

도메인 지식을 통한
아파트 실거래가 예측 분석

Table of Contents.

Part 1. 분석 전략

Part 2. 사용 데이터 및 EDA

Part 3. Feature Engineering

Part 4. Modeling

주거의 가치를 높이는
근린환경

1 아파트 가격에 영향을 미치는 근린환경 요인

물리적 근린 환경은 학교·공공 기관·의료 기관·상업 시설과 같이 생활에 편의를 제공하는 생활 환경을 말한다.

*아파트 가격에 영향을 미치는 근린환경 요인 분석(이소영, 김명연, 김은정)

2 인프라에 대한 사람들의 선호도

도심권 단지 중 반경 1km 주변에 교통, 쇼핑, 교육, 문화 등 생활 인프라가 고루 갖춰진 '*올인원 아파트' 인기

*역세권, 학세권, 물세권, 공세권 등이 통합된 아파트

이런 데이터들을 사용했습니다.

공원

제공 데이터

어린이집

제공 데이터, 서울 및 부산 어린이집 공공 데이터, 성동구 어린이집 공공 데이터

병원

서울 및 부산 병원 공공 데이터

약국

전국 약국 공공 데이터

학원

서울 학원 데이터, 부산 학원 및 교습소 공공 데이터

상권

서울 및 부산 상권 정보 공공 데이터

역세권

서울 및 부산 지하철 공공 데이터

데이터 출처

어린이집

서울시 어린이집 정보(표준 데이터) : <http://data.seoul.go.kr/dataList/OA-20300/S/1/datasetView.do>

서울시 성동구 어린이집 정보(표준 데이터) : <http://data.seoul.go.kr/dataList/OA-20304/S/1/datasetView.do>

전국어린이집표준데이터 : https://data.busan.go.kr/dataSet/detail.nm?contentId=10&publicdatapk=OA_SS00012

병원

서울시 병의원 위치 정보 : <http://data.seoul.go.kr/dataList/OA-20337/S/1/datasetView.do>

부산광역시지방행정인허가공개데이터 : https://bigdata.busan.go.kr/data/bigDataDetailView.do?menuCode=M00000000007&hdfs_file_sn=20210201050100939001

약국

전국 병의원 및 약국 현황 : <https://opendata.hira.or.kr/op/opc/selectOpenData.do?sno=11925&publDataTpCd=&searchCnd=&searchWrd=%EC%A0%84%EA%B5%AD&pageIndex=1>

데이터 출처

상권

소상공인시장진흥공단_상가(상권)정보 : <https://www.data.go.kr/data/15083033/fileData.do>

학원

서울시 학원 교습소 정보 : <http://data.seoul.go.kr/dataList/OA-20528/S/1/datasetView.do>

부산광역시교육청 학원 및 교습소 현황 : <https://www.data.go.kr/data/15015227/fileData.do>

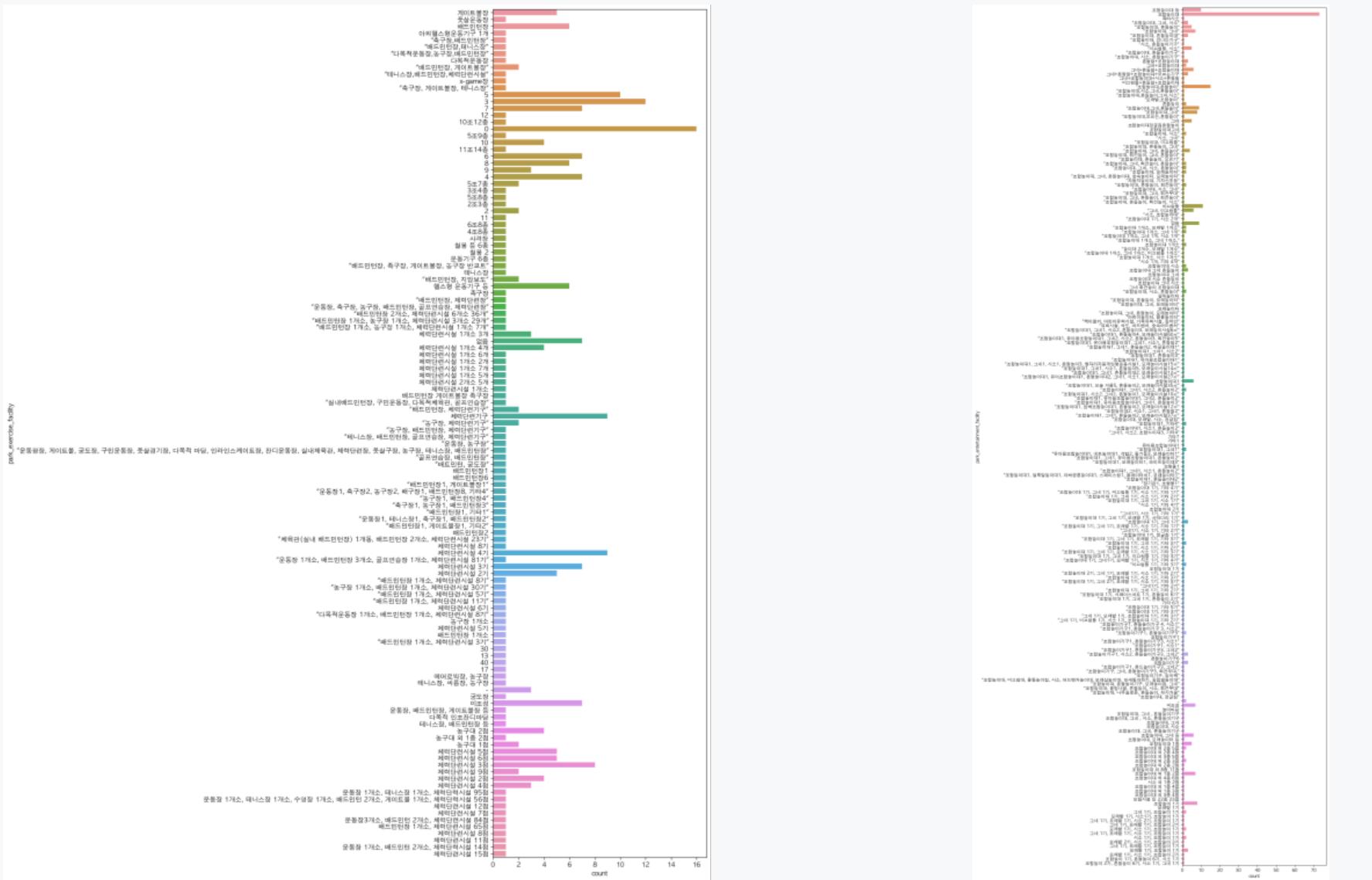
역세권

서울교통공사 지하철역 주소 및 전화번호 정보 : <http://data.seoul.go.kr/dataList/OA-12035/S/1/datasetView.do>

국가철도공단_부산_지하철_주소데이터 : <https://www.data.go.kr/data/15041101/fileData.do>

공원

공원의 각종 시설물 변수들은 전처리가 번거롭고 그만큼의 성능 향상도 기대하기 어려워 제거

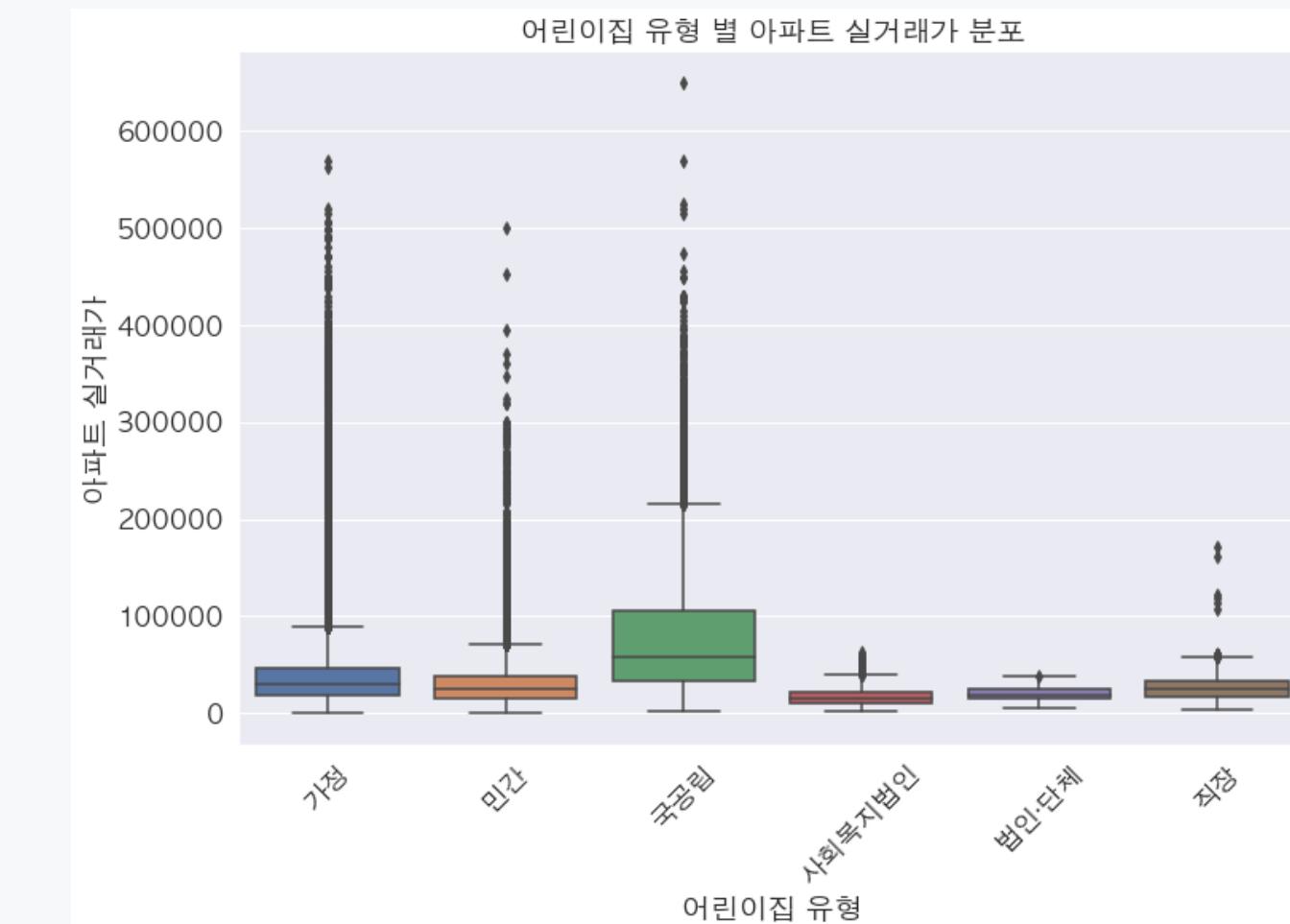
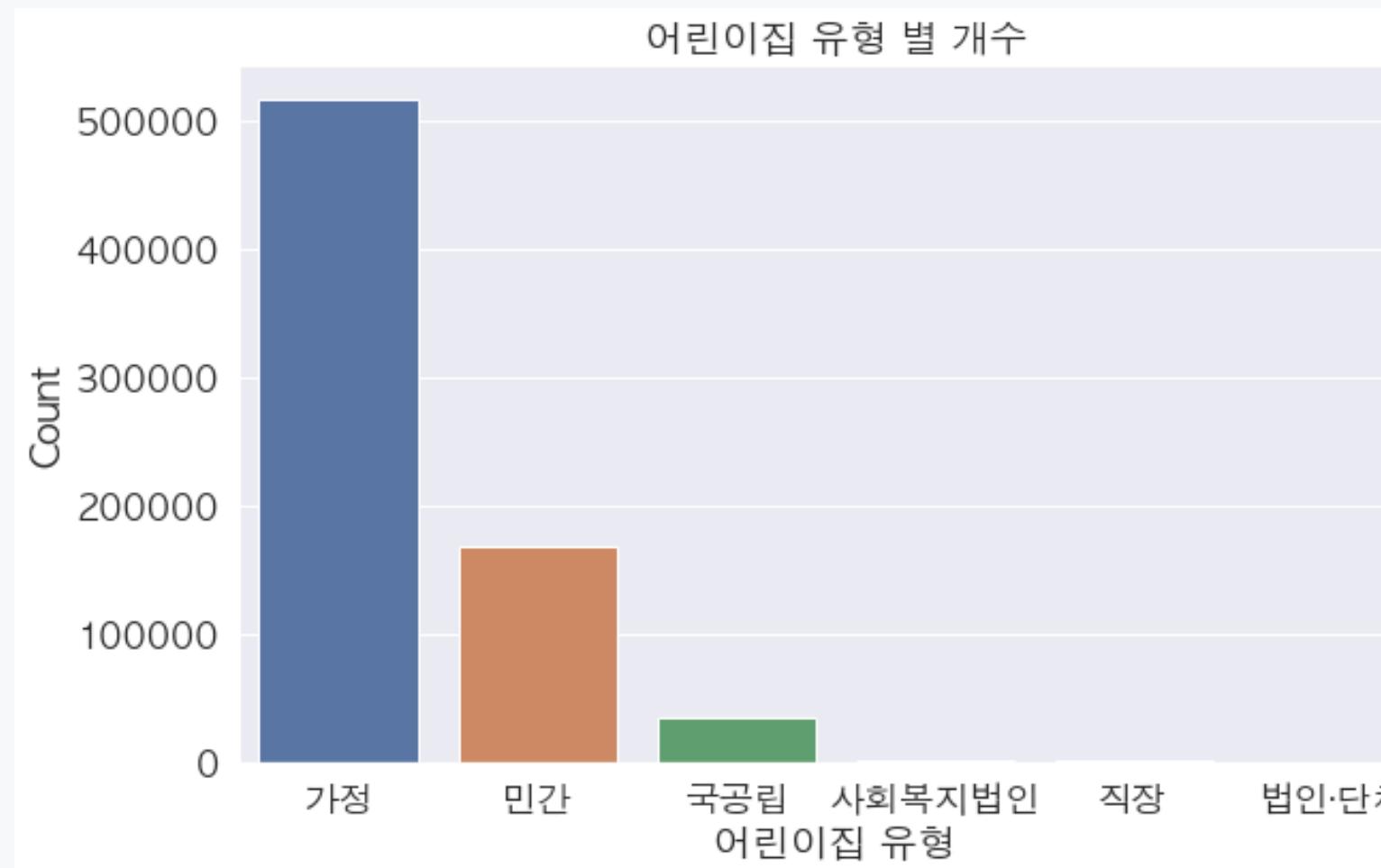


어린이집

어린이집의 유형에 따라 아파트 실거래가 분포에 차이 존재

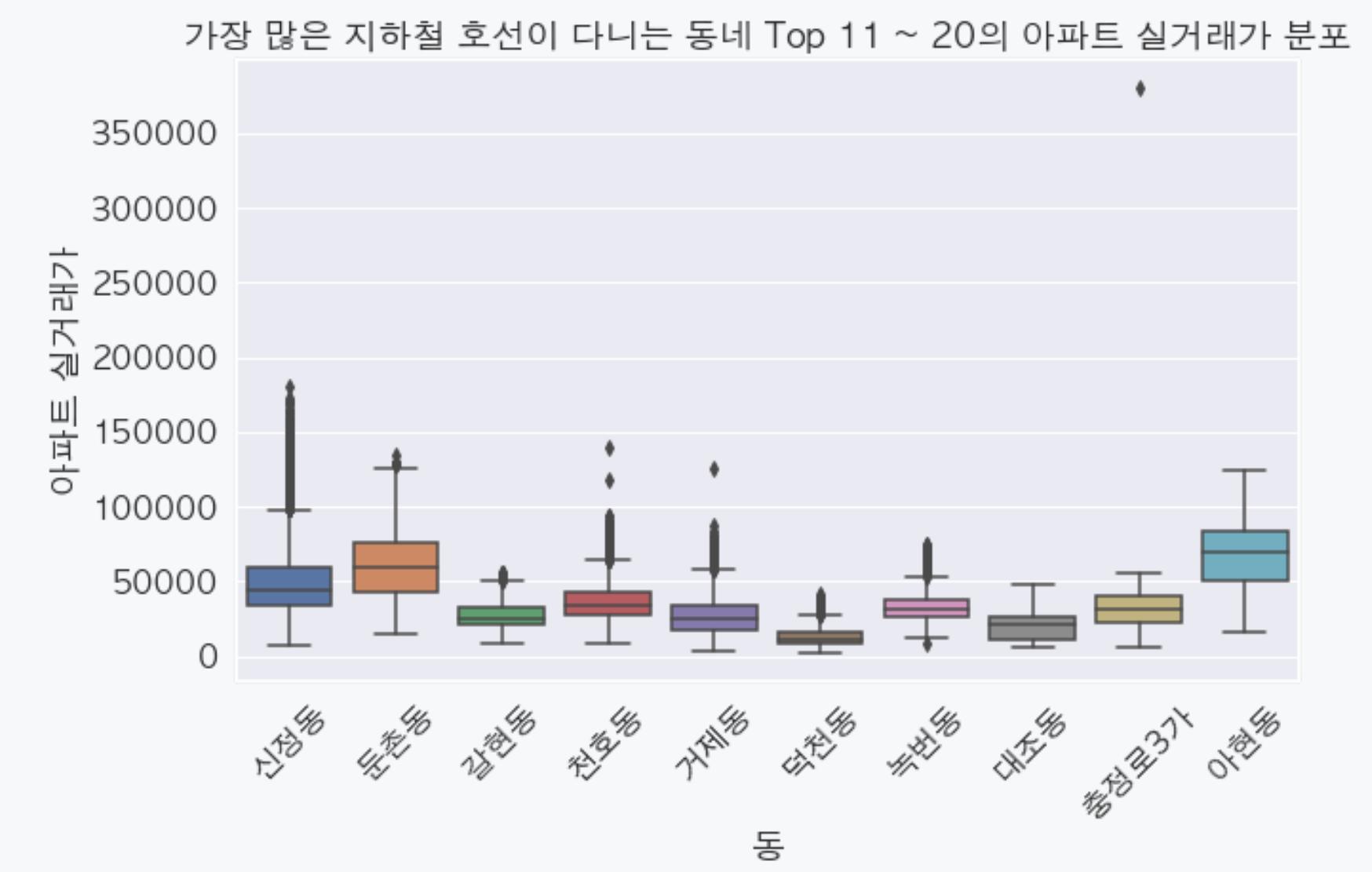
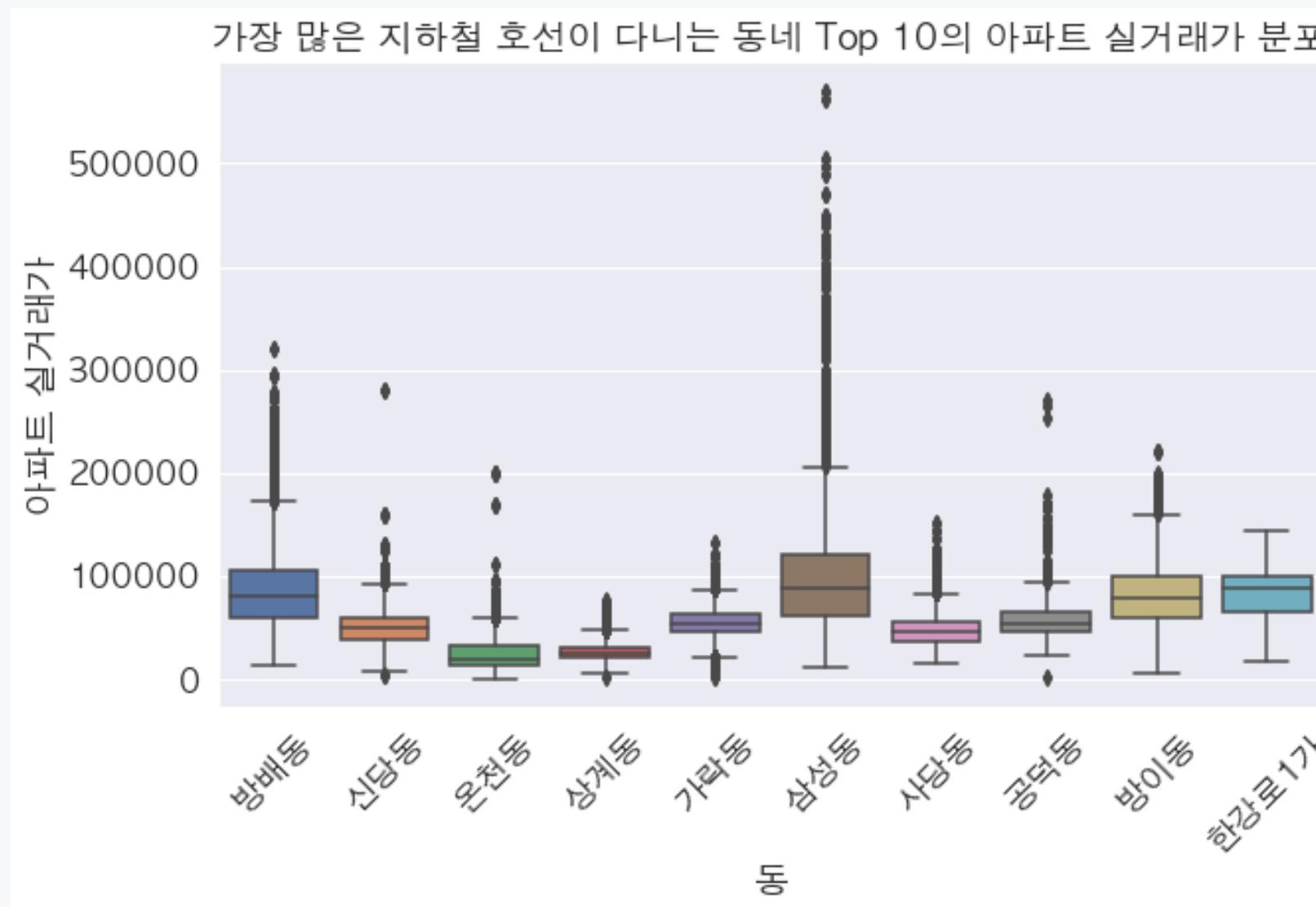
국공립 어린이집이 있는 아파트의 예비 청약자 경쟁률

*래미안 길음 센터피스 58대 1 / e편한세상 상도 노빌리티 19.1 대 1



역세권

다양한 호선이 다니는 동네일수록 실거래가가 높음



데이터 유형에 따라 Feature를 생성했습니다.

공원

동네 별 공원의 개수를 세어 수치형 변수로 생성 (dong_park_count)

어린이집

동네 별 어린이집의 개수를 세어 수치형 변수로 생성 (care_counts_count)

어린이집

동네에 있는 어린이집 중 가장 많은 어린이집 유형을 삽입해 범주형 변수로 생성 (major_care)

병원

동네 별 병원의 개수를 세어 수치형 변수로 생성 (hos_dongs_count)

약국

동네 별 약국의 개수를 세어 수치형 변수로 생성 (phar_dongs_count)

학원

동네 별 학원의 개수를 세어 수치형 변수로 생성 (aca_dongs_count)

데이터 유형에 따라 Feature를 생성했습니다.

상권

동네 별 상권의 개수를 세어 수치형 변수로 생성 (mar_counts_count)

상권

동네에 있는 상권 중 가장 많은 상권 업종 대분류를 삽입해 범주형 변수로 생성 (major_industry)

	submission.csv	Complete · shin yuin · 17d ago · RF : Validation Root Mean Squared Error (RMSE): 3717.55210545186	7811.98397	7937.04085	<input type="checkbox"/>	상권 X
	sub_3807.csv	Complete · shin yuin · 14d ago · val rmse : 3807 역세권, 병원, 학원, 공원, 약국, 상권	4306.42663	4512.33249	<input type="checkbox"/>	상권 O

역세권

동네 별 다니는 호선을 삽입해 범주형 변수로 생성 (호선)

역세권

동네에 다니는 호선의 전체 개수를 삽입해 수치형 변수로 생성 (count)

수치형

수치형 변수로 생성한 모든 변수들을 연산함수 'log'를 사용해 변수 생성

결측치를 처리했습니다.

- 수치형 변수의 경우 0, 범주형 변수의 경우 null 값으로 결측치 처리
- 찾을 수 있는 값의 경우 직접 찾아 결측치 처리

```

train_df['호선'] = train_df['호선'].fillna('null')
train_df['count'] = train_df['count'].fillna(0)
train_df['dong_park_count'] = train_df['dong_park_count'].fillna(0)
train_df['hos_dongs_count'] = train_df['hos_dongs_count'].fillna(0)
train_df['aca_dongs_count'] = train_df['aca_dongs_count'].fillna(0)
train_df['phar_counts_count'] = train_df['phar_counts_count'].fillna(0)
train_df['mar_counts_count'] = train_df['mar_counts_count'].fillna(0)
train_df['major_industry'] = train_df['major_industry'].fillna('null')
train_df['care_counts_count'] = train_df['care_counts_count'].fillna(0)
train_df['major_care'] = train_df['major_care'].fillna('null')

# test_df 열 처리
test_df['호선'] = test_df['호선'].fillna(0)
test_df['count'] = test_df['count'].fillna(0)
test_df['dong_park_count'] = test_df['dong_park_count'].fillna(0)
test_df['hos_dongs_count'] = test_df['hos_dongs_count'].fillna(0)
test_df['aca_dongs_count'] = test_df['aca_dongs_count'].fillna(0)
test_df['phar_counts_count'] = test_df['phar_counts_count'].fillna(0)
test_df['mar_counts_count'] = test_df['mar_counts_count'].fillna(0)
test_df['major_industry'] = test_df['major_industry'].fillna('null')
test_df['care_counts_count'] = test_df['care_counts_count'].fillna(0)
test_df['major_care'] = test_df['major_care'].fillna('null')

```

```

## 부산 ##
pu_subway['dong'] = pu_subway['지번주소'].str.extract(r'\b(\w+)동\b', expand=False) # 주소 혼용 데이터에서 동만 추출
pu_subway['dong'] = pu_subway['dong'] + '동'
pu_subway.loc[4, 'dong'] = '남포동1가'
pu_subway.loc[11, 'dong'] = '동대신동2가'
pu_subway.loc[24, 'dong'] = '서대신동2가'
pu_subway.loc[32, 'dong'] = '남포동6가'
pu_subway.loc[36, 'dong'] = '중앙동4가'
pu_subway.loc[38, 'dong'] = '토성동3가'
pu_subway.loc[50, 'dong'] = '물금읍 범어리'
pu_subway.loc[65, 'dong'] = '물금읍 범어리'
pu_subway.loc[78, 'dong'] = '물금읍 증산리'
pu_subway.loc[81, 'dong'] = '동면 가산리'
pu_subway.loc[100, 'dong'] = '철마면 고촌리'
pu_subway.loc[110, 'dong'] = '철마면 안평리' |

```

```

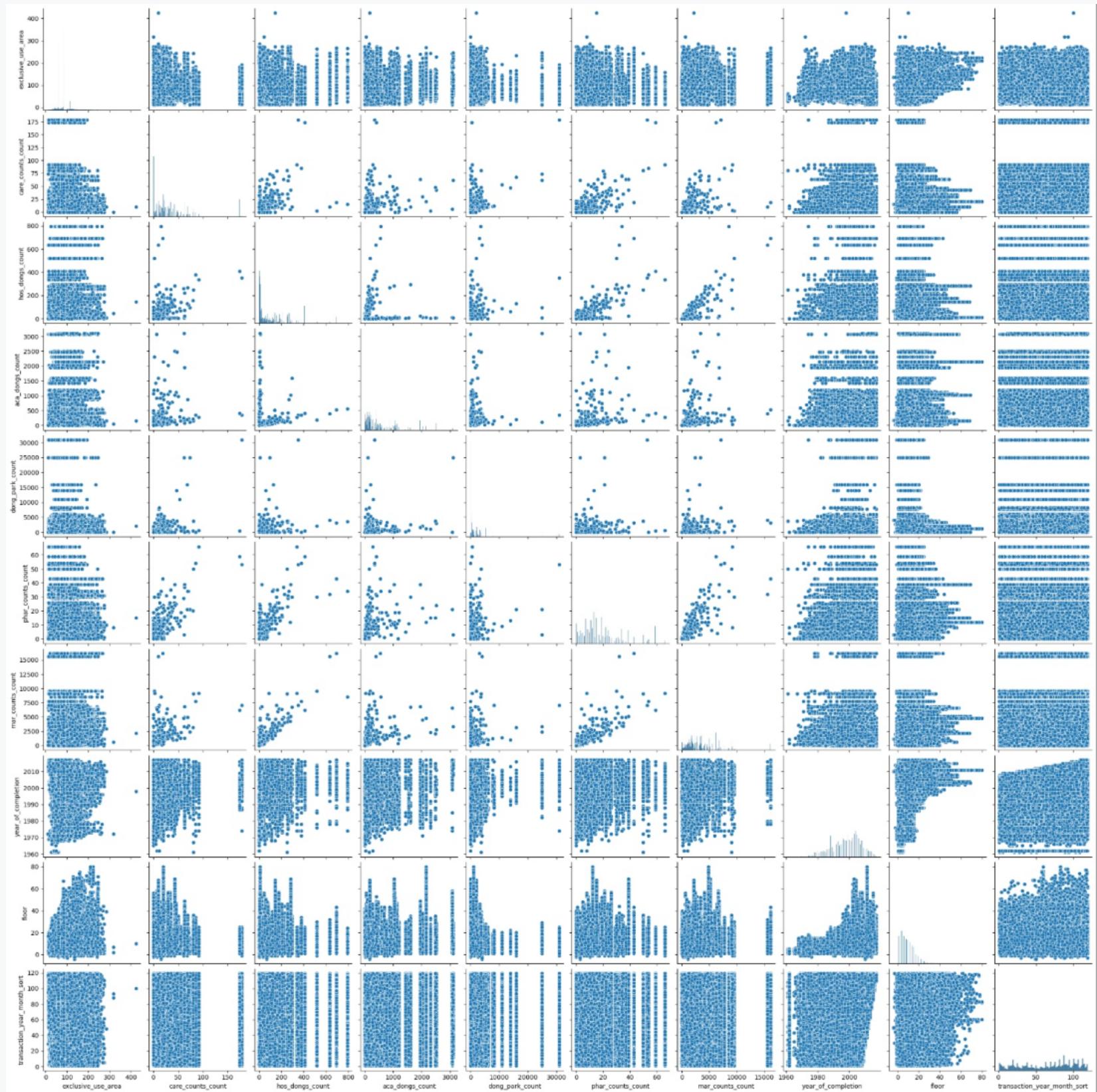
park_df.at[565, 'gu'] = '중구'
park_df.at[566, 'gu'] = '부산진구'
park_df.at[567, 'gu'] = '동래구'

```

Part 3. Feature Engineering

아파트 실거래가 예측 분석

전체 데이터의 상관관계는 다음과 같습니다.



데이터 유형에 따라 Feature Encoding을 진행했습니다.

```
## 특성별 피처
binary_features = ['city']

ordinal_features = ['count']

categorical_features = ['dong', '호선', 'major_industry', 'major_care']
```

```
column_transformer = ColumnTransformer(
    transformers=[
        ('cat', TargetEncoder(), categorical_features),
        ('bin', BinaryEncoder(), binary_features),
        ('ord', OneHotEncoder(), ordinal_features)
    ],
    remainder='passthrough'
)
```

- Binary_features ; Binary Encoding

city 변수는 서울특별시, 부산광역시로 구성되어 있어
Binary Encoding을 적용했습니다.

- Ordinal_features ; OneHot Encoding

동별 호선의 개수를 범주로 취급하여 OneHot Encoding 진행

```
train_df['count'].value_counts()
1.0    373426
2.0    290048
0.0    251837
3.0    136410
4.0     8854
Name: count, dtype: int64
```

- Categorical_features ; Target Encoding

해당 변수들은 범주 간의 차이가 존재하기에 해당 차이를 보존
하기 위하여 Target Encoding을 적용했습니다.

두 가지 모델을 사용하여 실험하였습니다.



1st Test



2nd Test

1st Test: Random Forest Regressor

- Random Forest Regressor

여러 개의 결정 트리(Decision Tree)를 조합하여 높은 성능과 일반화 능력을 제공하는 모델입니다.

- Random Forest 선택 이유

사용한 여러 도메인 데이터들을 바탕으로 범주형, 수치형 변수들을 다양하게 생성하였습니다. 따라서, 어떤 유형의 변수든 원활하게 처리 가능한 모델을 선택하고자 했습니다.
또한, 다양한 데이터를 추가하여 발생할 수 있는 과적합을 줄이고 일반화 능력을 향상시키기 위해 Random Forest를 선택했습니다.

```
## train_test_split
from sklearn.model_selection import train_test_split

X = train_df

column_transformer = ColumnTransformer(
    transformers=[
        ('cat', TargetEncoder(), categorical_features),
        ('bin', BinaryEncoder(), binary_features),
        ('ord', OneHotEncoder(), ordinal_features)
    ],
    remainder='passthrough'
)

preprocessor = Pipeline(
    steps=[
        ("column", column_transformer),
        ("selector", SelectPercentile(percentile=98)),
    ]
)
model = Pipeline(
    steps=[
        ("preprocessor", preprocessor),
        ("regressor", RandomForestRegressor(
            n_estimators=100,           #
            max_depth=None,
            min_samples_split=2,
            min_samples_leaf=1,
            max_leaf_nodes=None,
            min_impurity_decrease=0.0,
            n_jobs=-1,                 #
            random_state=42,            #
            warm_start=True,             #
            ccp_alpha=0.0
        )),
    ]
)
```

1st Test: Random Forest Regressor

```
preprocessor = Pipeline(  
    steps=[  
        ("column", column_transformer),  
        ("selector", SelectPercentile(percentile=98)),  
    ]  
)  
model = Pipeline(  
    steps=[  
        ("preprocessor", preprocessor),  
        ("regressor", RandomForestRegressor(  
            n_estimators=100,  
            max_depth=None,  
            min_samples_split=2,  
            min_samples_leaf=1,  
            max_leaf_nodes=None,  
            min_impurity_decrease=0.0,  
            n_jobs=-1, #  
            random_state=42, #  
            warm_start=True,  
            ccp_alpha=0.0  
        )  
    ],  
)
```

- 모델링 방법

하이퍼파라미터 값을 편리하게 조절하기 위하여 파이프라인을 사용해 모델을 구성하였습니다.

- RMSE 성능

CV mean : 4730.94

Public Score: 4245.44399

Private Score: **4116.6972**

2nd Test: Extra Trees Regressor

- Extra Trees Regressor

"Extremely Randomized Trees"로도 불리는 결정 트리를 기반으로 하는 양상을 학습 방법입니다.

- Random Forest와의 차이점

트리를 생성할 때 각 노드에서 랜덤하게 특성을 선택하고 최적의 분할이 아닌 랜덤 분할을 수행합니다.

따라서, 더 다양한 모델들을 생성하며 과적합을 줄일 수 있습니다.

```
## train_test_split
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesRegressor

X = train_df

column_transformer = ColumnTransformer(
    transformers=[
        ('cat', TargetEncoder(), categorical_features),
        ('bin', BinaryEncoder(), binary_features),
        ('ord', OneHotEncoder(), ordinal_features)
    ],
    remainder='passthrough'
)

preprocessor = Pipeline(
    steps=[
        ("column", column_transformer),
        ("selector", SelectPercentile(percentile=98)),
    ]
)
model = Pipeline(
    steps=[
        ("preprocessor", preprocessor),
        ("regressor", ExtraTreesRegressor()),
    ]
)
```

2nd Test: Extra Trees Regressor

```
preprocessor = Pipeline(  
    steps=[  
        ("column", column_transformer),  
        ("selector", SelectPercentile(percentile=98)),  
    ]  
)  
model = Pipeline(  
    steps=[  
        ("preprocessor", preprocessor),  
        ("regressor", ExtraTreesRegressor()),  
    ]  
)
```

- 모델링 방법

별 다른 값 조정 없이 단일 모델을 파이프라인에 적용하여 모델을 구성하였습니다.

- RMSE 성능

CV mean : 4576.44

Public Score: 4070.42388

Private Score: **3989.82076**



1st Test

CV mean : 4730.94

Public Score: 4245.44399

Private Score: 4116.6972



2nd Test

CV mean : 4576.44

Public Score: 4070.42388

Private Score: 3989.8207

감사합니다!

Machine Learning