산 임대료		
	전체	1층	1층 외
서울시 전체	123,845	140,151	108,130
종로구	172,671	207,758	137,585
청운효자동	131,133	160,242	102,023
사직동	153,017	200,198	105,836
삼청동	124,398	183,892	64,904
부암동	100,555	104,196	96,915
평창동	85,758	82,299	89,218
무악동	131,596	162,791	100,402
교남동	196,610	223,640	169,580
가회동	134,474	150,658	
종로1.2.3.4가동	185,579	249,923	
종로5.6가동	243,667	255,336	
이화동	121,320	162,176	

상가 유형별 분류



층별 효용 비율 반영

구분	'22.1Q							
	지하1층	1층	2층	3층	4층	5층	6-10층	11층이상
임대료	11.6	27.3	16.3	14.3	14.1	14.3	15.5	21.7
효용비율	42.4	100	59.9	52.4	51.8	52.5	56.9	79.7

추정 임대료 계산

```
nearby_stores[['bizesNm','i
성수1가2동
m2당 임대료:  140979 원
```

결과 시각화

응암1동
m2당 임대료: 127226 원

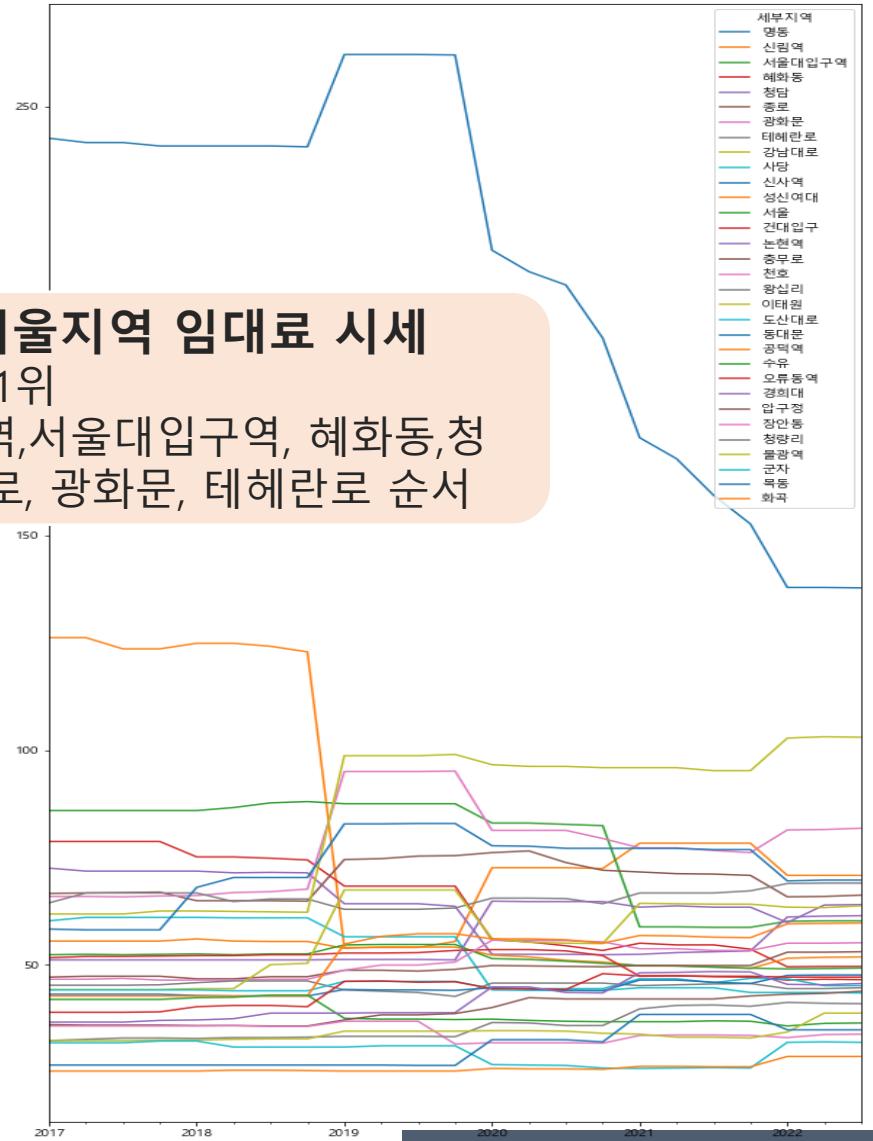
동별 분석에서 확보한 데이터로 주변 입지 분석

지도에 후보지 정보와 경쟁 상가 위치 및 정보 출력

	bizesNm	indsMclsNm
0	홍샌집	한식
0	나눌베이커리	제과제빵떡케익
0	돈치킨은평구청점	닭/오리요리
0	크린토피아응암1동점	세탁/가사서비스
0	웰빙할인마트	종합소매점
0	은평왕돈가스	양식
0	웰빙씽크	가정/주방/인테리어
0	조은환경	세탁/가사서비스
0	커피에반하다	커피점/카페
0	카프	커피점/카페
0	청담동말자싸롱	유흥주점
0	금밥	한식
0	미닝풀	커피점/카페
0	에스에이치골프	실내운동시설
0	마라하오	중식
0	라와마라탕	중식
0	데니스	커피점/카페
0	마젤필라테스	요가/단전/마사지
		분식
		/정밀기기소매

일정 반경 내 상가 및 업종 출력

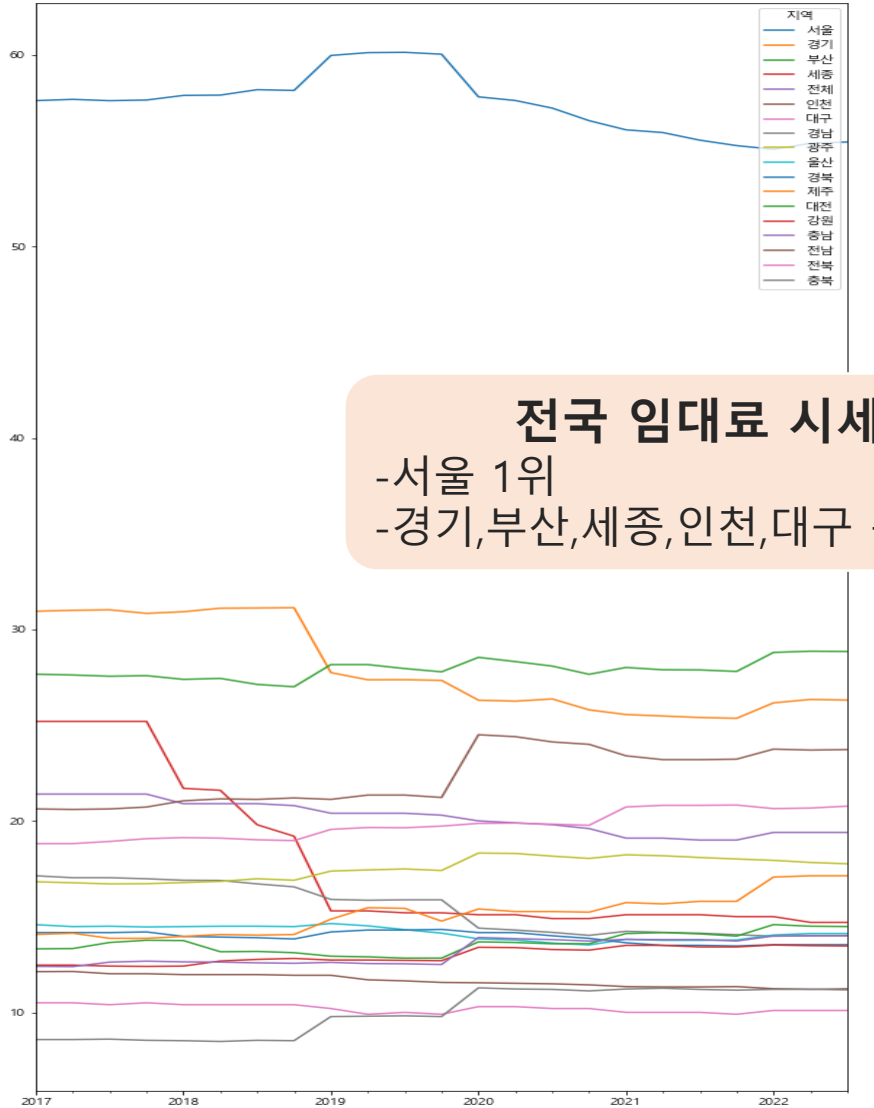
상가 임대료 시계열 그래프 (2017~2022)



서울지역 임대료 시세

-명동 1위

-신림역,서울대입구역, 혜화동,청담, 종로, 광화문, 테헤란로 순서



전국 임대료 시세

-서울 1위

-경기,부산,세종,인천,대구 순서

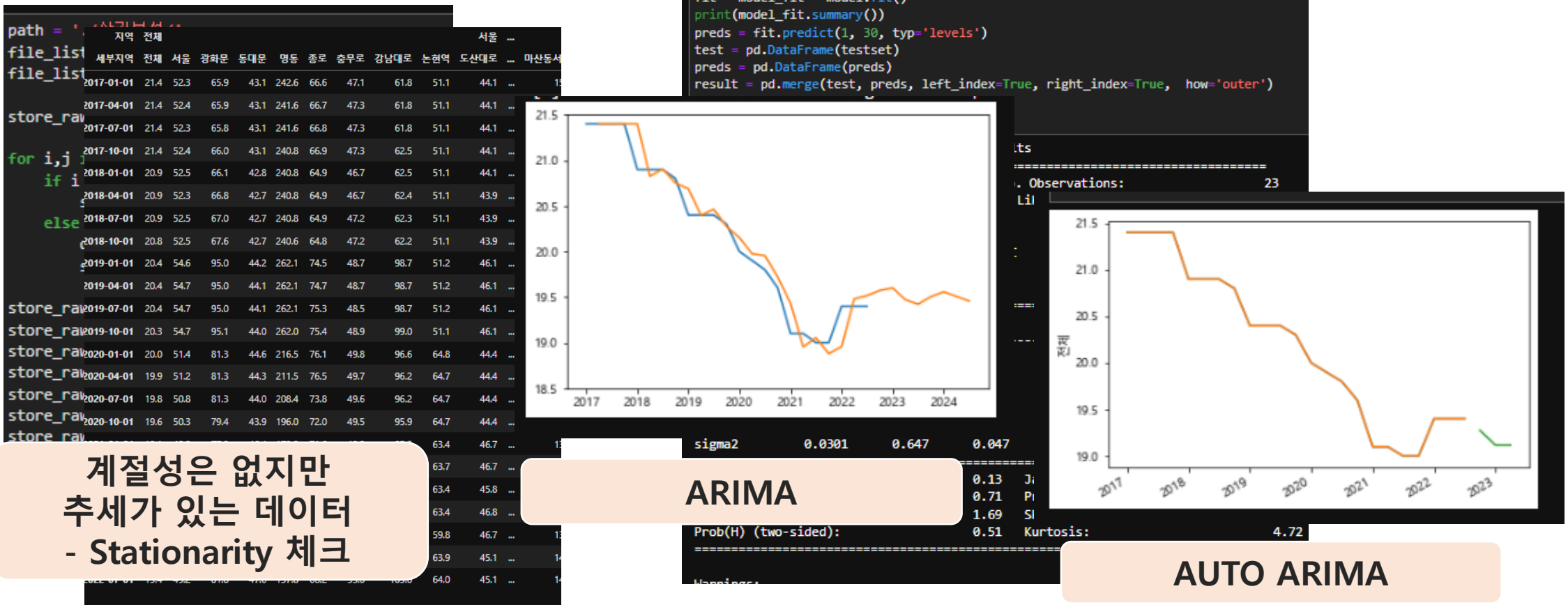
상가 임대료 시계열 예측

2017년~2022년 상가 임대료 자료 크롤링 및 병합

ARIMA를 통한 임대료 예측

국내 전체 상가 시계열

```
testset = store_raw_df['전체']['전체']
model = ARIMA(testset, order=  (3, 1, 2))
fit = model_fit = model.fit()
print(model_fit.summary())
preds = fit.predict(1, 30, typ='levels')
test = pd.DataFrame(testset)
preds = pd.DataFrame(preds)
result = pd.merge(test, preds, left_index=True, right_index=True,  how='outer')
```



계절성은 없지만 추세가 있는 데이터
- Stationarity 체크

ARIMA

AUTO ARIMA

상가 임대료 시계열 예측

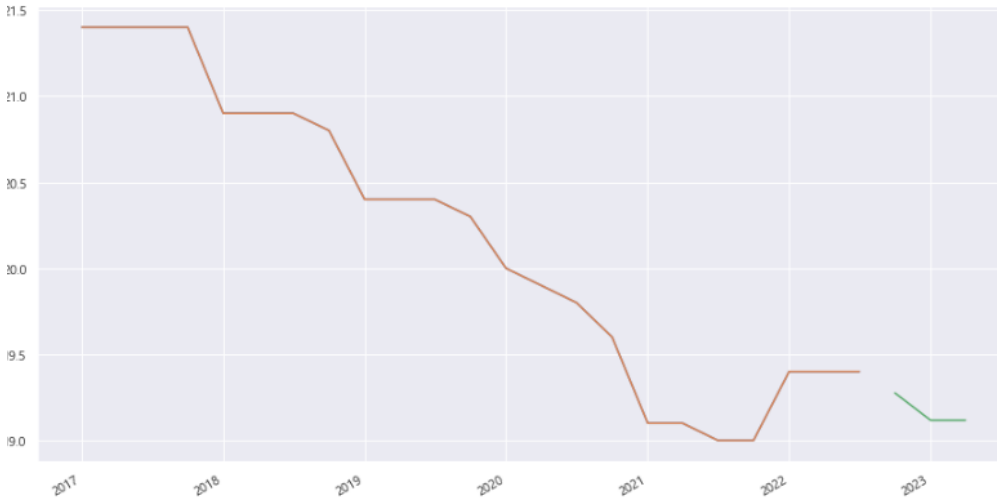
서울 지역 상가 시계열 예측

```
testset = store_raw_df['서울']['서울']
model = ARIMA(testset, order=  (3, 2, 3))
fit = model_fit = model.fit()
print(model_fit.summary())
preds = fit.predict(1, 30, typ='levels')
test = pd.DataFrame(testset)
preds = pd.DataFrame(preds)
result = pd.merge(test, preds, left_index=
plt.ylim([45, 57])
plt.plot(result);
```

동대문 지역 상가 시계열 예측

```
testset = store_raw_df['서울']['동대문']
model = ARIMA(testset, order=  (3, 2, 3))
fit = model_fit = model.fit()
print(model_fit.summary())
preds = fit.predict(1, 30, typ='levels')
test = pd.DataFrame(testset)
preds = pd.DataFrame(preds)
result = pd.merge(test, preds, left_index=True,
plt.ylim([40, 52])
plt.plot(result);
```

결론

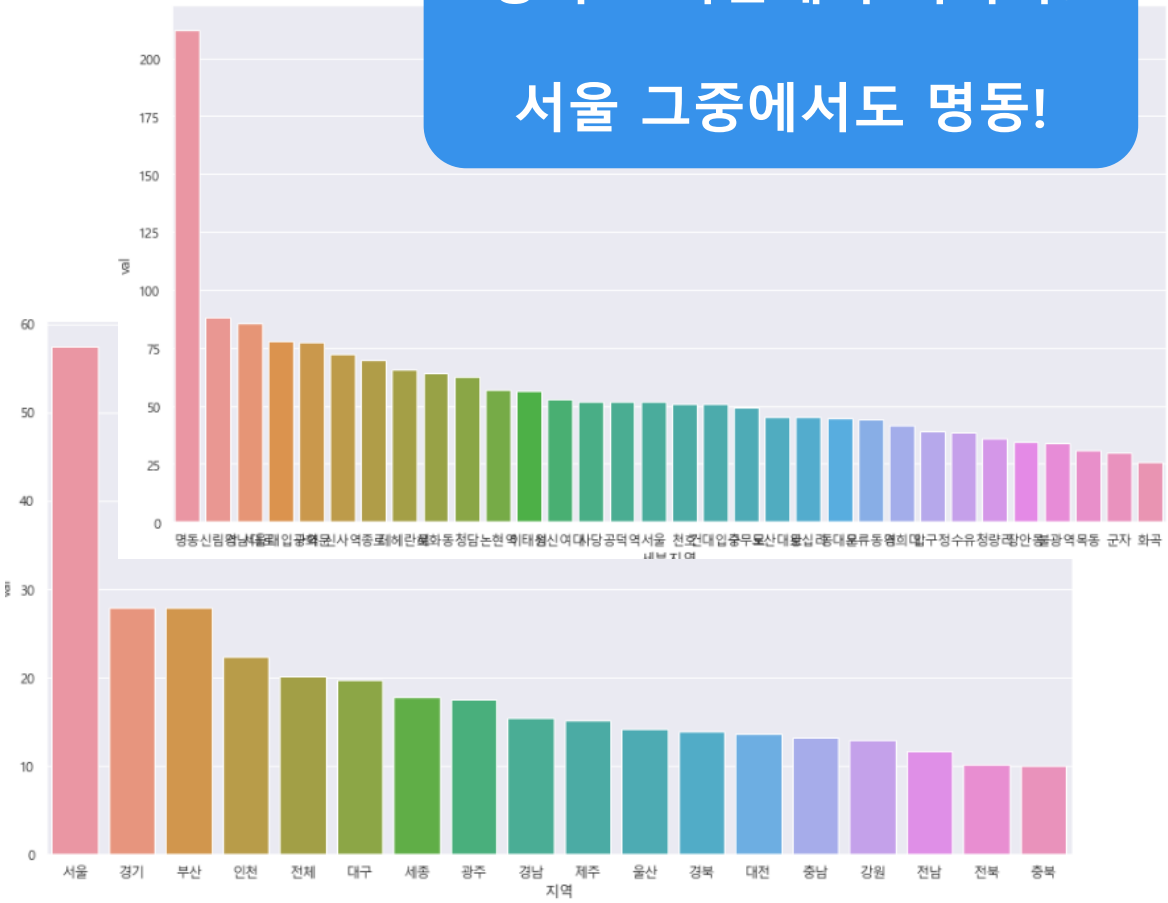


상가 임대료는 최근 5년간
상당한 내리막을 겪고 있음

**매수심리저하, 금리 인상, 경기 침체,
정부의 주택공급 등**

상가도 비싼데가 비싸다!

서울 그중에서도 명동!



전체 결론

- 영향력이 큰 Feature를 수집하고 처리하기 쉽지 않음
- 이에 맞는 알고리즘을 선정하는 것도 난이도가 있다.
- 두 가지를 조합해서 결론을 내더라도
  실제 현실에 적용하는 것은 훨씬 어려운 일이다.