

# 아파트 실거래가 예측

8조

권혁찬, 김소연, 김태한, 문정의, 이현진

# Content

---

01. 팀원 소개

02. 대회 소개

03. Data Description

04. Modeling

05. 결과

06. 경진대회 진행 소감

01

# 팀원 소개

# 팀원 소개

이름	김소현	권혁찬	김태한	문정의	이현진
역할	조장	조원	조원	조원	조원
전공	컴퓨터공학과 (석사)	선박기관시스템 공학과	응용통계학 인공지능 (석사)	전자계산학과	산업경영공학과
관심사	Computer Vision, NLP, 최적화, 경량화	기계 시스템 고장진단, 시계열 예측, LLM, MLops	Computer Vision, NLP, 추천 시스템 OTT, 문화산업 (책, 웹툰, 영화 등등), 쇼핑 등	ML, 강화학습	데이터 분석, LLM, NLP

02

# 대회 소개

대회명 : 아파트 실거래가 예측

기 간 : 2024.01.15 ~ 2024.01.25 19:00

내 용 : 서울시 아파트 가격을 예측하기 위해, 아파트의 기본 요소와 (위치, 단지 규모, 주거면적, 주차대수, 건축연도, 연도별 매매가격 등등) 관련된 요소들을 (교통 편의성, 편의시설, 학교, 공원 등등) 활용하여 최적의 아파트 가격을 예측하는 모델을 구축함

예측방법 : Regression

평가지표 : RMSE (Root Mean Squared Error)

평가분석 : RMSE값이 작을수록 더 좋은 예측 성능을 가짐

## 제출 파일

---

파일명 : submission.csv

내 용 : 학습이 완료된 모델을 통해서 예측된 9,272개의 아파트 예측 가격



03

# Data Description

## 1. Dataset 목록

Dataset	설명	Rows	Columns	비고
Train.csv	<ul style="list-style-type: none"><li>- 아파트 거래 데이터</li><li>- 주소, 아파트명, 면적, 건축년도 등 (Target Feature 포함)</li></ul>	1,118,822	52	
Test.csv	<ul style="list-style-type: none"><li>- 아파트 거래 데이터</li><li>- 주소, 아파트명, 면적, 건축년도 등 (Target Feature 제외)</li></ul>	9,272	51	
Subway_feature.csv	<ul style="list-style-type: none"><li>- 서울시 지하철 승강장의 주소, 위도, 경도 데이터</li></ul>	768	5	
Bus_feature.csv	<ul style="list-style-type: none"><li>- 서울시 버스 승강장의 주소, 위도, 경도 데이터</li></ul>	12,584	6	
Address_xy.csv	<ul style="list-style-type: none"><li>- 번지 주소 기준 위도, 경도 데이터</li></ul>	8,953	3	카카오 API 활용
Address_subway_bus.csv	<ul style="list-style-type: none"><li>- 번지 주소 기준 지하철, 버스 정류장 건수</li></ul>	8,953	5	
Apartment_region_index.csv	<ul style="list-style-type: none"><li>- 아파트 매매 실거래 가격 지수 (시군구)</li></ul>	1,775	7	
Apartment_Scale_index.csv	<ul style="list-style-type: none"><li>- 아파트 규모별 매매 실거래 가격 지수 (면적별)</li></ul>	1,070	7	
Bank_interest_index.csv	<ul style="list-style-type: none"><li>- 한국은행 기준금리 자료</li></ul>	204	3	
국토교통부_표준지공시지가.csv	<ul style="list-style-type: none"><li>- 주소지별 표준지 공시지가 자료</li></ul>	560,000	5	필요 Feature 추출

2. Dataset 상세내역

2.1 Train, Test

구분	Feature	비고
기본 정보	시군구, 번지, 본번, 부번, 아파트명, 층, 도로명, 전용면적(m <sup>2</sup> )	
계약 정보	Target(매매금액), 계약년월, 계약일, 등기신청일자, 거래유형, 중개사소재지, 기타/의무/임대/임의	
아파트 정보	건축년도, k-단지분류(아파트,주상복합등등), k-세대타입(분양형태), k-관리방식, k-복도유형, k-난방방식, k-전체동수, k-전체세대수, k-건설사(시공사), k-시행사, k-사용검사일-사용승인일, k-연면적, k-주거전용면적, k-관리비부과면적, k-전용면적별세대현황(60m <sup>2</sup> 이하), k-전용면적별세대현황(60m <sup>2</sup> ~85m <sup>2</sup> 이하), k-85m <sup>2</sup> ~135m <sup>2</sup> 이하, k-135m <sup>2</sup> 초과, 단지승인일, 단지신청일, 건축면적, 주차대수	
기타 정보	k-전화번호, k-팩스번호, k-팩스번호, k-홈페이지, k-등록일자, k-수정일자, 고용보험관리번호, 경비비관리형태, 세대전기계약방법, 청소비관리형태, 사용허가여부, 관리비 업로드, 좌표X, 좌표Y	

2.2 Subway\_feature (768건)

구분	설명	Null	비고
역사_ID	역사 고유 ID	0	
역사명	역사명	0	
호선	지하철 호선	0	
위도	위도	0	
경도	경도	0	

2.3 Bus\_feature (12,584건)

구분	설명	Null	비고
노드 ID	정류소 고유 ID	0	
정류소 번호	정류소 번호	0	
정류소명	정류소 명칭	0	
X좌표	위도	0	
Y좌표	경도	0	
정류소 타입	차로구분 정보 (일반차로, 마을버스, 중앙차로, 가로변시간, 가로변전일, 가상정류장)	0	

2.4 Address\_xy (8,953건)

구분	설명	Null	비고
Addr	번지 주소 정보	0	
x	위도	141	
y	경도	141	

2.5 Address\_subway\_bus (8,953건)

구분	설명	Null	비고
Addr	번지 주소 정보	0	
x	위도	141	
y	경도	141	
Subways	현재 주소 근처의 지하철 갯수 (1Km 이내)	0	
Bus_stops	현재 주소 근처의 버스 정류장 갯수 (3Km 이내)	0	

2.6 Apartment\_region\_index (1,775건)

구분	설명	Null	비고
index	시점 그룹핑 정보 (0 ~ 70)	0	
시	시 정보	0	
구	구 정보	0	
index_region	구 기준 실거래 가격 지수	0	
시점	년과 분기	0	
분기	분기	0	

2.7 Apartment\_Scale\_index (1,070건)

구분	설명	Null	비고
index	시점 그룹핑 정보 (0 ~ 70)	0	
시	시 정보	0	
주거면적	면적 정보	0	
index_region	면적 기준 실거래 가격 지수	0	
시점	년과 월 (2006.01)	0	
계약년월	년과 월 (200601)	0	

2.8 Bank\_interest\_index (204건)

구분	설명	Null	비고
날짜	년과 월 (2007-01)	0	
한국은행기준금리	한국은행 기준금리 (4.5)	0	
정부대출금리	정부대출 금리 (4.75)	0	

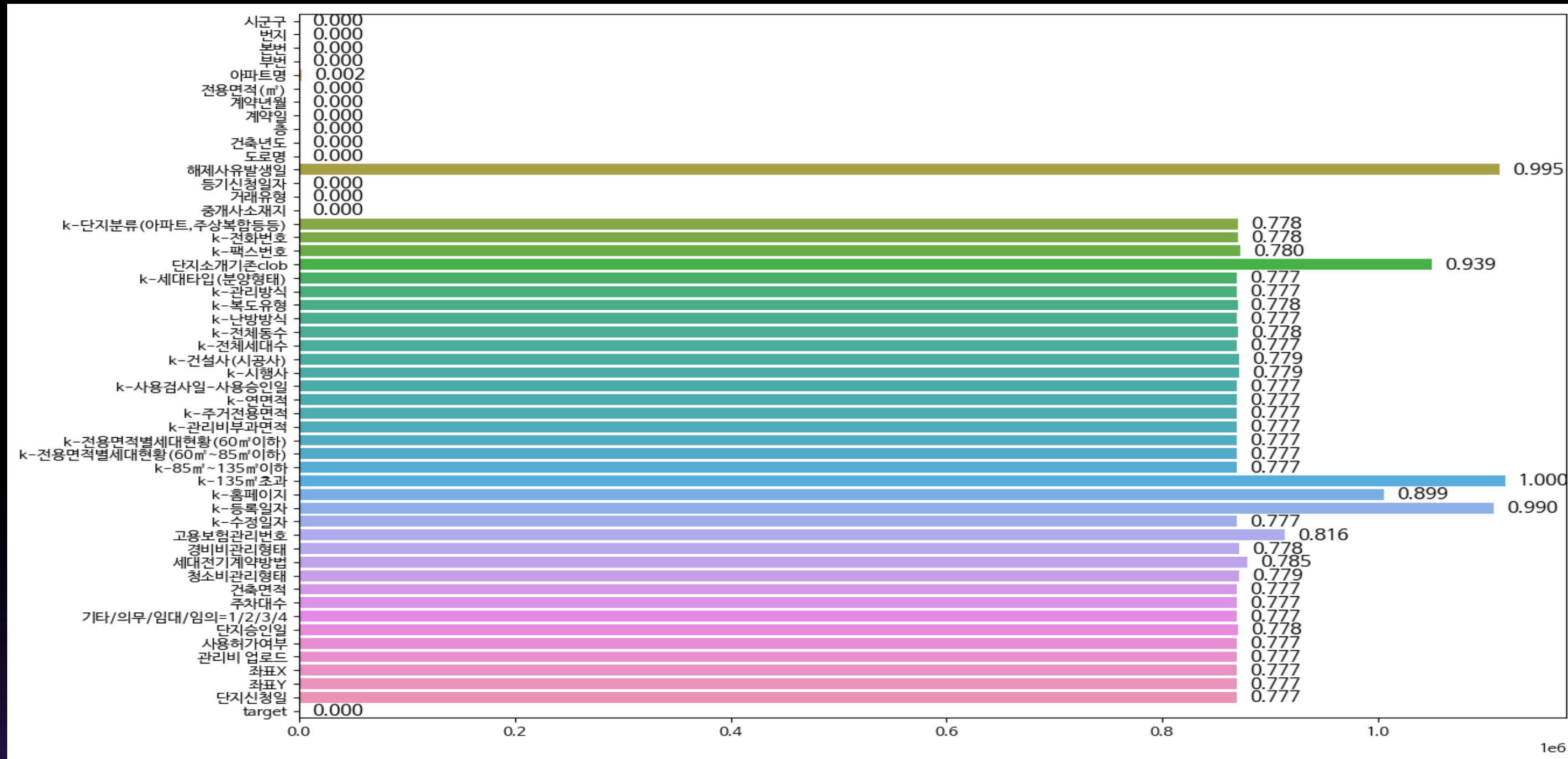
2.9 국토교통부\_표준지공시지가 (560,000건)

구분	설명	Null	비고
시도명	시도명 (서울특별시)	0	
시군구명	시군구명 (종로구)	0	
소재지	동이하 주소 (청운동 3-52)	2,594	
공시지가	공시지가 (456800)	0	
전년지가	년과 분기 (493500.0)	0	



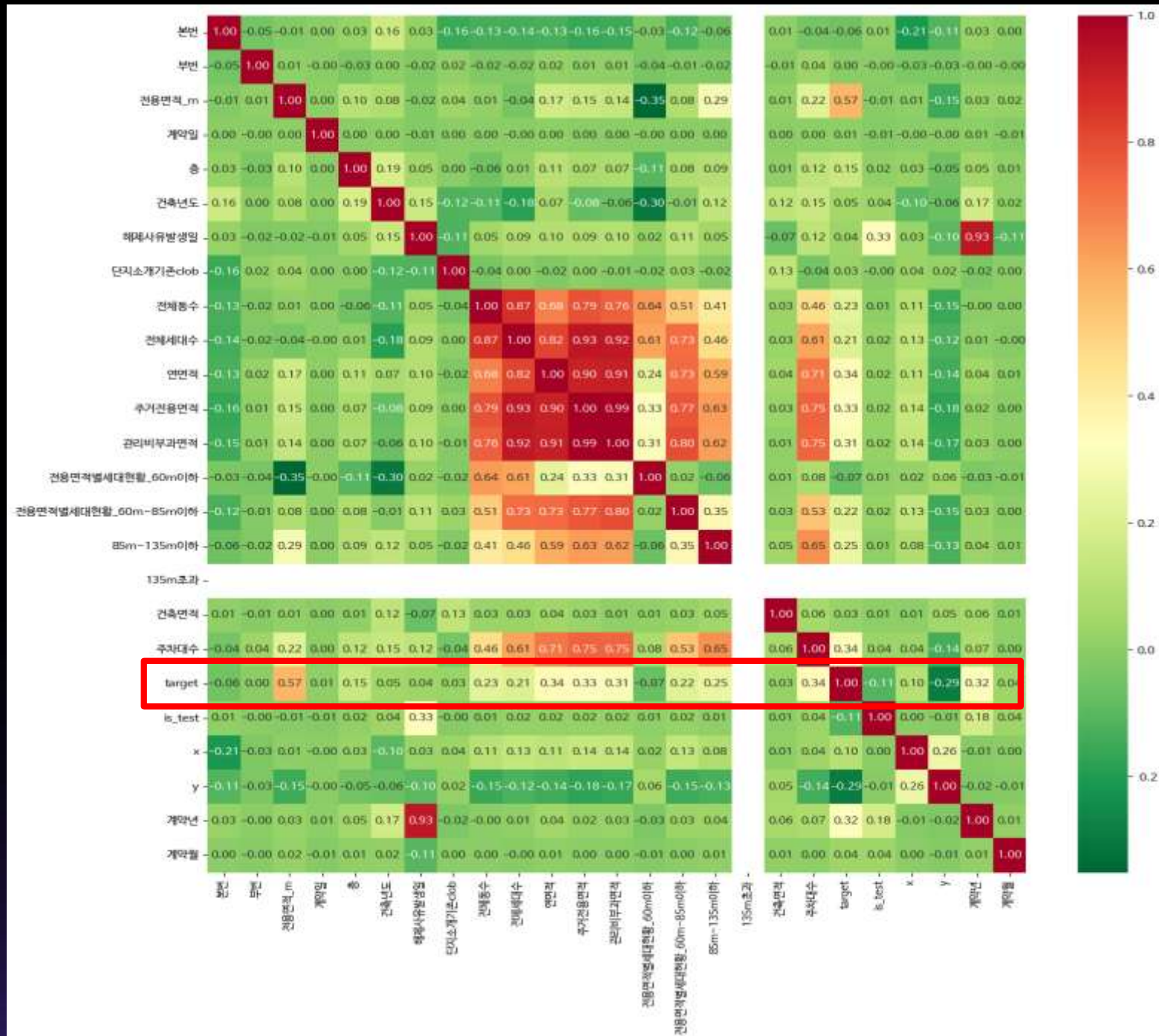
# 데이터의 기초 통계 및 정보 요약

- 연속형 변수 : 23개, 범주형 변수 : 29개
- train/test의 column 52개중 40개가 77% 이상의 결측



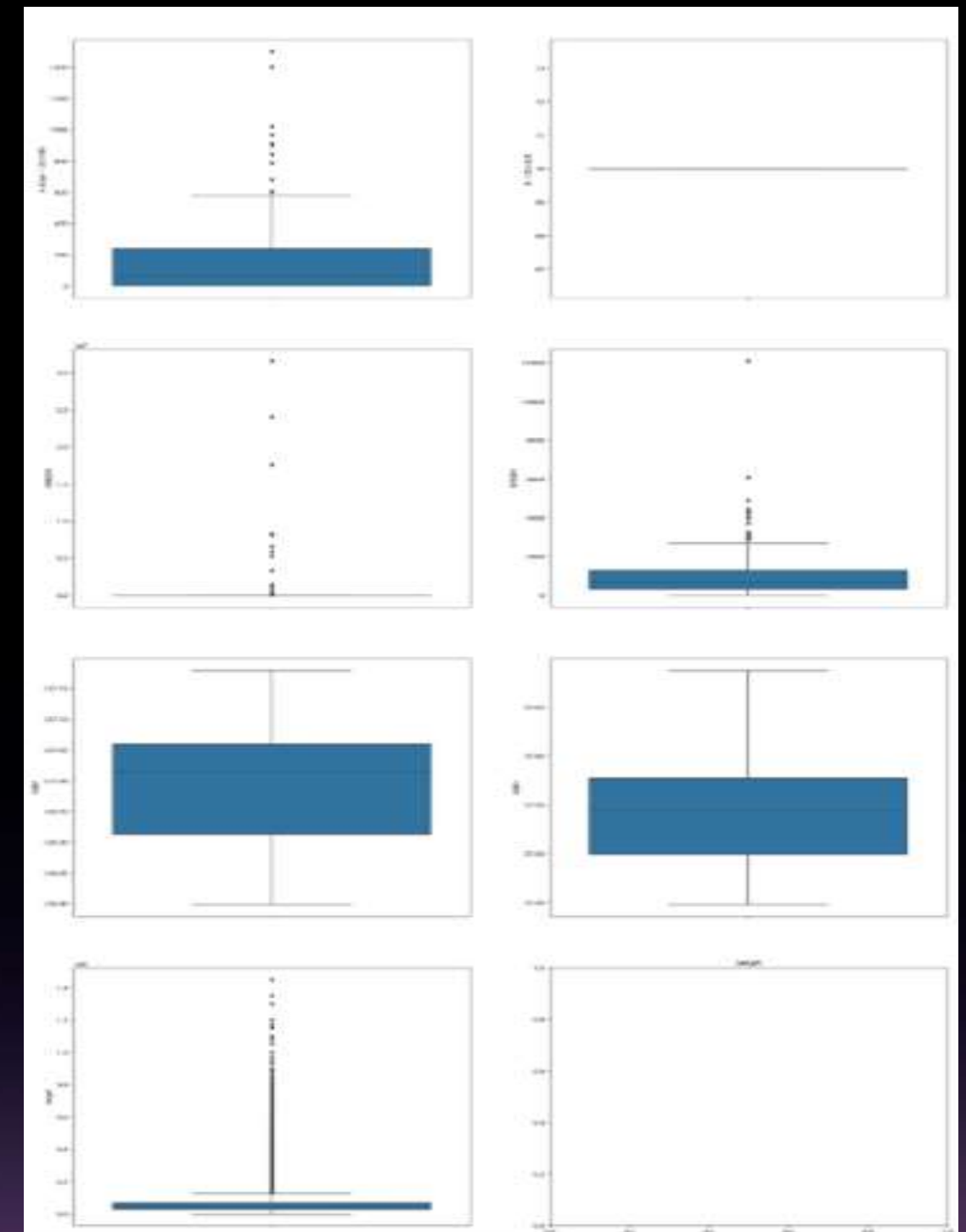
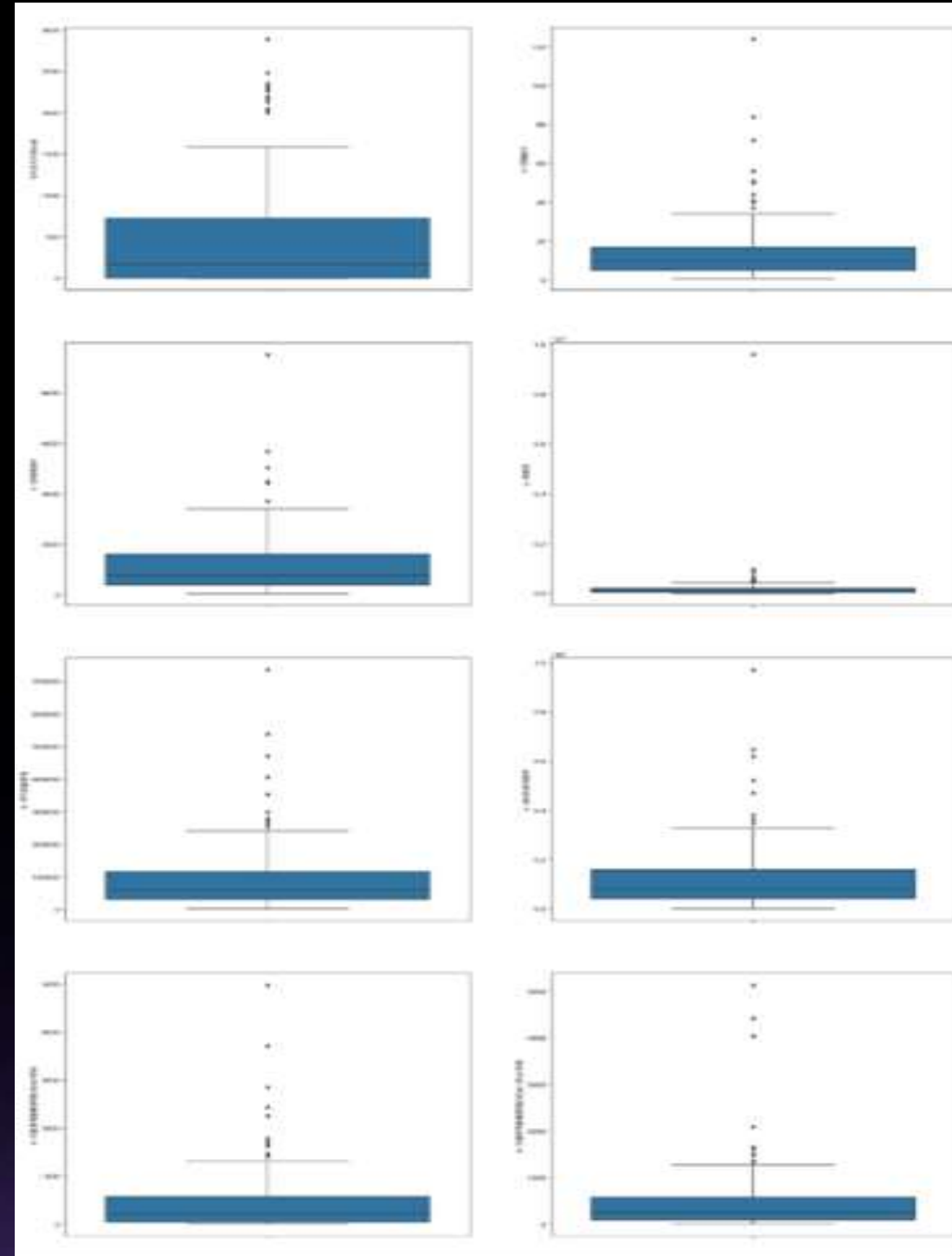
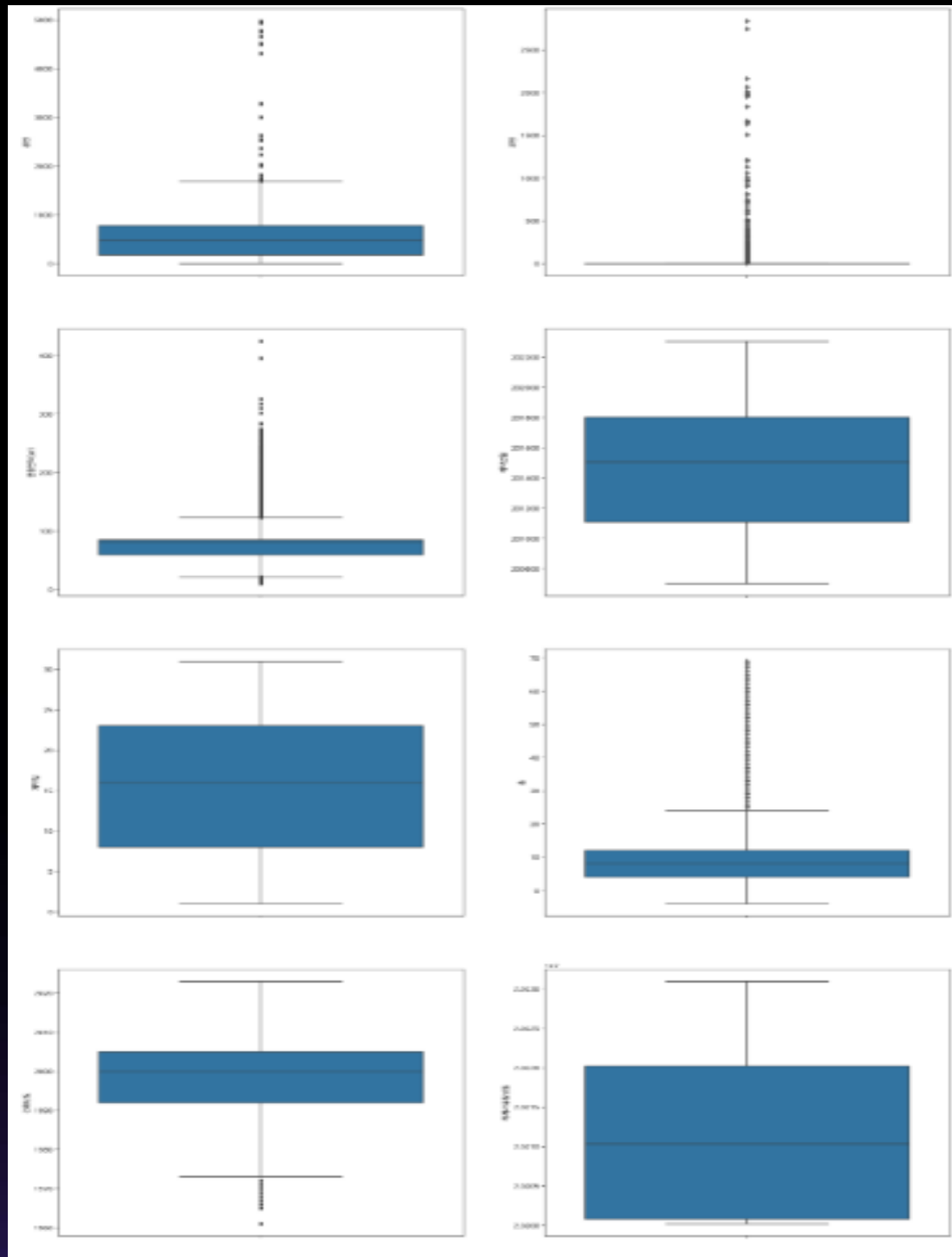
# EDA - 상관관계

- ‘면적’ 관련 feature들은 서로 상관관계가 강함
- Target과 상관관계 높은 feature는 전용면적, 주거전용면적, 연면적, 주차대수, 좌표Y(-) 임 (전용면적을 제외한 feature는 결측비율이 높음)



# EDA – 데이터 분포

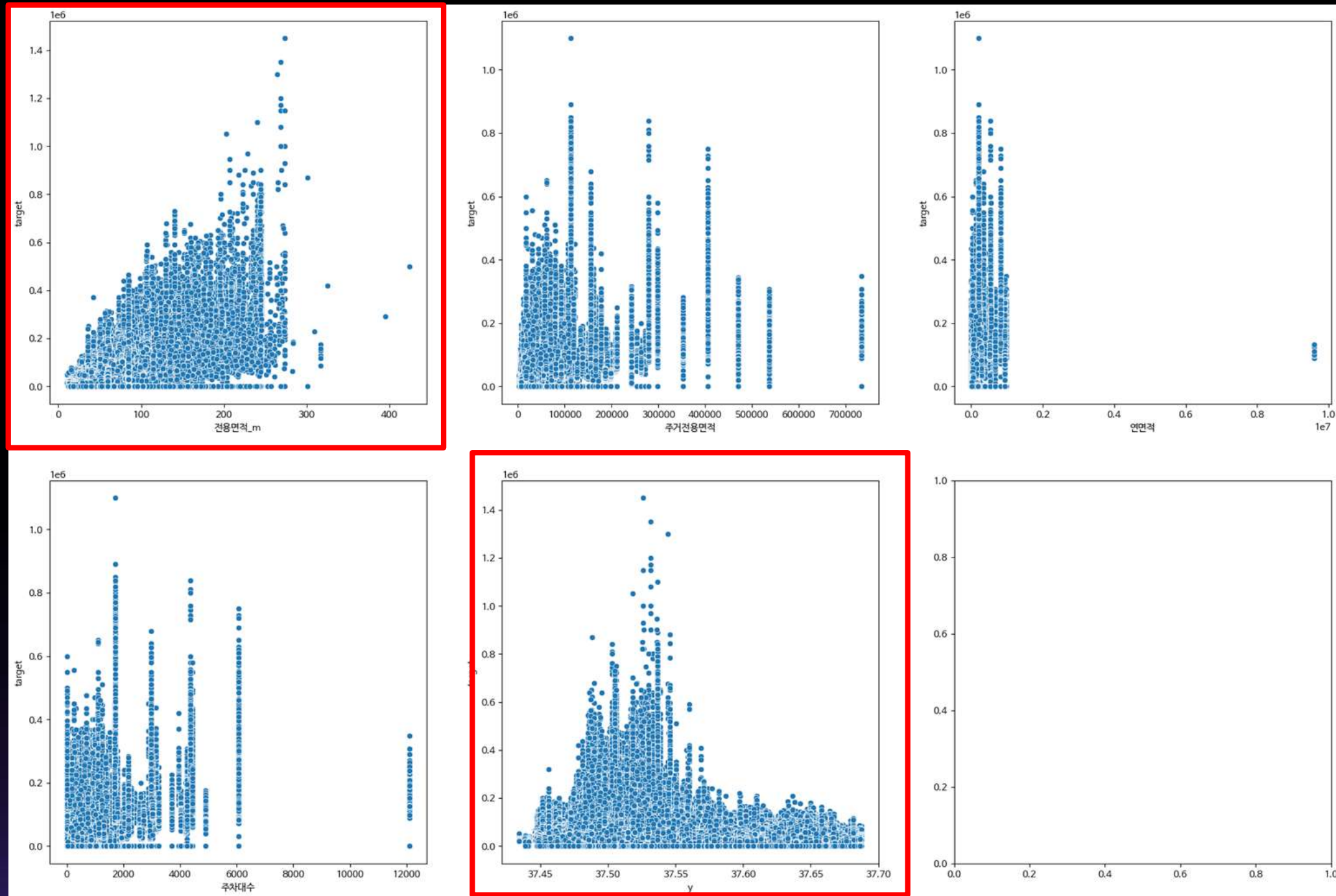
- 숫자형 Data의 대부분이 낮은 쪽으로 몰려있음 (Right-Skewed)





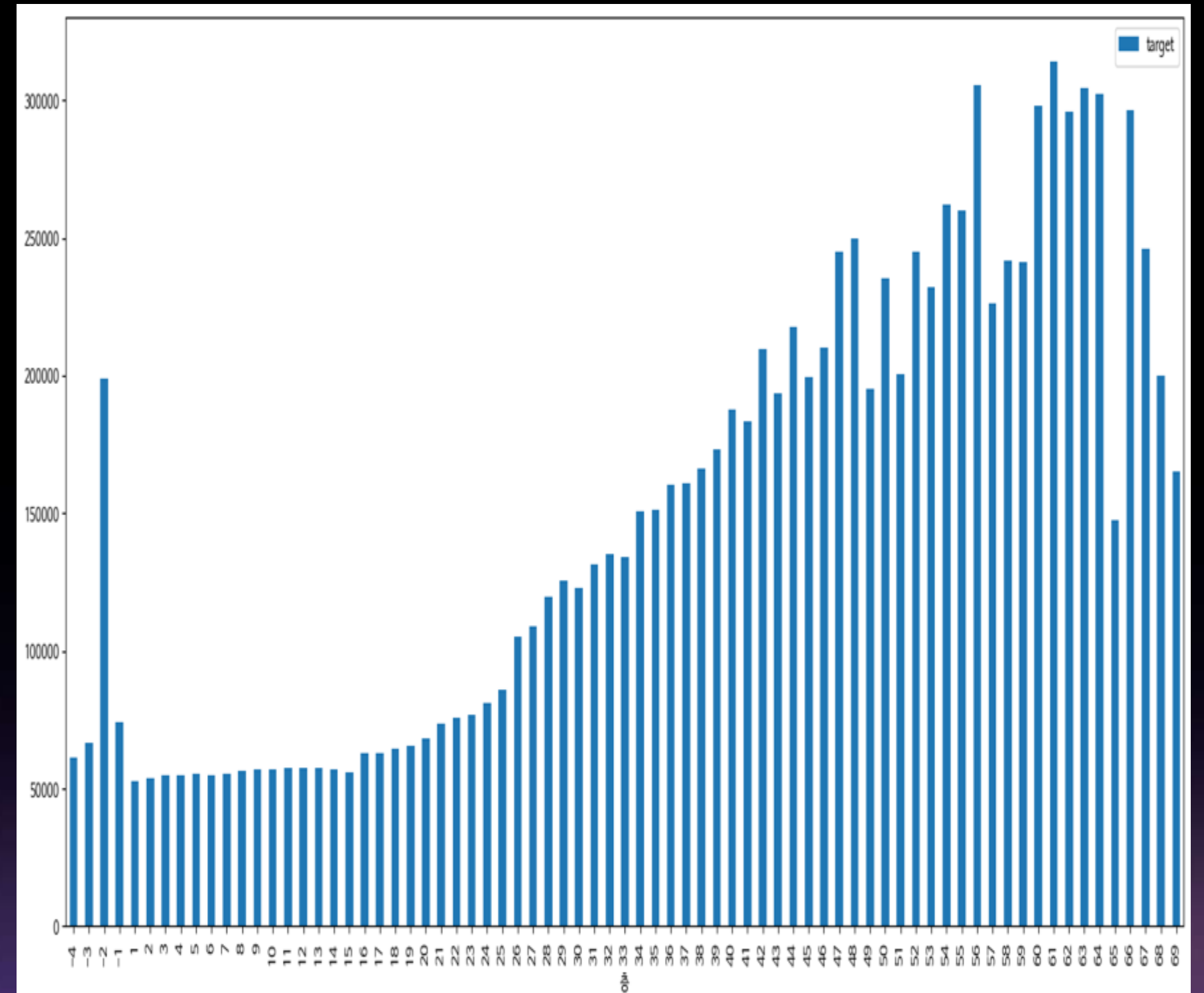
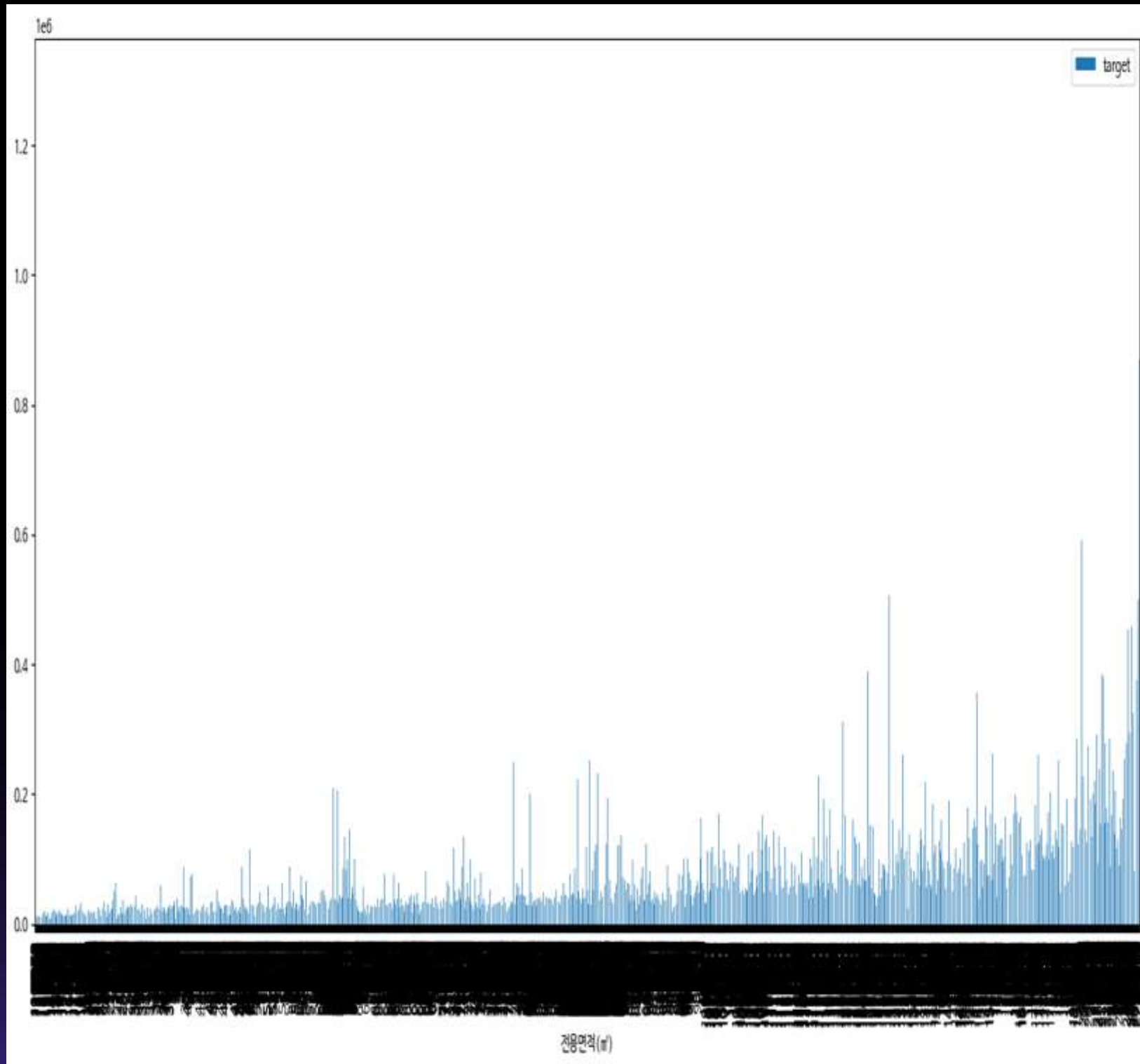
# EDA – Target과 상관관계 높은 Feature 그래프

- 전용면적 - 평균적으로 우상향하는 모습을 볼 수 있으나, outlier가 있음
- 좌표Y - 음의 상관관계보다는 특정 값에서 target의 값이 크게 높아짐
- 주거전용면적, 연면적, 주차대수- 같은 값이더라도 값의 차이가 많이남



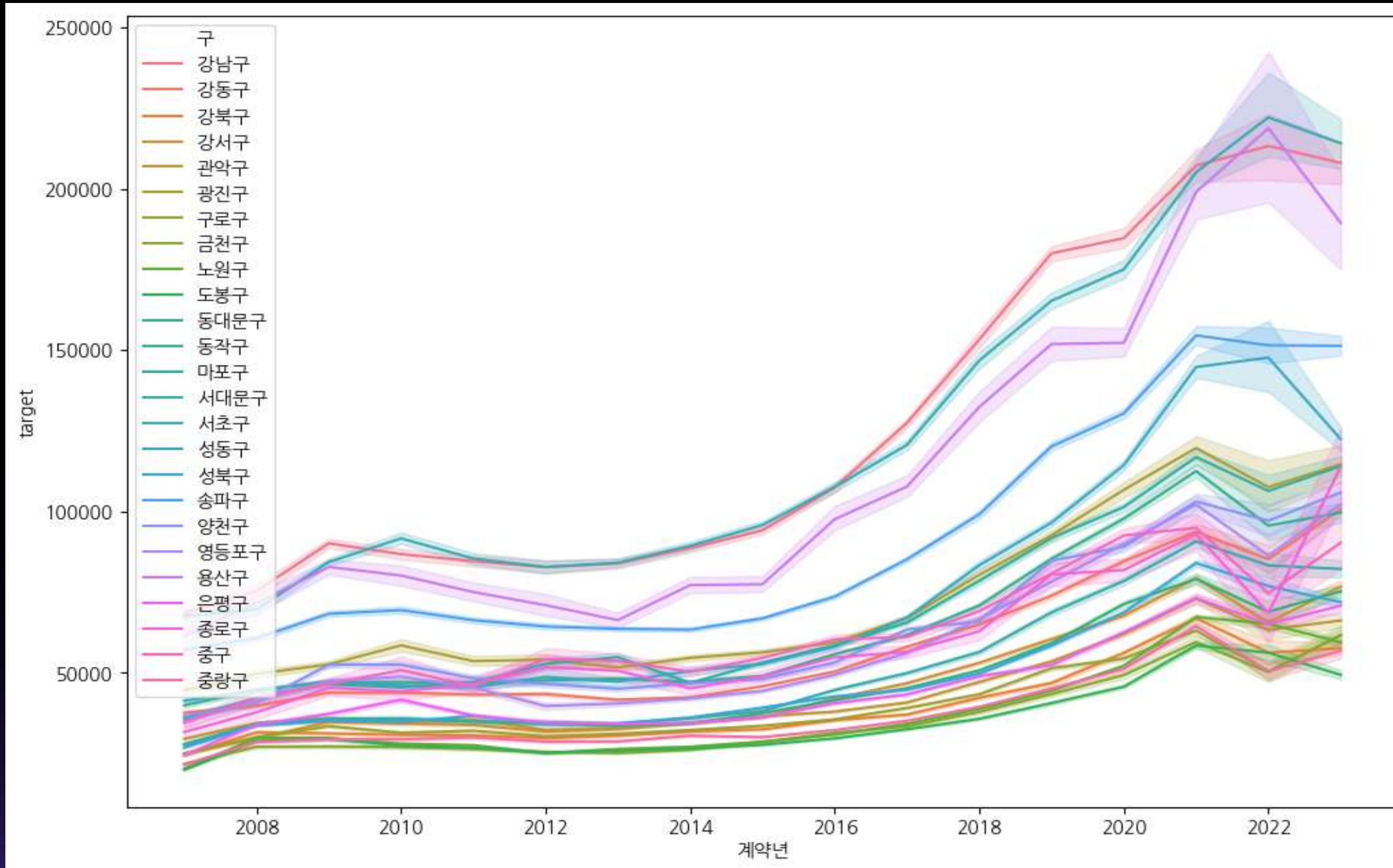
## EDA – Target과 전용면적, 층과 관계

- 전용면적 - 특정 값들이 이상치처럼 비슷한 크기의 면적들에 비해 큰 값을 보였으나 대체적으로 넓을수록 큰 값을 보임
- 층 - 고층일수록 매매 가격이 높아짐



## EDA – 구별 아파트 가격 추이

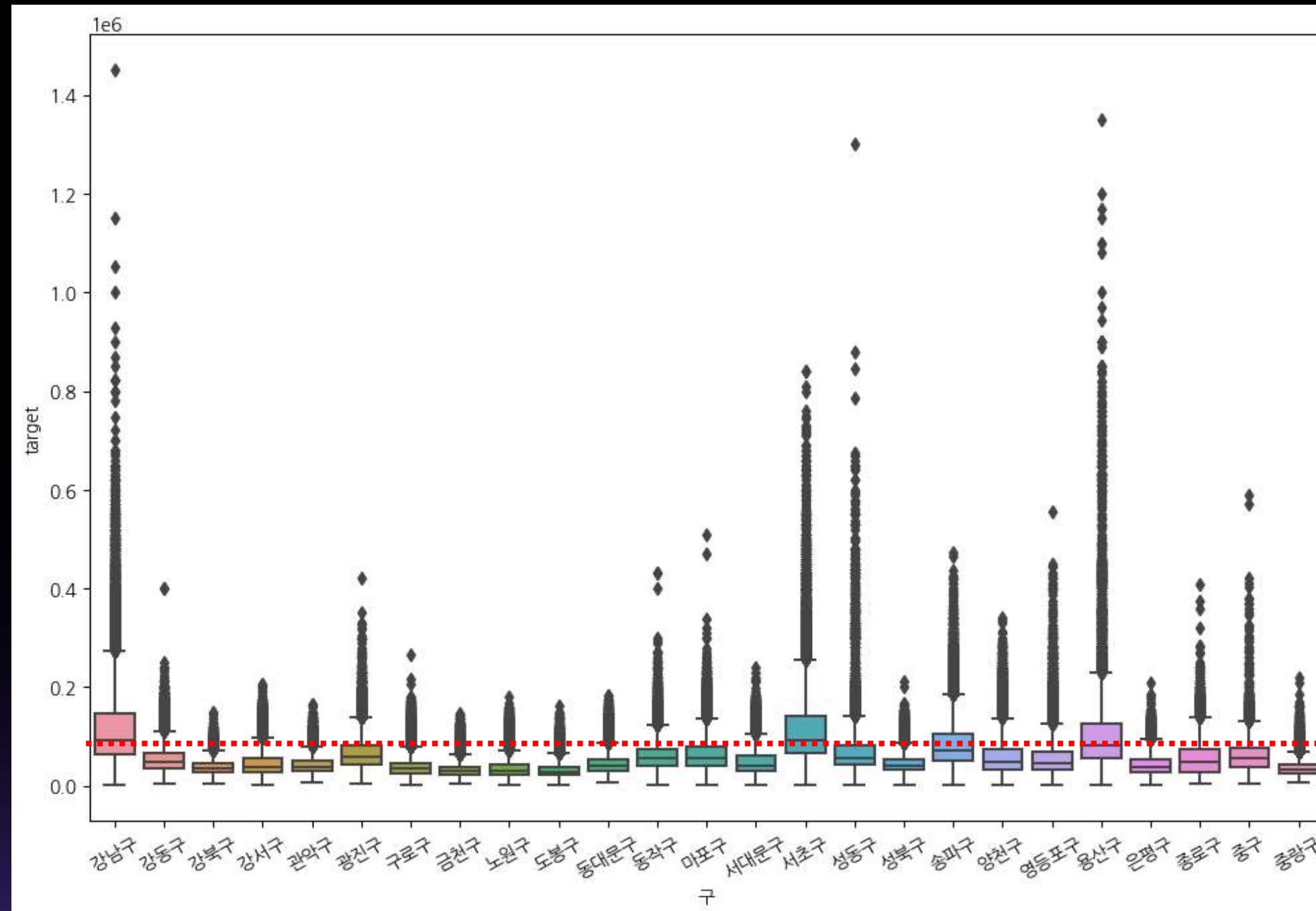
- 전체적으로 시간에 따라서 매매 가격이 높아짐
- 2022년에 상승했다가 2023년 하락하는 패턴과 2022년에 하락했다가 2023년에 상승하는 패턴이 존재





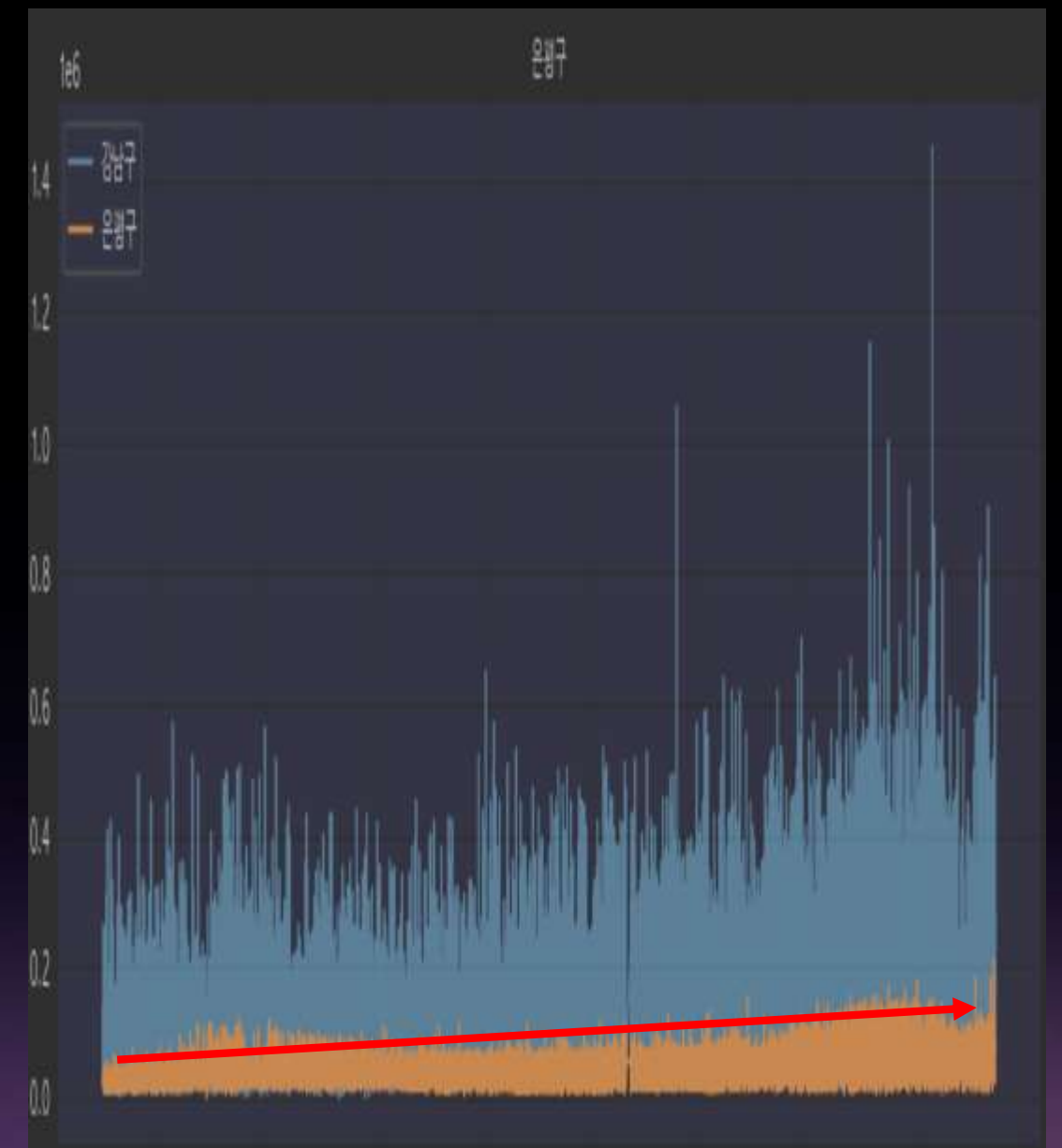
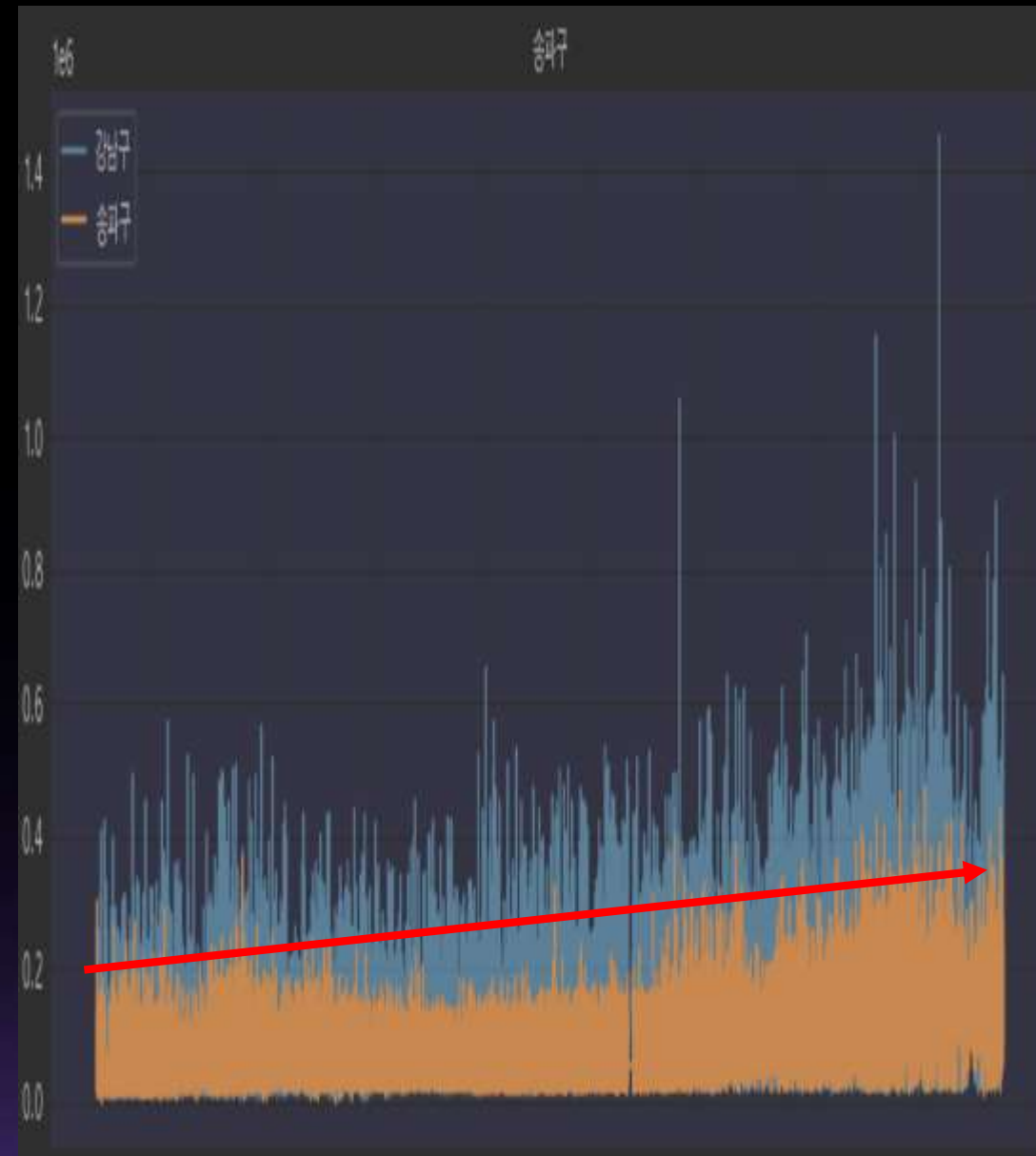
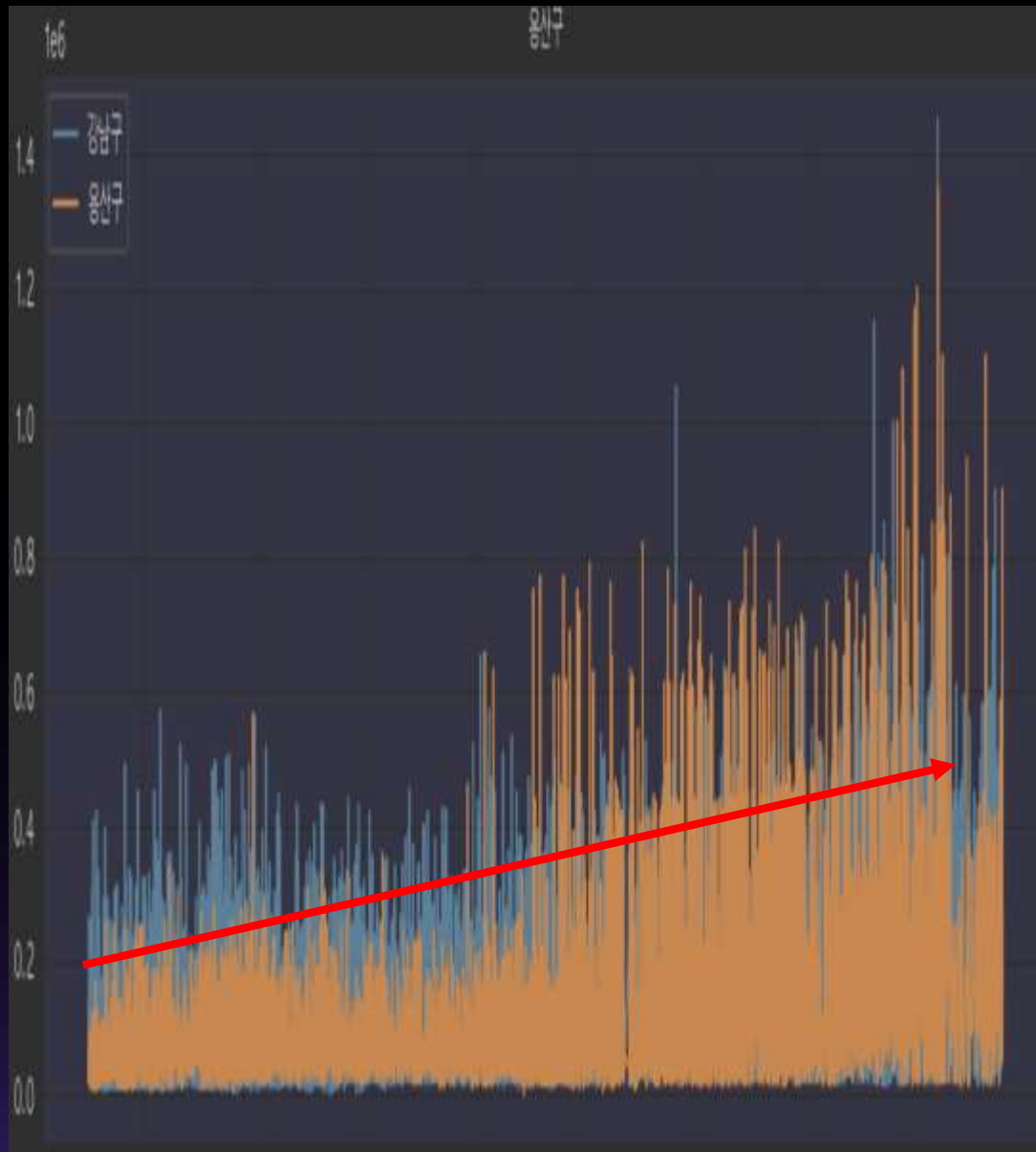
## EDA - 구별 아파트 가격 추이

- 강남구, 서초구, 송파구, 용산구가 매매 가격이 높음



## EDA - 서울시 구별 아파트 매매 가격 추이 비교(기준 강남구)

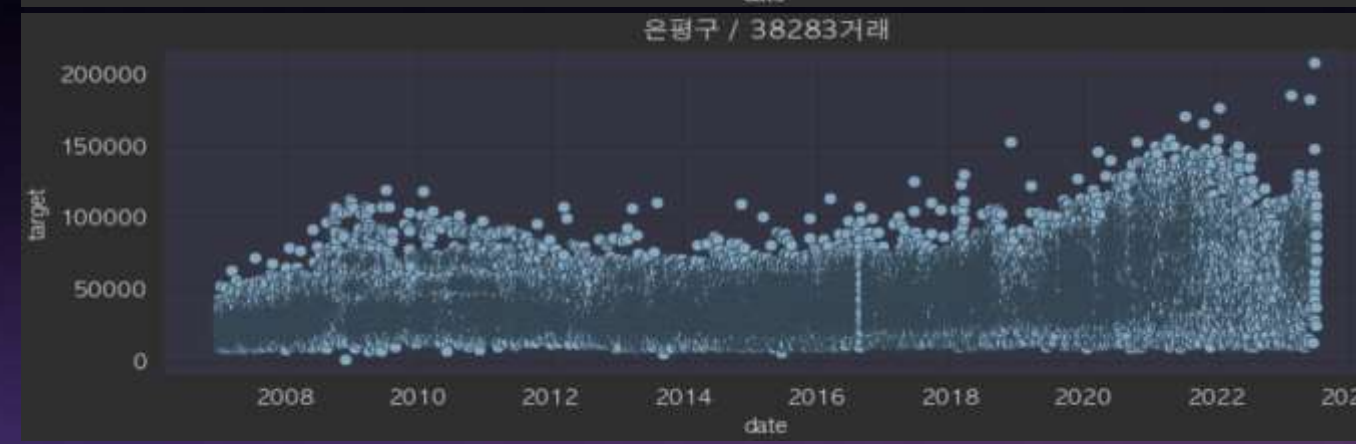
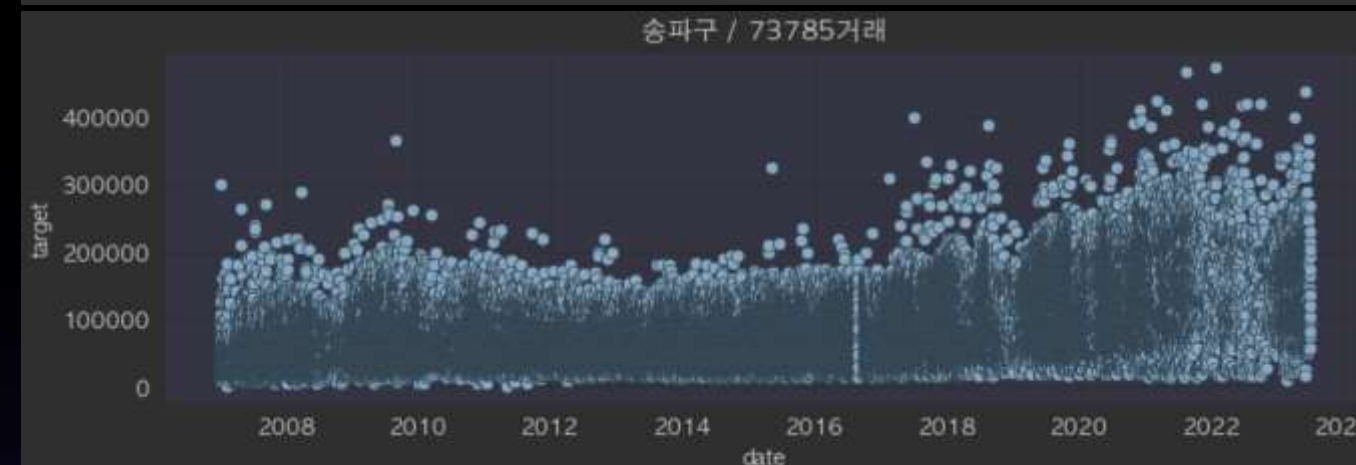
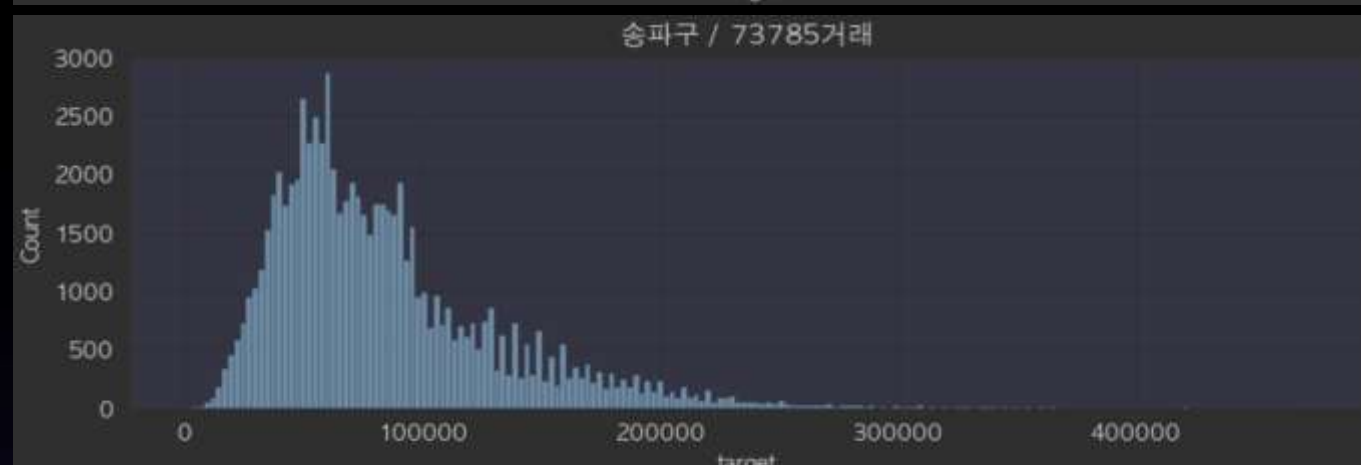
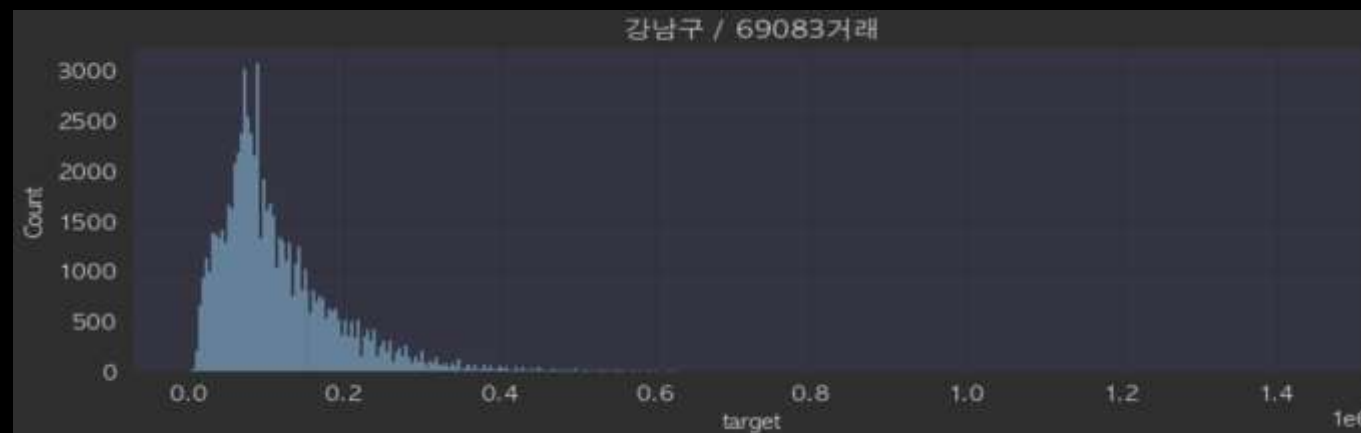
- 높은 가격으로 강남구, 서초구, 용산구, 성동구 등이 있음
- 중간 가격으로 송파구, 마포구, 양천구, 영등포구 등이 있음
- 낮은 가격으로 금천구, 강북구, 관악구, 은평구 등이 있음





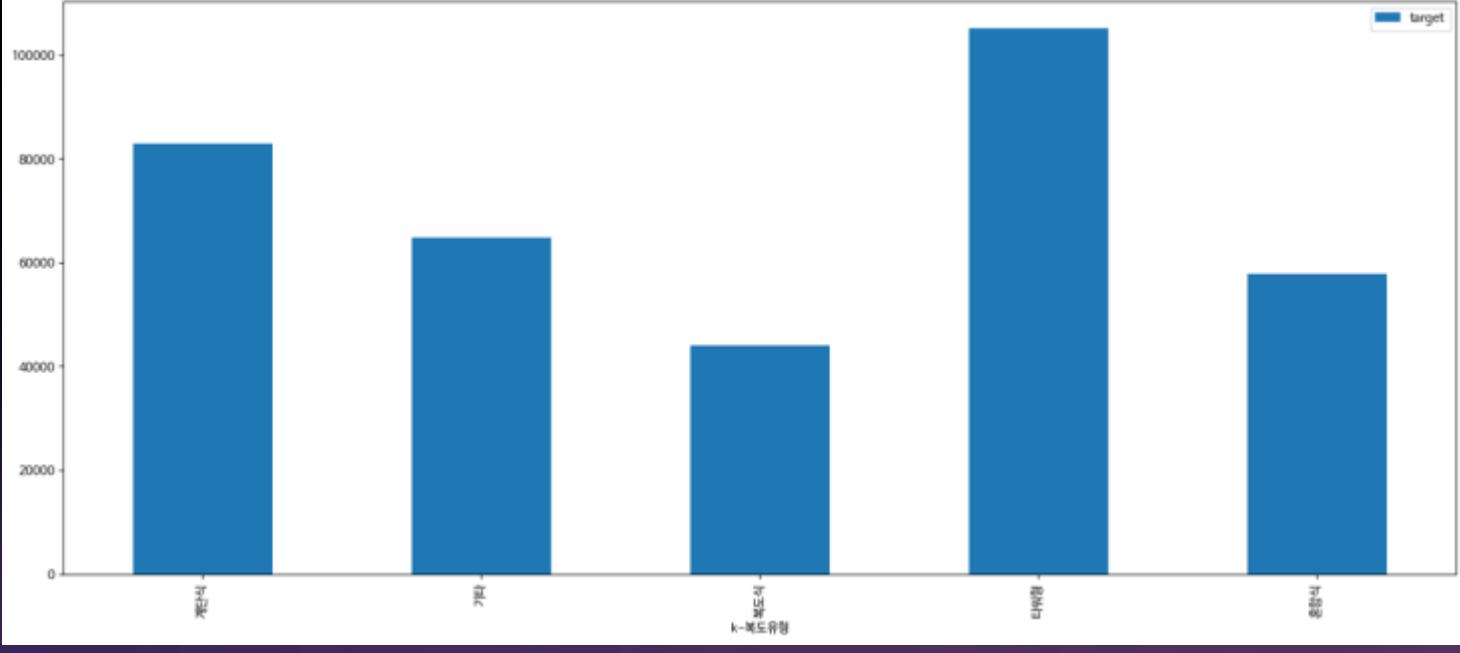
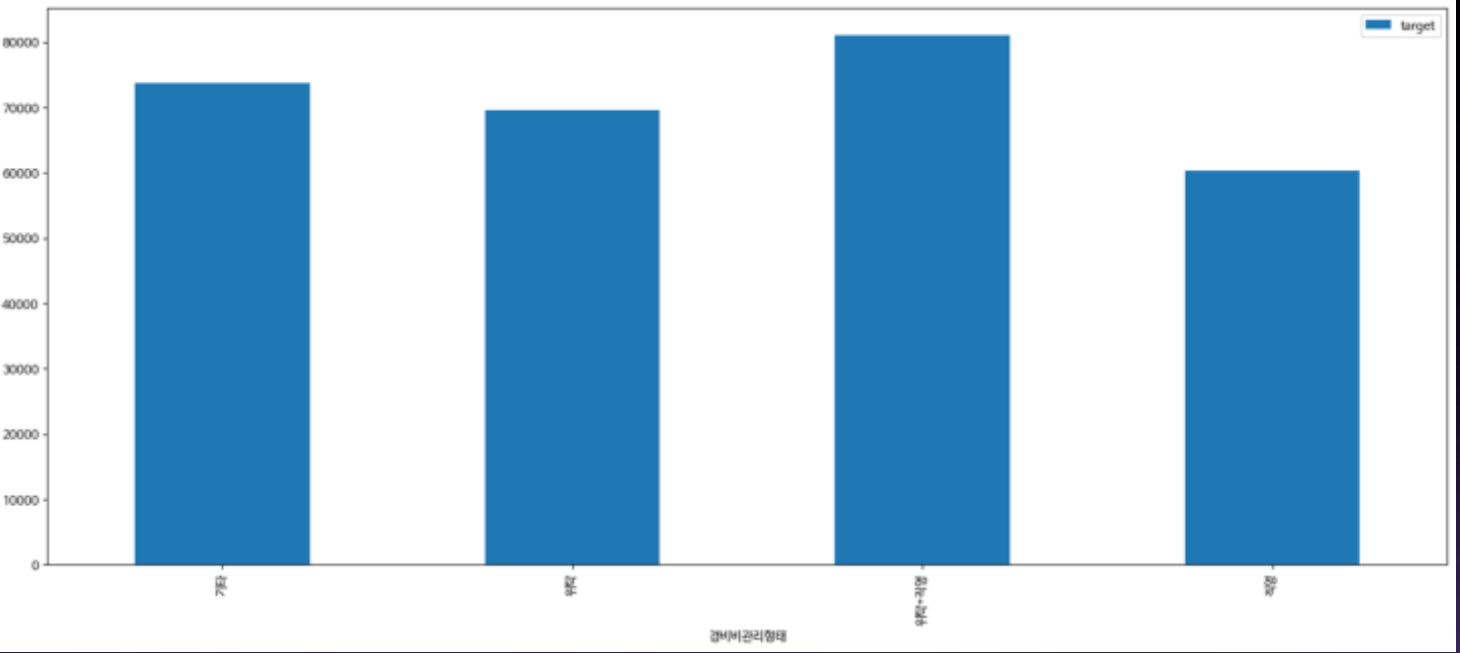
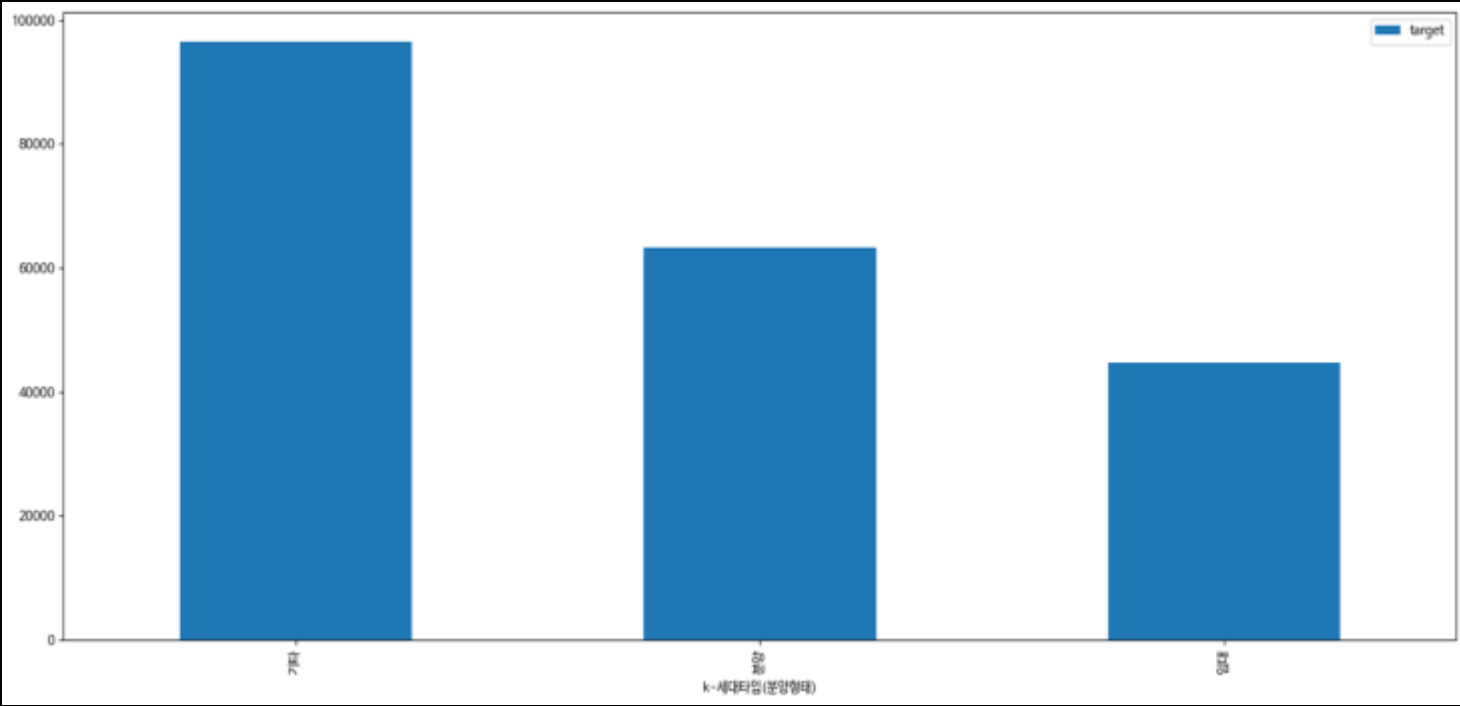
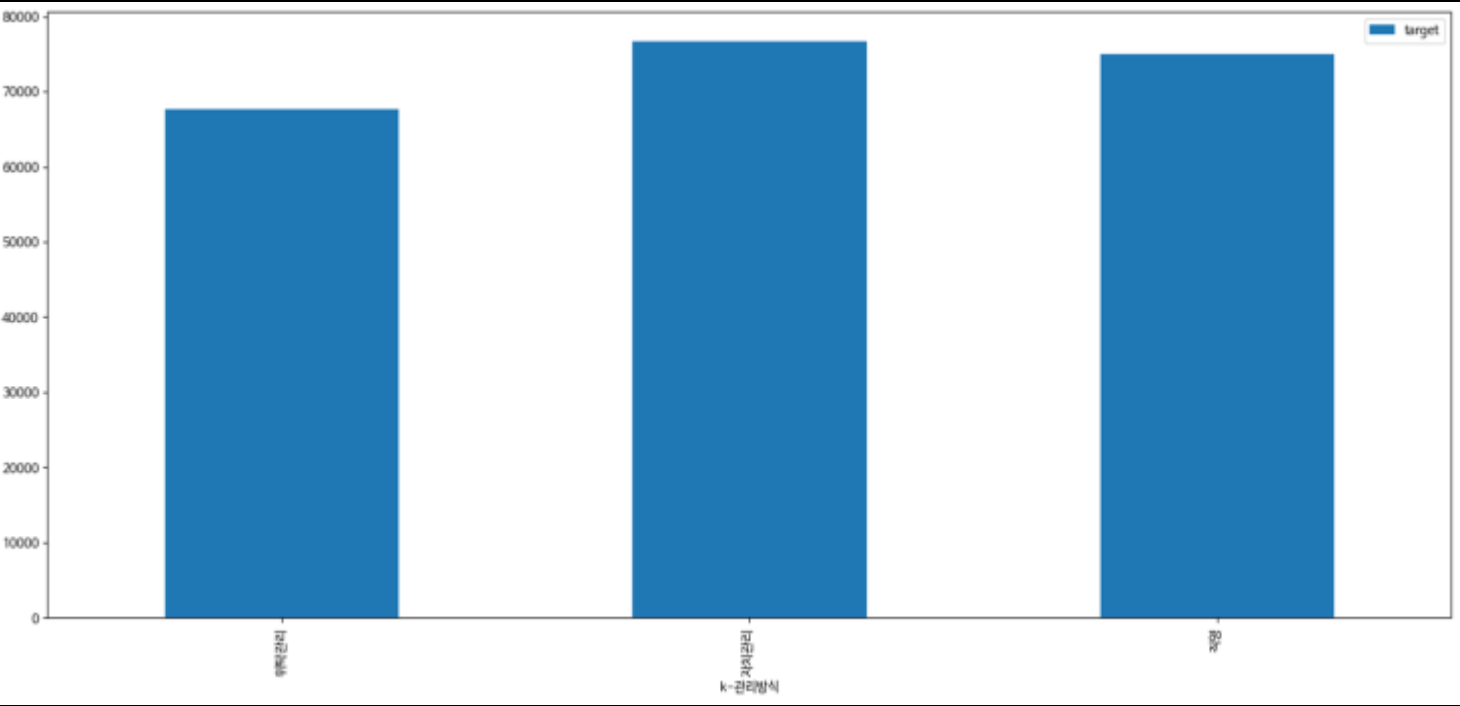
# EDA - 구별 매매 현황 시각화

- 강남구 (높은 가격) - 10억원대 주위로 매매가가 가장 많이 형성 되어 있음 / 점진적 우상향
- 송파구 (중간 가격) - 5~6억원대 주위로 매매가가 가장 많이 형성되어 있음 / 상대적으로 고가의 개별 매매가 관찰됨
- 은평구 (낮은 가격) - 3~4억원대 주위로 매매가가 가장 많이 형성되어 있음 / 2022년 전후로 가격 변동 폭이 컸음



# EDA – 관리방식, 경비비관리형태, 세대타입, 복도유형 분석

- 관리방식, 경비비관리형태 – 피처의 값에 따른 target 값에 변화가 적음
- 세대타입, 복도유형 – 피처의 값에 따라 target 값에 변화가 있지만 구분되지 않는 기타값의 갯수가 높음



# 데이터 전처리

---

## 결측치 처리

- 결측치 100만건 이상인 Feature 제거
- 좌표X, 좌표Y - 카카오맵, 구글맵 API를 통해 결측치 데이터 채움

## 이상치 처리

- 전용면적 - 사분위값을 사용하는 IQR로 전용면적의 이상치를 제외
- 범주형 변수 - NULL로 채움
- 연속형 변수 - 선형보간으로 채움

## 기타 처리

- Rows - 계약기준으로 2020년 이전데이터 제외

## 외부 데이터 활용

---

### 1. 아파트매매 실거래 가격지수

- 시군별 아파트 가격의 연도별 가격 흐름을 참조하고자 함

### 2. 아파트 규모별 매매 실거래 가격지수

- 면적에 따른 연도별 가격 흐름을 참조하고자 함

### 3. 한국은행 기준금리

- 기준금리에 따른 매매 가격 흐름을 참조하고자 함

### 4. 국토교통부 표준지공시지가

- 공시지가에 따른 매매 가격 흐름을 참조하고자 함



# Feature Engineering

## 1. 계약년월, 계약일을 활용한 Date 피처 생성 및 시계열 정렬

```
# train data를 시계열로 정렬
df_train['date'] = df_train['계약년월'].astype('str')+df_train['계약일'].astype('str')
df_train['date'] = pd.to_datetime(df_train['date'], format='%Y%m%d')
df_train = df_train.sort_values(by='date')
df_train = df_train.set_index('date')
print(df_train.index[0], '~', df_train.index[-1], ' / ', df_train.shape)

# test data를 시계열로 정렬
df_test['test_indexer'] = [i for i in range(df_test.shape[0])]
df_test['date'] = df_test['계약년월'].astype('str')+df_test['계약일'].astype('str')
df_test['date'] = pd.to_datetime(df_test['date'], format='%Y%m%d')
df_test = df_test.sort_values(by='date')
df_test = df_test.set_index('date')
TEST_INDEXER = df_test['test_indexer'].copy()
del df_test['test_indexer']
print(df_test.index[0], '~', df_test.index[-1], ' / ', df_test.shape)
```

## 2. 전용면적의 구간별 피처 생성

```
df = df.assign(**{'전용면적(㎡)': df['전용면적(㎡)'].round().astype(int)}) # train 272 / test 206
df = df.assign(**{'전용면적div3': df['전용면적(㎡)'] // 3}) # train 96 / test 80
df = df.assign(**{'전용면적div5': df['전용면적(㎡)'] // 5}) # train 60 / test 53
df = df.assign(**{'전용면적div10': df['전용면적(㎡)'] // 10}) # train 33 / test 28
df = df.assign(**{'전용면적div20': df['전용면적(㎡)'] // 20}) # train 19 / test 15
df = df.assign(**{'전용면적div30': df['전용면적(㎡)'] // 30}) # train 13 / test 11
```

# Feature Engineering

## 3. 전용면적을 5개의 카테고리별로 나누어 매월 매매지수 'index\_region' 피쳐 생성

```
def add_index_region_feature(df):
    df_scale = pd.read_csv('../apartment_scale_index.csv', encoding='cp949')
    df_scale
    def give_grade_area_str(x):
        if x == '초소형(40㎡ 이하)': return '0'
        elif x == '소형(40㎡초과 60㎡이하)': return '1'
        elif x == '중소형(60㎡초과 85㎡이하)': return '2'
        elif x == '중대형(85㎡초과 135㎡이하)': return '3'
        elif x == '대형(135㎡ 초과)': return '4'
    def give_grade_area_int(x):
        if x <= 40: return '0'
        elif x <=60: return '1'
        elif x <=85: return '2'
        elif x <=135: return '3'
        else: return '4'
    df_scale['전용면적cat'] = df_scale['주거면적'].apply(lambda x: give_grade_area_str(x))
    df['전용면적cat'] = df['전용면적(㎡)'].round().apply(lambda x: give_grade_area_int(x))
    merged_df = pd.merge(df, df_scale[['전용면적cat', '계약년월', 'index_region']],
                          on=['전용면적cat', '계약년월'], how='left')
    return merged_df
```

## 4. 아파트 매매 가격 / 전용면적 별 매매지수 => 'ratio' 피쳐 생성

```
def add_ratio_feature(df):
    df['ratio'] = df['target'] / df['index_region']
    df['ratio'] = df['ratio'].round().astype(int)
    return df
df_train = add_ratio_feature(df_train)
```



# Feature Engineering

## 5. Train data를 활용하여 Test data에 Ratio 피쳐 생성

1) (도로명, 전용면적) Groupby 사용하여 Test data의 같은 그룹에 2020년, 2015년, 2007년 최신 거래 우선으로 Ratio 적용

```
for year in ['2020', '2015', '2007']:
    df_concat = pd.concat([df_train[df_train.index >= year], df_test_temp], axis=0)
    print(f'도로명 {year}:', df_concat['ratio'].isna().sum())
    for i in ['전용면적(㎡)']:
        df_concat['ratio'] = df_concat.groupby(['도로명', i])['ratio'].transform(lambda x: x.fillna(x.mean()))
        print(i, df_concat['ratio'].isna().sum())
    print(df_concat['ratio'].isna().sum(), '/', df_test_temp.shape[0])
    df_test_temp = df_concat[df_concat.index >= '2023.07']
```

도로명 2020: 9272  
전용면적(㎡) 311  
311 / 9272

도로명 2015: 311  
전용면적(㎡) 163  
163 / 9272

도로명 2007: 163  
전용면적(㎡) 141  
141 / 9272

2) (동~구, 전용면적div5~30) Groupby를 사용하여 위 방법에 모두 채워지지 않은 나머지 Ratio 적용

```
interpolate_ (variable) interpolate_date: Literal['2020']
df_concat = df_concat.groupby(interpolate_date, df_test_temp], axis=0)
print(f'동 {interpolate_date}:', df_concat['ratio'].isna().sum())
for i in ['전용면적div5', '전용면적div10', '전용면적div20', '전용면적div30']:
    df_concat['ratio'] = df_concat.groupby(['동', i])['ratio'].transform(lambda x: x.fillna(x.median()))
    print(i, df_concat['ratio'].isna().sum())
print(df_concat['ratio'].isna().sum(), '/', df_test_temp.shape[0])
df_test_temp = df_concat[df_concat.index >= '2023.07']
```

동 2020: 141  
전용면적div5 72  
전용면적div10 69  
전용면적div20 65  
전용면적div30 65  
65 / 9272

구 2020: 65  
전용면적div5 0  
전용면적div10 0  
전용면적div20 0  
전용면적div30 0  
0 / 9272

# Feature Engineering

## 6. 계약일자를 기준으로 은행 기준 금리 피처 생성 (데이터 출처 : 한국은행 기준금리 추이)

```
with open('../data/interest_rate.csv') as f:
    interest = pd.read_csv(f)

#출처 : https://www.bok.or.kr/portal/singl/baseRate/list.do?dataSeCd=01&menuNo=200643

t = interest
t = t.astype('str')
interest.loc[:, 'datetime'] = t['year'] + t['month'].apply(lambda x: '0'+x if len(x) == 1 else x) + t['date'].apply(lambda x: '0'+x if len(x) == 1 else x)
interest.sort_values(by = ['year', 'month', 'date'], ascending=True, inplace=True)
interest.reset_index(inplace=True)
interest.drop(columns = 'index', inplace=True)

import datetime
df['interest_rate'] = [-1] * len(df)
for i in range(len(df)):
    contract_date = str(df.loc[i, '계약년']) + '-' + str(df.loc[i, '계약월']) + '-' + str(df.loc[i, '계약일'])
    contract_date = datetime.datetime.strptime(contract_date, '%Y-%m-%d')
    for j in range(len(interest)-1):
        compare_date1 = datetime.datetime.strptime(interest.loc[j, 'datetime'], '%Y%m%d')
        compare_date2 = datetime.datetime.strptime(interest.loc[j+1, 'datetime'], '%Y%m%d')
        if (compare_date1 < contract_date) and (contract_date < compare_date2):
            df.loc[i, 'interest_rate'] = interest.loc[j, 'rate']
            break

df.loc[:, 'interest_rate'] = df['interest_rate'].apply(lambda x : 3.5 if x == -1 else x)
```

## 7. 강남여부 피처 생성

```
def gangnam_parser(x):
    gu_li = ['강서구', '영등포구', '동작구', '서초구', '강남구', '송파구', '강동구']
    if x.split(' ')[1] in gu_li:
        return 1
    else:
        return 0

df.loc[:, 'is_gangnam'] = df['시군구'].apply(gangnam_parser)

df.drop(columns = ['시군구', '아파트명'], inplace = True)
```



# Feature Engineering

## 8. 계약년월을 계약년, 계약월 로 분리

```
def year_month_parser(x):
    year = int(str(x)[0:4])
    month = int(str(x)[4:6])
    return [year, month]

train['계약년'] = train['계약년월'].apply(lambda x : year_month_parser(x)[0])
train['계약월'] = train['계약년월'].apply(lambda x : year_month_parser(x)[1])
train.drop(columns = '계약년월', inplace = True)
```

## 9. 역세권 피쳐 생성

- 지하철 승강장과의 거리가 500m이하 이면 1, 지하철 승강장과의 거리가 500m초과하면 0

```
with open('../data/subway_feature.csv') as f:
    subway_df = pd.read_csv(f)

def subway_distance(x, y):
    y_building = y
    x_building = x
    for i in range(len(subway_df)):
        x_subway = subway_df.loc[i, '경도']
        y_subway = subway_df.loc[i, '위도']

        x_distance = abs(x_building - x_subway)
        y_distance = abs(y_building - y_subway)

        #위도 경도 변환
        x_distance = 88000 * x_distance
        y_distance = 110000 * y_distance

        distance = np.sqrt(x_distance ** 2 + y_distance ** 2)
        if distance <= 500:
            return 1

    return 0

tmp = train.progress_apply(lambda row : subway_distance(row['x'], row['y']), axis = 1)

tmp.to_csv('../data/is_subway.csv', index = False)

with open('../data/is_subway.csv') as f:
    tmp = pd.read_csv(f)

df['is_subway'] = tmp
```

# Feature Engineering

## 10. Target값 변경

1) target을 도로명 group으로 묶어서 평균처리 (엄효범님 idea)

```
df['price'] = df[df['is_test'] == 0].groupby(['도로명'])['target'].transform('mean')
```

2) 전용면적을 5개의 유형으로 구분한 후 이를 group으로 묶어서 평균처리 (엄효범님 idea)

```
1 def class_area(data):  
2     if data < 40: #16평 이하  
3         result = 'small'  
4     elif data >= 40 and data < 61: #16~24평  
5         result = 'mid-small'  
6     elif data >= 61 and data < 86: #24~33평  
7         result = 'middle'  
8     elif data >= 86 and data < 132: #34~40평  
9         result = 'mid-large'  
10    elif data >= 132 and data < 166: #41~50평  
11        result = 'large'  
12    else: # 51평 이상  
13        result = 'huge'  
14    return result
```

```
train_x['price'] = train_x.groupby(['area_class'])['target'].transform('mean')
```

```
train_x['area_class'] = train_x['전용면적(m²)'].apply(class_area)
```

# Feature Engineering

## 11. 신축 / 재건축 피처 생성

- 건축년도가 2009년 이후이면 신축, 1990년 이전이면 재건축

```
concat_select['신축여부'] = concat_select['건축년도'].apply(lambda x: 1 if x >= 2009 else 0)
concat_select['재건축'] = concat_select['건축년도'].apply(lambda x: 1 if x < 1990 else 0)
```

## 12. 브랜드 피처 생성

- 아파트가 유명 브랜드이면 1, 아니면 0

```
top10 = ['자이', '푸르지오', '더샵', '롯데캐슬', '이편한|e편한|e-편한',
        '힐스테이트', '아이파크', '래미안', 'sk|SK|에스케이', '데시앙']
```

```
apt_names = ['그레이스', '양지', '쌍용', '현대', '한신', '삼성', '대우', '신동아', '두산', '주공',
            '우성', '벽산', '동원로알듀크', '경남', '삼환', '쌍용', '삼익', '대림', '코오롱', '파크리오',
            '엘지', '성원', '잠실', '동궁리치웰', '동성']
```

```
# top 10 시공사 키워드와 25개 리스트를 통합
```

```
apt_names_list = top10 + apt_names
```

```
concat_select['브랜드'] = 'others'
```

```
for brand in tqdm(apt_names_list):
```

```
    concat_select.loc[concat_select['아파트명'].str.contains(brand), '브랜드'] = brand
```

## 13. 건물연수 피처 생성

```
concat_select['건물연수'] = concat_select['계약년월일'] // 10000 - concat_select['건축년도']
```

# Feature Engineering

14. year\_avg\_price / year\_max\_price / price\_ratio 피처 생성
- 아파트명, 전용면적, year 그룹의 평균 값과 최대값
  - (최대값 - 평균값) / 최대값으로 비율 계산

```
concat_select['year_avg_price'] = concat_select.groupby(['아파트명', '전용면적', 'year'])['target'].transform('mean')
concat_select['year_max_price'] = concat_select.groupby(['아파트명', '전용면적', 'year'])['target'].transform('max')
concat_select['price_ratio'] = (concat_select['year_max_price'] - concat_select['year_avg_price']) / concat_select['year_max_price']
```

15. 전년지가 피처 생성

```
dt_landprice['addr'] = dt_landprice['시도명'] + ' ' + dt_landprice['시군구명'] + ' ' + dt_landprice['소재지']
dt_landprice = dt_landprice[['addr', '전년지가']].drop_duplicates(subset = ['addr'])
concat = pd.merge(concat, dt_landprice, how = 'left', on='addr')
concat['전년지가'] = concat['전년지가'].fillna(concat.groupby('시군구')['전년지가'].transform('mean'))
```

16. 청소비관리형태통합 피처 생성

```
concat_select['청소비관리형태'].fillna('기타')

def define_wash_fee(fee) :
    if fee == '위탁+직영' :
        return 2
    elif fee == '위탁' or fee == '직영' :
        return 1
    else :
        return 0

concat_select['청소비관리형태통합'] = concat_select['청소비관리형태'].apply(define_wash_fee)
```

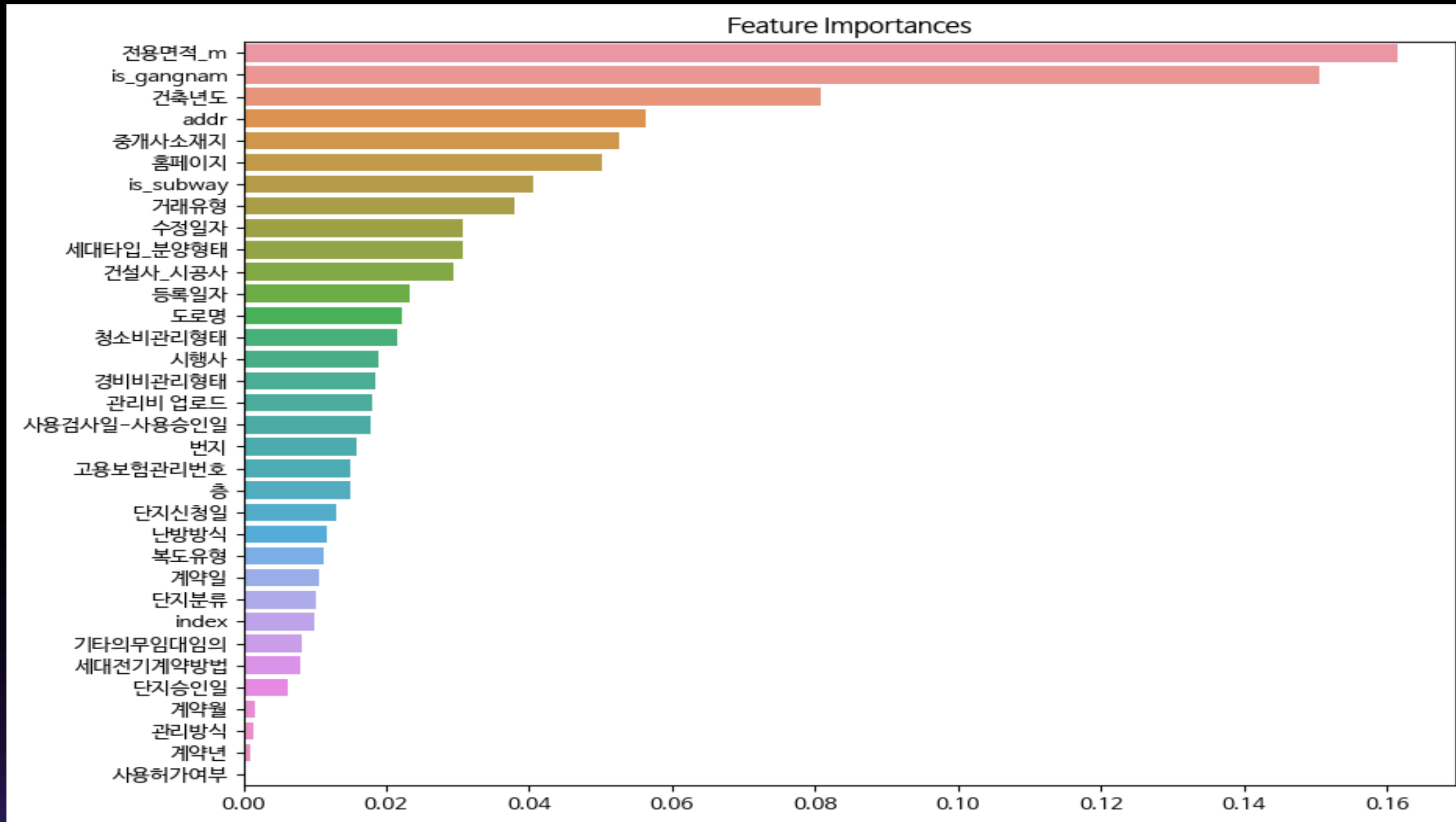
# Feature Engineering

## 17. 고층여부 피쳐 생성

```
def is_high_floor(floor) :  
    if floor < 0 :  
        return -1  
    elif floor >= 0 and floor <= 25 :  
        return 0  
    elif floor > 25 and floor <= 33 :  
        return 1  
    elif floor > 33 and floor <= 49 :  
        return 2  
    elif floor > 49 and floor <= 67 :  
        return 3  
    else :  
        return 4  
  
concat_select['고층여부'] = concat_select['층'].apply(is_high_floor)
```

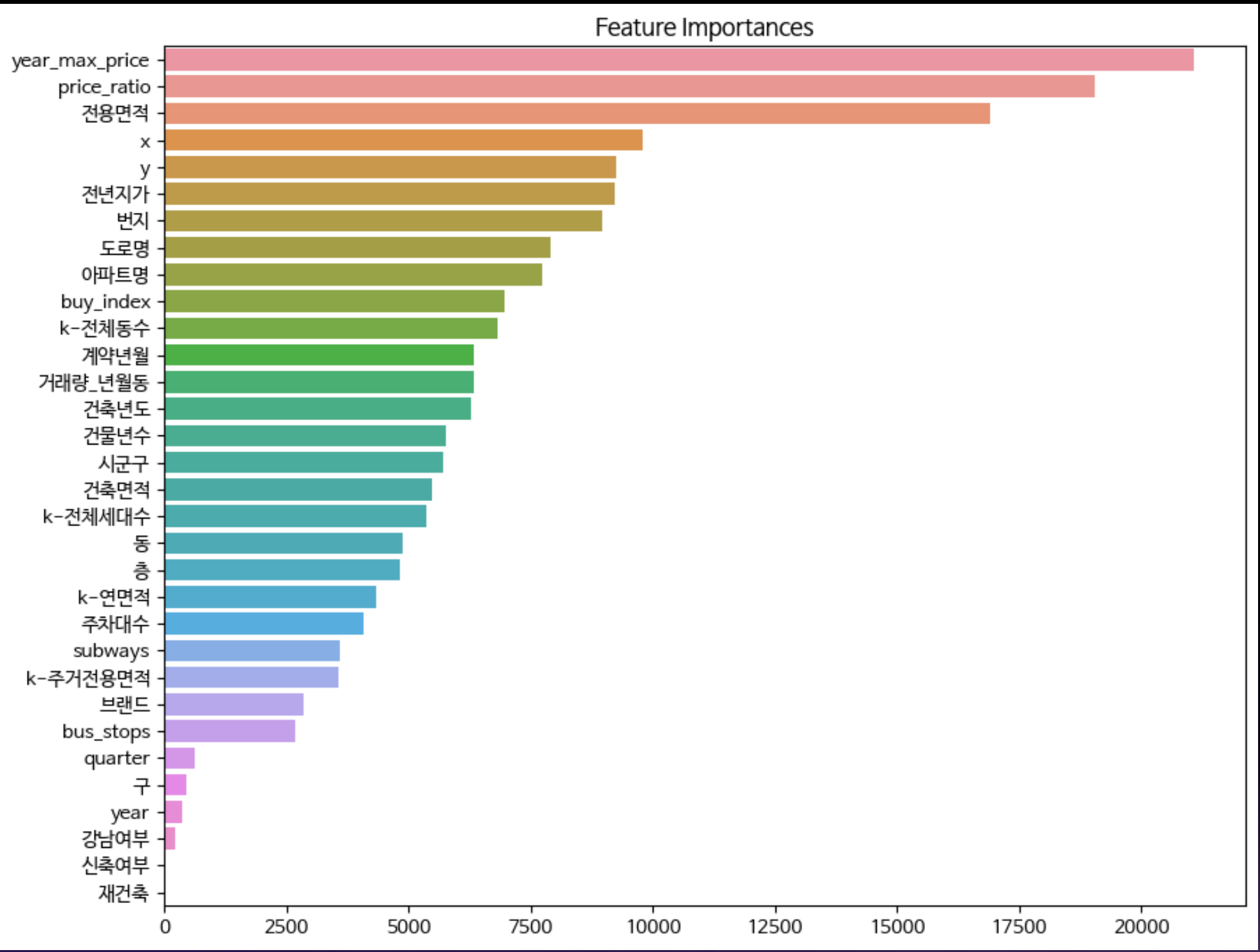
# Feature Select

- XGBRegressor().feature\_importances\_, InterpretML의 그래프를 통해 '전용면적', '강남여부', '건축년도', '주소' 선택



# Feature Select

- LGBMRegressor().feature\_importances\_의 그래프를 통해 'year\_max\_price, price\_ratio, 전용면적, x, y, 전년지가' 선택





04

# Modeling



# Model Select

이름	사용모델	적용 결과	비고
김소연	▪ Light GBM	▪ <b>Hyper parameter</b> 를 튜닝보다 <b>feature</b> 를 선택하는 것이 더 많은 성능향상을 이끌어 낼 것으로 기대하여 <b>default</b> 값들로 수행	
	▪ LSTM	▪ 건물의 신축 여부, 전체적으로 시간이 지나며 우리나라의 집값이 상승한 추세였기에 시계열 모델인 LSTM으로도 실험 ▪ 계약년, 계약월을 기준으로 데이터 정렬	
권혁찬	▪ Light GBM		
김태한	▪ XGBoost	▪ 'n_estimators', 'max_depth', 'colsample_bynode', 'reg_lambda', 'learning_rate'을 Optuna로 튜닝. ▪ 가장 중요도가 높은 hyperparameter를 고정하고 다시 튜닝하는 방식으로 수행. ▪ <b>약 13시간 가량의 시간이 소요되어 지나치게 오래 걸림. 1회 수행 후 나머지 시도는 모두 LGBM으로 수행</b>	
	▪ Light GBM	▪ 데이터 수(row의 수)가 3만개가 넘고, XGB는 너무 오래 걸리기 때문에 모델을 LGBM으로 고정하고 실험들 수행. ▪ 하이퍼파라미터 튜닝 보다는 피쳐의 선택이 중요하다고 생각해서 LGBM은 디폴트 하이퍼파라미터로 수행	
문정의	▪ Light GBM	▪ <b>Hyper parameter</b> 를 튜닝을 하는 것보다 <b>Feature</b> 를 추가 / 삭제 / 보정 하는 부분에서 <b>성능 향상이 나타남</b>	
이현진	▪ Light GBM ▪ XGBoost	▪ 데이터의 크기가 커 대부분의 모델링은 LGBM을 이용해서 시도함 ▪ 다만 팀원의 대부분이 LGBM을 시도하여, 추가적으로 XGB에서의 성능을 확인하기 위해 후반부에는 XGB를 이용한 모델링을 시도	

# Various Models with Trial and Error (김소현)

Feature Engineering	사용한 모델	Public Score
<ul style="list-style-type: none"><li>▪ <b>Baseline</b> 전처리 적용</li><li>▪ <b>15층 이상이면 '고층여부' feature</b> 추가</li><li>▪ <b>['k-전화번호', 'k-팩스번호', 'k-건설사(시공사)', 'k-시행사'] drop</b></li></ul>	LGBM	<b>98966.4035</b>
<ul style="list-style-type: none"><li>▪ Baseline 전처리 적용</li><li>▪ 15층 이상이면 '고층여부' feature 추가</li><li>▪ ['k-전화번호', 'k-팩스번호'] drop, 'k-135초과' 컬럼 추가 후 nan 값을 0으로 대체</li></ul>		98174.0075
<ul style="list-style-type: none"><li>▪ [k-관리방식, 경비비관리형태, k-홈페이지, k-전화번호, k-팩스번호, 단지소개기존clob, 고용보험관리번호] drop</li><li>▪ 기타/의무/임대/임의=1/2/3/4 : 결측치 모두 기타 처리</li><li>▪ 청소비관리형태 : 위탁+직영은 2, 위탁or직영1, 기타 0으로 재그룹화</li><li>▪ 'k-135 초과' 컬럼 추가 후 nan 값을 0으로 대체</li><li>▪ 단지승인일, 단지신청일, k수정일자, k사용검사일사용승인일 모두 연도와 월별 분리</li><li>▪ 층 : 지하, 0~25, 26~33, 34~49, 50~67, 68~ 그룹화</li></ul>		100575.2761
<ul style="list-style-type: none"><li>▪ hidden_size = 64, num_layers = 2, num_epochs = 5,</li><li>▪ 15층 이상이면 '고층여부' feature 추가</li><li>▪ ['k-전화번호', 'k-팩스번호'] drop, 'k-135초과' 컬럼 추가 후 nan 값을 0으로 대체</li></ul>	LSTM	<b>138768.7871</b>

# Various Models with Trial and Error (김소현)

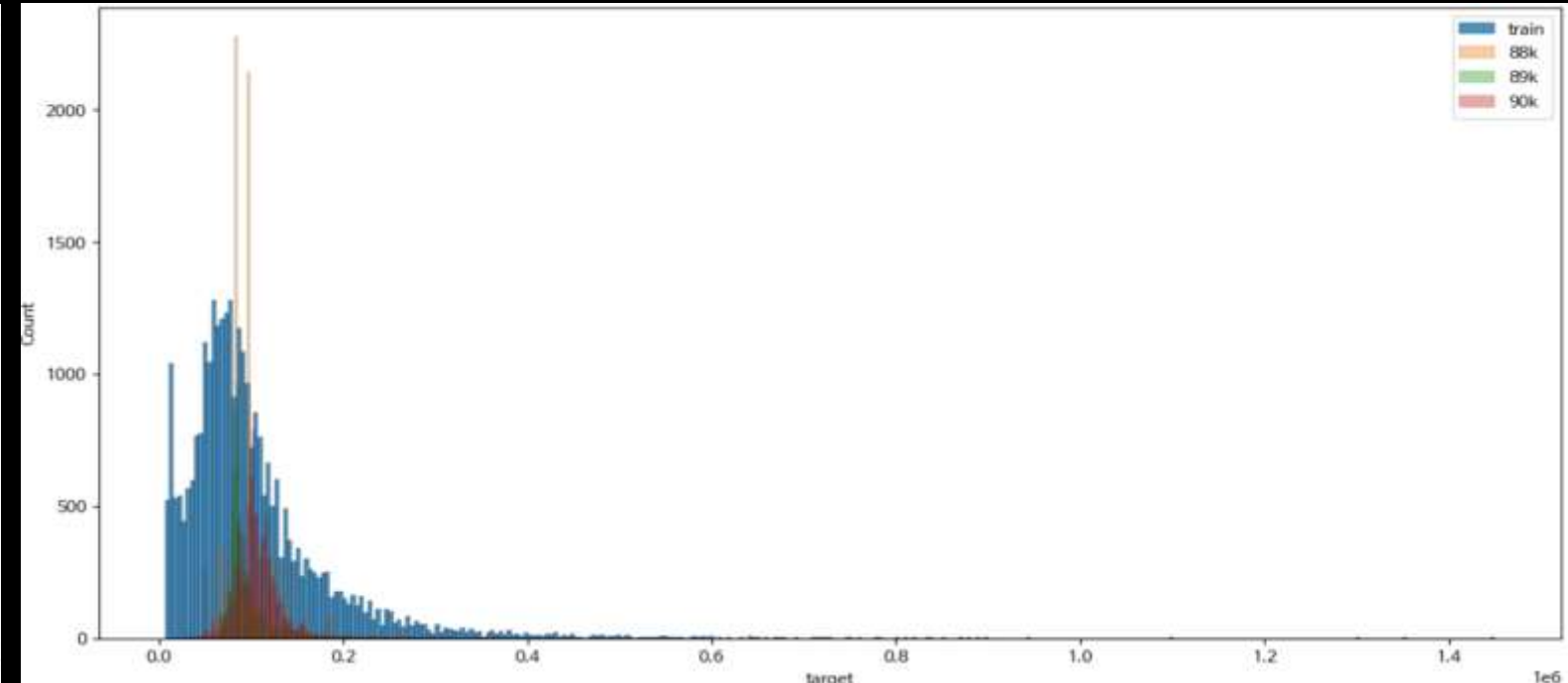
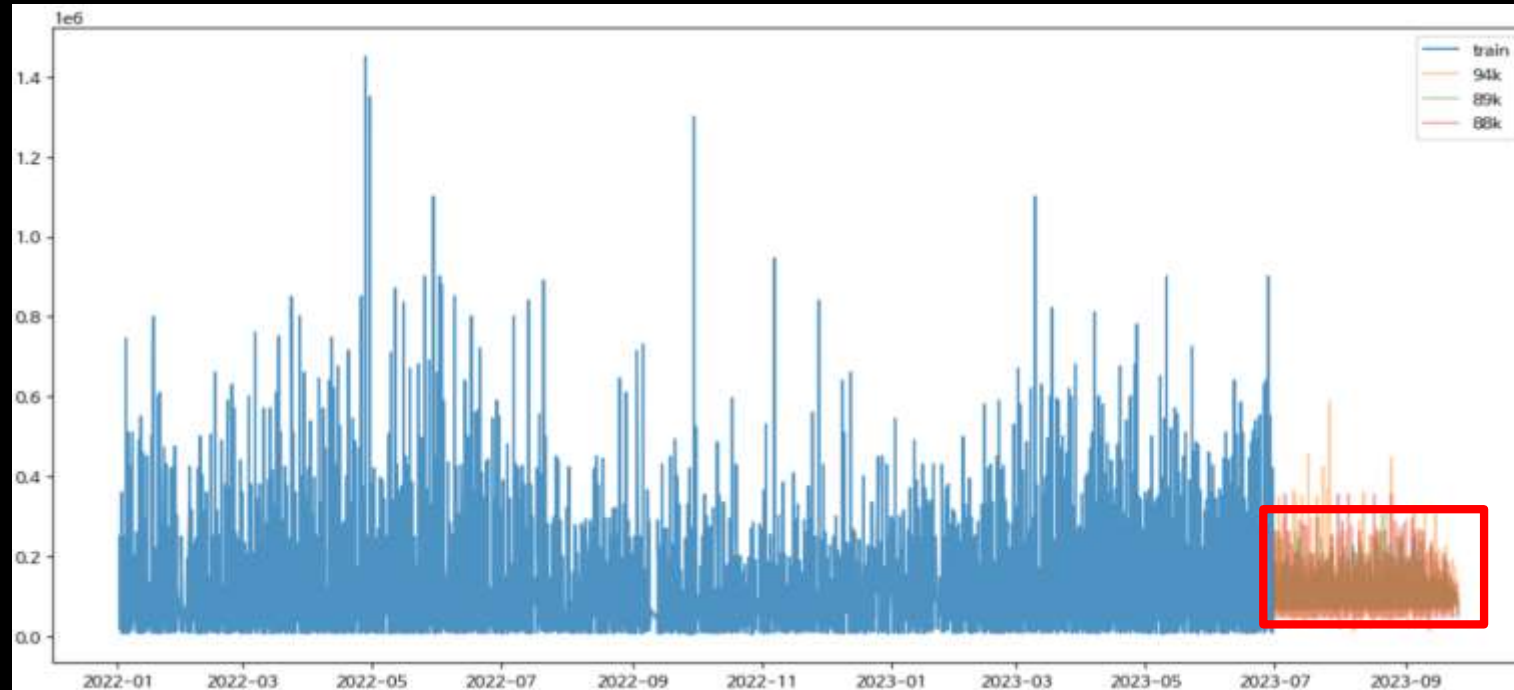
Feature Engineering	사용한 모델	Public Score
<ul style="list-style-type: none"><li>전용면적, 좌표X, 좌표Y</li><li>층 데이터 추가</li></ul>	LGBM	95047.1263
<ul style="list-style-type: none"><li>전용면적, 좌표X, 좌표Y</li><li>층 데이터 추가</li><li>target값 대신 층별 전용면적으로 groupby한 price 값 사용</li></ul>		97707.1264
<ul style="list-style-type: none"><li>전용면적, 좌표X, 좌표Y</li><li>좌표 X, 좌표 Y 모두 채움</li><li>price 대신 기존의 target 그대로 학습</li></ul>		102827.8155
<ul style="list-style-type: none"><li>전용면적, 좌표X, 좌표Y</li><li>target값 대신 층별 전용면적으로 groupby한 price 값 사용</li><li>(상대적으로 단순한 선형 모델로 시도)</li></ul>	Linear Regression	100366.3601

# Various Models with Trial and Error (권혁찬)

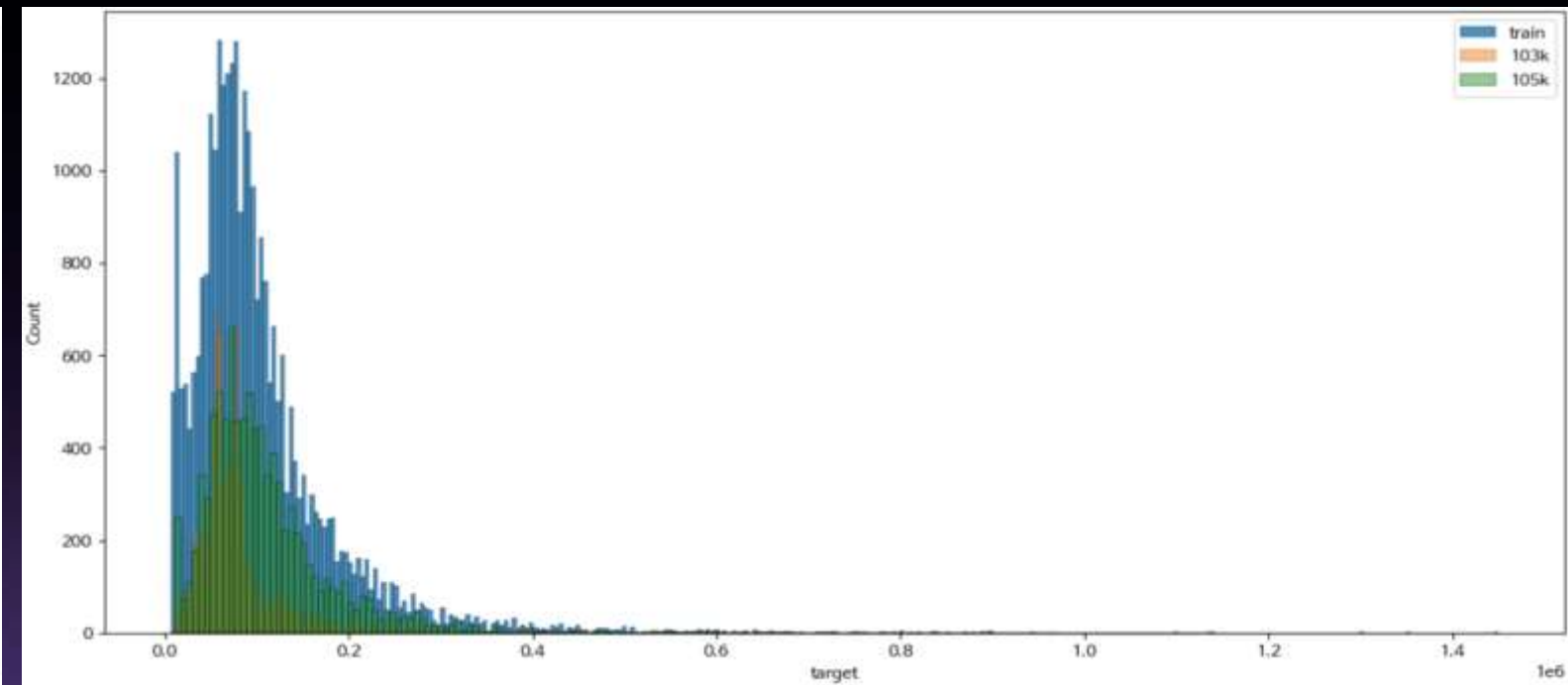
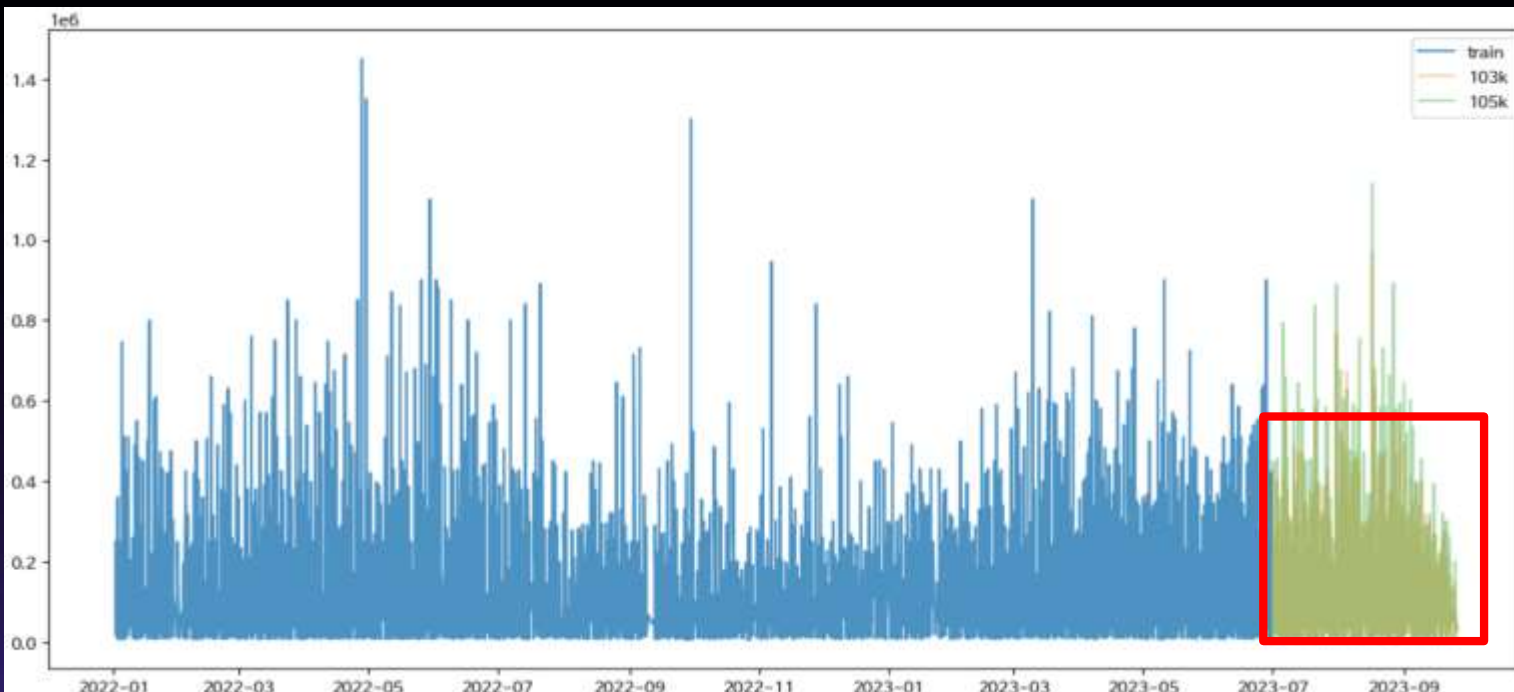
Feature Engineering	owner	사용한 모델	Public Score
<ul style="list-style-type: none"><li>▪ 사용피처: 전용면적, X좌표</li><li>▪ target 대신 price를 y로 사용</li></ul>	엄효범님	LGBM	88k
<ul style="list-style-type: none"><li>▪ Date : 2024.01.24 08:21:06</li><li>▪ Condition : 2021년 이후 데이터 대상, Target 수정 - Target(year_avg_price)와 year_max_price가 15%이상 차이 날 경우 year_max_price의 90% 값을 year_avg_price에 할당함</li><li>▪ Validation RMSE : 1102.4869</li><li>▪ Best Score: 1215477.4247</li></ul>	문정의님	LGBM	89k
<ul style="list-style-type: none"><li>▪ 엄효범님 94k 코드활용</li><li>▪ 새로운 피처 : GroupBy (전용면적,도로명)로 target의 mean과 median 생성</li><li>▪ 학습에 사용한 컬럼 : '전용면적(㎡)', '도로명', '좌표X', '좌표Y', 'target_mean', 'target_median'</li><li>▪ drop_duplicate 사용 단, keep='last'옵션 적용</li><li>▪ RMSE: 70803</li></ul>	권혁찬	LGBM	90k
<ul style="list-style-type: none"><li>▪ 2007년 전체 데이터 train에 사용</li><li>▪ 월별 아파트 면적에 따른 매매지수 사용: index_scale</li><li>▪ 학습에 사용한 컬럼 : '전용면적(㎡) // 5', '도로명', 'index_scale'</li><li>▪ drop_duplicate 사용</li><li>▪ RMSE: 52709</li></ul>	권혁찬	LGBM	103k
<ul style="list-style-type: none"><li>▪ 2007년 전체 데이터 train에 사용</li><li>▪ 월별 아파트 면적에 따른 매매지수 사용: index_scale</li><li>▪ train data : ratio(target / index_scale) 피처 생성 &amp; test data : ratio 보간</li><li>▪ 학습에 사용한 컬럼 : '전용면적(㎡)', '도로명', '동','구', index_scale' , ratio</li><li>▪ drop_duplicate 미적용</li><li>▪ RMSE: 9305(2022.01 이후 데이터)</li></ul>	권혁찬	LGBM	105k

# Various Models with Trial and Error (권혁찬)

- Public score 88k, 89k, 90k 모델 Test Prediction과 Train Target 데이터 분포 비교 (예측 분포가 좁음)



- Public score 103k, 105k 모델 Test Prediction과 Train Target 데이터 분포 비교 (예측 분포가 넓음)





# Various Models with Trial and Error (김태한)

Feature Engineering	사용한 모델	Public Score
<ul style="list-style-type: none"><li>▪ 관련이 없어 보이는 변수 'k-전화번호', 'k-팩스번호', 'k-관리방식', 'k-수정일자', '고용보험관리번호', '시군구', '계약년월' 삭제</li><li>▪ 결측치가 100만개 이상인 변수들 : ['해제사유발생일', '단지소개기존clob', 'k-135㎡초과', 'k-홈페이지', 'k-등록일자'] 삭제,</li><li>▪ 평수에 따라서 소형, 중형, 대형으로 분류한 컬럼 추가</li><li>▪ 평균면적 이상치 삭제 안함, 강남구 컬럼 추가 안함</li><li>▪ 나머지 전처리와 라벨 인코딩 등은 baseline과 동일</li><li>▪ Train, Val 8:2 Split. Valid RMSE: 7917</li></ul>	Random Forest	117223.1650
<ul style="list-style-type: none"><li>▪ 관련이 없어 보이는 변수 'k-전화번호', 'k-팩스번호', 'k-관리방식', 'k-수정일자', '고용보험관리번호', '시군구', '계약년월' 삭제</li><li>▪ 결측치가 100만개 이상인 변수들 : ['해제사유발생일', '단지소개기존clob', 'k-135㎡초과', 'k-홈페이지', 'k-등록일자'] 삭제</li><li>▪ 평수에 따라서 소형, 중형, 대형으로 분류한 컬럼 추가</li><li>▪ 이상치 삭제 안함, 강남구 컬럼 추가 안함</li><li>▪ 나머지 전처리와 라벨 인코딩 등은 baseline과 동일</li><li>▪ OOF Train RMSE: 2653.1172, OOF Val RMSE: 6762.8830</li></ul>	XGBoost with 5-Fold Optuna 튜닝	116961.4286
<ul style="list-style-type: none"><li>▪ 암호범님 코드 베이스</li><li>▪ 면적, X좌표, Y좌표, 계약년, 계약월, 해당 년의 평균가격, 해당 월의 평균가격</li><li>▪ 2020~2023년 데이터 사용</li><li>▪ Val RMSE 21784.6934</li></ul>	LGBM	112001.1583
<ul style="list-style-type: none"><li>▪ 전용면적, 층, 건설년도, 계약년도, 계약월, X, Y 좌표, 연도별 평균, 월별 평균, 구별 평균, 동별 평균, 도로별 평균 적용 예정</li><li>▪ 데이터는 모두 2020년대 데이터, Train은 2020, 2021, 2022 데이터, Val은 2023년 데이터</li><li>▪ 개별 가격 예측</li><li>▪ Train RMSE: 45188.4010, Val RMSE : 47689.8046</li></ul>		116057.0811

# Various Models with Trial and Error (김태한)

Feature Engineering	사용한 모델	Public Score
<ul style="list-style-type: none"><li>전용면적, 층, 건설년도, 계약년도, 계약월, X, Y 좌표, 연도별 평균, 월별 평균, 구별 평균, 동별 평균, 도로별 평균 적용 예정</li><li>데이터는 모두 2020년대 데이터, Train은 2020, 2021, 2022 데이터, Val은 2023년 데이터</li><li>개별 가격 예측</li><li>Train RMSE: 45188.4010 Val RMSE : 47689.8046</li></ul>	LGBM	116057.0811
<ul style="list-style-type: none"><li>엄효범님의 groupby에서 면적의 카테고리를 변경</li><li>면적을 소형, 증소형, 증형, 증대형, 대형, 초대형으로 나눠서 mean으로 target인 price 생성</li><li>년, 월, 일, 소수점만 반영한 x, y 좌표, 면적을 이용.</li><li>Train RMSE: 35518.5737, Val RMSE : 36488.0280</li></ul>		106043.4189

# Various Models with Trial and Error (문정의)

Feature Engineering	사용한 모델	Public Score
<ul style="list-style-type: none"><li>기존 Feature</li><li>Optuna 100회</li><li>Local RMSE : 4445.2855</li></ul>	LGBM	100976.8078
<ul style="list-style-type: none"><li>Feature 추가 - 브랜드</li><li>Optuna 100회</li><li>Local RMSE : 4187.1013</li></ul>		100026.1381
<ul style="list-style-type: none"><li>Feature제거 - 사용허가여부, 임대방식, 청소비관리형태, 관리비 업로드, k-세대타입, 경비비관리형태, k-관리방식, 세대전기계약방법</li><li>Local RMSE : 1594.9173 4</li></ul>		100042.2425
<ul style="list-style-type: none"><li>피쳐 추가 - 건물년수, subways, bus_stops</li><li>피쳐 보정 - k-전체동수, k-전체세대수, k-연면적, k-주거전용면적, k-관리비부과면적, k-전용면적별세대현황(60㎡이하), k-전용면적별세대현황(60㎡~85㎡이하), k-85㎡~135㎡이하, 건축면적, 주차대수</li><li>Local RMSE : 3942.3867</li></ul>		100064.5063
<ul style="list-style-type: none"><li>validation후에 오차가 많이 발생하는 Row 10000건 삭제</li><li>Local RMSE : 3774.0949</li></ul>		99682.1164
<ul style="list-style-type: none"><li>2021년 이후 데이터 대상</li><li>Target 수정 - year_avg_price 값을 Target으로 변경 (임효범님 idea 참조)</li><li>Local RMSE : 5372.2644</li></ul>		89077.1826
<ul style="list-style-type: none"><li>2021년 이후 데이터 대상</li><li>Target 수정 - Target(year_avg_price)와 year_max_price가 15%이상 차이 날 경우 year_max_price의 90% 값을 year_avg_price에 할당함</li><li>Local RMSE : 1102.4869</li></ul>		89609.5746



# Various Models with Trial and Error (이현진)

Feature Engineering	owner	사용한 모델	Public Score
<ul style="list-style-type: none"><li>▪ feature : '전용면적_m', '건축년도', 'y', 'addr'</li><li>▪ 좌표y를 외부 데이터로 보간, 도로명으로 group 후 target 평균</li><li>▪ local RMSE : 37367.538</li></ul>	이현진	<b>XGBoost</b>	<b>95280.8867</b>
<ul style="list-style-type: none"><li>▪ feture : 강남여부, 전용면적, 도로명, 전체동수,층,계약년, 금리</li><li>▪ 한국은행의 기준금리 feature 추가, 강남여부 feature 추가, 계약년월 중 계약년만 사용</li><li>▪ local RMSE : 21616.7339</li></ul>	이현진	XGBoost	99413.4124
<ul style="list-style-type: none"><li>▪ feature : 결측비율 80% 이하 columns</li><li>▪ 아파트명, 시군구를 target을 기준으로 ranking encoding</li><li>▪ local RMSE : 7006.8349</li></ul>	이현진	<b>LGBM</b>	<b>104787.0635</b>
<ul style="list-style-type: none"><li>▪ feature : feature importances 기준 가장 좋은 feature 13개 사용</li><li>▪ (optuna) 하이퍼파라미터 조정 (trial = 100)</li><li>▪ local RMSE : 5481.3258</li></ul>	이현진	LGBM	105371.2048
<ul style="list-style-type: none"><li>▪ feature : 결측비율 80% 이하 columns</li><li>▪ 파라미터 수정(n_estimators 증가)</li><li>▪ local RMSE : 3724.07</li></ul>	이현진	LGBM	107747.0060

# Hyperparameter Tuning

---

- 디폴트 하이퍼파라미터가 좋은 성능을 제공하여서 하이퍼파라미터 튜닝에 대한 Task 보다는 Feature를 추가 / 삭제 / 보정하는 Task에 시간을 할애하였음

강사님께 받은 피드백 및 의견을 정리해주세요.

---

**Q. 혹시 강사님께서 시계열 예측 대회를 하시면서 경험하셨을 때 트리계열 머신러닝 모델과 RNN 계열 딥러닝 모델 둘중 일반적으로 어느쪽이 성능이 더 좋았더라는 것이 있을까요?**

시계열과 크로스섹셔널 앙상블은 성능이 별로인 경우가 많습니다.

**Q. 시계열 모델과 트리계열 모델 중 어떤 것을 적용할지에 대한 선택을 어떻게 할까요?**

데이터에 seasonality, trend 같은 시계열 특성을 가지고 있는지에 대한 판단이 모델 적용 전에 필요합니다.  
팀원끼리 모델을 분담해서 실험해 봅시다.

**Q. 피쳐들 생성을 언제까지 확정하는게 좋은가요?**

feature들을 만지고 tuning하는 데에 시간을 대부분 소요합니다. submission 시간까지 고려해봅시다.

강사님께 받은 피드백 및 의견을 정리해주세요.

---

**Q. Prophet이나 Dart 같은 라이브러리를 조금 살펴보니까 시계열 예측 부분에서 상당히 많은 부분을 표준화해서 기능을 제공해주던데요. 데이터를 라이브러리에 잘 맞춰서 밀어넣어서 대회에서 높은 성능을 기록할 수도 있을까요? 일반적인 트리계열 모델을 자유롭게 모델링한 것과 비교해서요**

당연히 모델에 넣을 feature의 개수까지 고려하여 데이터에 맞는 모델을 적용해야 합니다.

스케일링 같은 경우, 시계열 모델은 많은 영향을 받으나 tree모델에는 영향이 거의 없으며,

같은 데이터로 tree에 넣고 시계열에 넣고 하면 보통 둘 중 하나는 성능이 낮습니다.






















































05

# 결과



# 최종 순위 및 평가지표 결과

- 최종 순위 : 8위, RMSE : 89,077, 제출횟수 : 73회

순위	팀 이름	팀 멤버	RMSE ↕	제출 횟수	최종 제출
8 (-)	ML 8조	    	89077.1826	73	2d
1	ML 6조	    	85238.1922	61	2d
2	ML 7조	    	86148.2275	76	12h
3	ML 1조	    	87035.7452	60	5d
4	ML 2조	    	88064.8558	32	5d
5	ML 4조	    	88924.2337	65	2d
6	ML_3조	    	88924.2337	38	2d
7	ML 9조	      	88924.2337	10	1d
8	ML 8조	    	89077.1826	73	2d
9	ML 5조	     	94528.5981	50	1d

06

# 그룹 스터디 진행 소감

## 그룹 스터디 진행 소감

### 김소현

데이터에 결측치도 많고 시계열로 해석 가능한 데이터까지 들어 있어서 여러 가지 모델 실험을 할 수 있던 데이터였던 것 같다. 도메인 지식이 부족하고, 생각보다 public score에서 내가 만든 모델들의 개선을 확인할 수 없어서 방향을 잡기가 어려웠다.

그래도 팀원분들과 이야기를 나누며 데이터를 시각화한 것을 관찰하고 잘 아는 분들께서 도메인 관련 지식들을 공유해 주셔서 많이 알아가는 기회가 되었다.

### 권혁찬

처음 데이터를 살펴봤을 때 컬럼이 상당히 많아서 모델링에 사용할 수 있는 데이터가 많다고 생각했다. 그런데 막상 결측치를 확인해보니 대다수의 데이터가 상당한 결측치를 포함하고 있어서 모델링하는데 사용하기 어려운 상태였다. 또한 막상 모델링을 함에 있어서 피처를 추가하고 하이퍼파라미터 튜닝을 할수록 스코어가 더 안나오는 느낌을 받았다. 아무래도 더 맞지 않는 쪽으로 모델이 더욱 피팅이 되고 있는 것 같았다. 결국 더하는 것 보다는 빼는 것으로 중점적으로 모델링을 했으나 원하는 만큼 스코어를 달성하지는 못한 것 같다. 그래도 매일 매일 팀원과 함께 같은 문제로 토론하면서 문제 방향성을 잡고 끈기있게 파고드는데 큰 도움이 된 것 같다.

## 그룹 스터디 진행 소감

---

### 김태한

Validation 데이터를 활용한 RMSE, 특히 K-Fold를 적용해도 **Public Score와 같은 방향으로 움직이지 않아서 어떻게 스코어를 올려야할지 감을 잡기 어려웠다.** 특히 여러 기사를 찾아봤을 때 2023년 하반기에도 아파트 가격이 1% 내외로 하락했기에 더욱 어려웠다. 유의미한 피쳐의 확인이 어려웠다. 각자 EDA와 아이디어를 공유하여 시야의 폭을 넓힐 수 있었다.

### 문정의

팀원들 간에 지속적인 회의를 통해서 아이디어를 공유하고, insight를 얻을 수 있어서 좋았으며, Target값을 바꾸어서 학습을 할 수 있다는 점을 새롭게 배웠음. 제출 횟수가 1인당 2회로 제한되어서 아이디어를 적용하고 확인하는데 제한이 있어서 아쉬웠습니다.

## 그룹 스터디 진행 소감

---

### 이현진

Data에 결측치가 많고, 아파트 거래에 대한 도메인 지식이 부족하다보니 EDA부터 Feature Selection까지 어려움이 컸다. 다양한 feature를 추가해보고, 외부 데이터를 이용해서 feature를 만들어보는 경험은 좋았으나, 이것이 성능의 개선으로 이루어지지 않는 아쉽다.

팀 프로젝트를 통해 데이터를 보는 눈이나, 추가적인 데이터 활용에 대한 insight를 얻을 수 있는 기회라고 생각한다.



# Q&A

---

—  
감사합니다.