DACON
아파트 실거래가 예측

3조

김명준 진경두 박두환 김민 손휘용

Contents.

Part 1

외부데이터 소개

Part 1

외부데이터 소개

데이터청년캠퍼스 3조

1. 주택규모 및 매매지수
2. 대치동 인구밀도
3. 초등학교 거리

Part 1

외부데이터 소개

데이터청년캠퍼스 3조

4 금리 데이터

5 전세 가격

6 시군구별 지가변동률

Part 1

주택규모 및 매매지수

주택규모에 따른 매매지수 데이터 - 국가통계포털(KOSIS)

수도권 아파트 주택규모별 매매지수 가져오기

[8]:
```
korea_price = pd.read_csv('/project/datacamp/team3/t3user1/newapart.csv',encoding = 'cp949')
korea_price
```

[8]:

	주택유형별	지역별	주택규모별	시점	데이터
0	아파트	수도권	규모1	2017.11	90.0
1	아파트	수도권	규모1	2017.12	90.2
2	아파트	수도권	규모1	2018.01	90.4
3	아파트	수도권	규모1	2018.02	90.7
4	아파트	수도권	규모1	2018.03	90.9
...	...	...	...	...	...
633	아파트	수도권	규모6	2023.02	98.8
634	아파트	수도권	규모6	2023.03	97.9
635	아파트	수도권	규모6	2023.04	97.5
636	아파트	수도권	규모6	2023.05	97.4
637	아파트	수도권	규모6	2023.06	97.5

638 rows × 5 columns

3) 주택규모별

2017년 11월 소형주택에 대한 규모별 지표 세분화

아파트 : 전용면적 기준, 규모1) 40㎡이하, 규모2) 40㎡초과~60㎡이하, 규모3) 60㎡초과~85㎡이하, 규모4) 85㎡초과~102㎡이하,규모5) 102㎡초과~135㎡이하, 규모6) 135㎡초과

연립다세대 : 전용면적 기준, 규모1) 40㎡이하, 규모2) 40㎡초과~60㎡이하, 규모3) 60㎡초과~85㎡이하, 규모4) 85㎡초과

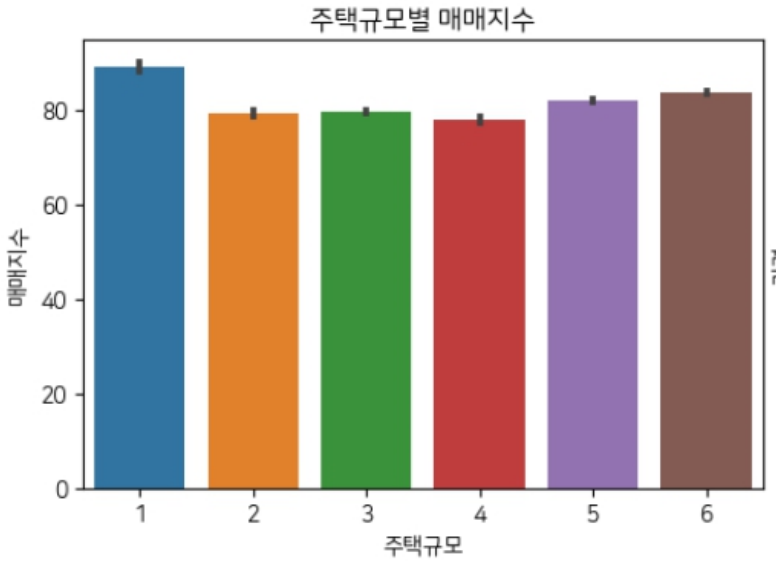
단독주택 : 규모1) 전용면적99㎡이하 & 대지면적231㎡이하, 규모2) 전용면적99㎡초과 331이하 & 대지면적662㎡이하 또는 전용면적99㎡이하 & 대지면적231㎡초과 662㎡이하, 규모3) 전용면적331㎡초과 또는 대지면적662㎡초과

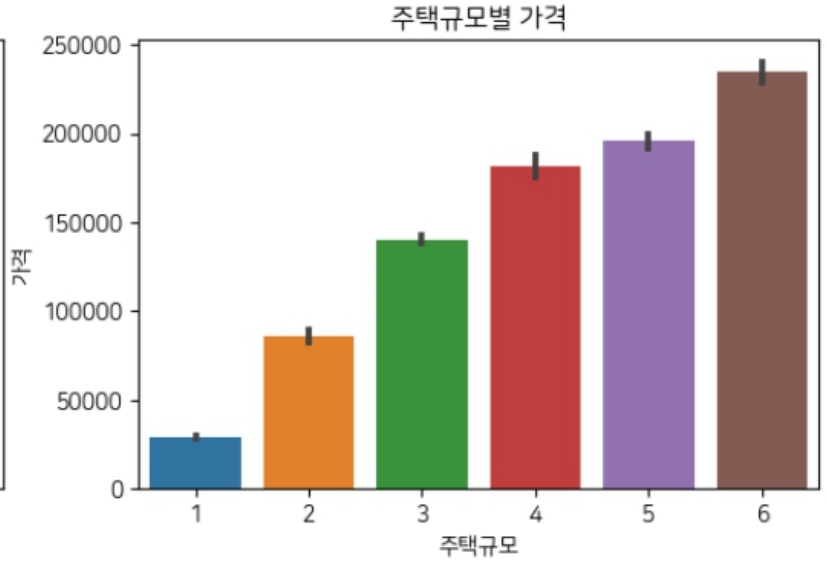
Part 1

주택규모 및 매매지수

주택규모에 따른 매매지수 데이터 - 국가통계포털(KOSIS)

매매지수는 동향을 확인하기 위함으로 가격과 단순 수치 비교는 불가능



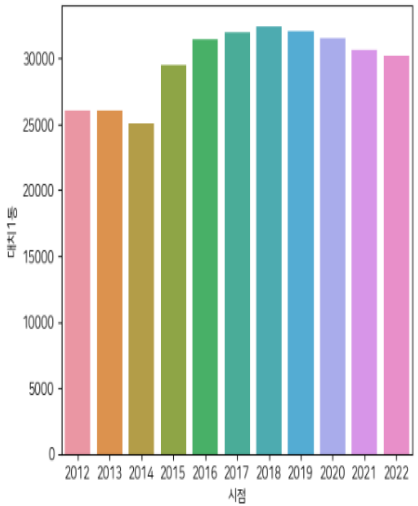


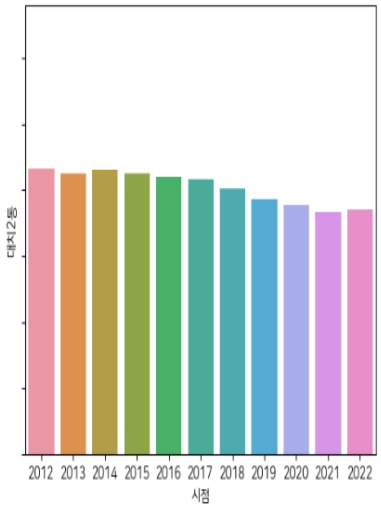
Part 1

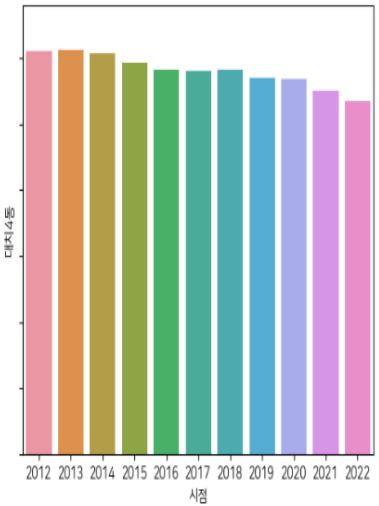
대치동 인구밀도

대치동 인구밀도 데이터 - 공공데이터포털

시점	항목	시점	대치1동	대치4동	대치2동	대치동
시점						
2012	인구밀도 (명/㎢)	2012	26080	30568	21597	78245
2013	인구밀도 (명/㎢)	2013	26051	30618	21255	77924
2014	인구밀도 (명/㎢)	2014	25068	30321	21559	76948
2015	인구밀도 (명/㎢)	2015	29472	29630	21283	80385
2016	인구밀도 (명/㎢)	2016	31435	29167	21034	81636
2017	인구밀도 (명/㎢)	2017	31977	29032	20783	81792
2018	인구밀도 (명/㎢)	2018	32392	29130	20082	81604
2019	인구밀도 (명/㎢)	2019	32028	28523	19324	79875
2020	인구밀도 (명/㎢)	2020	31547	28416	18876	78839
2021	인구밀도 (명/㎢)	2021	30632	27505	18332	76469
2022	인구밀도 (명/㎢)	2022	30185	26700	18549	75434







금리 데이터

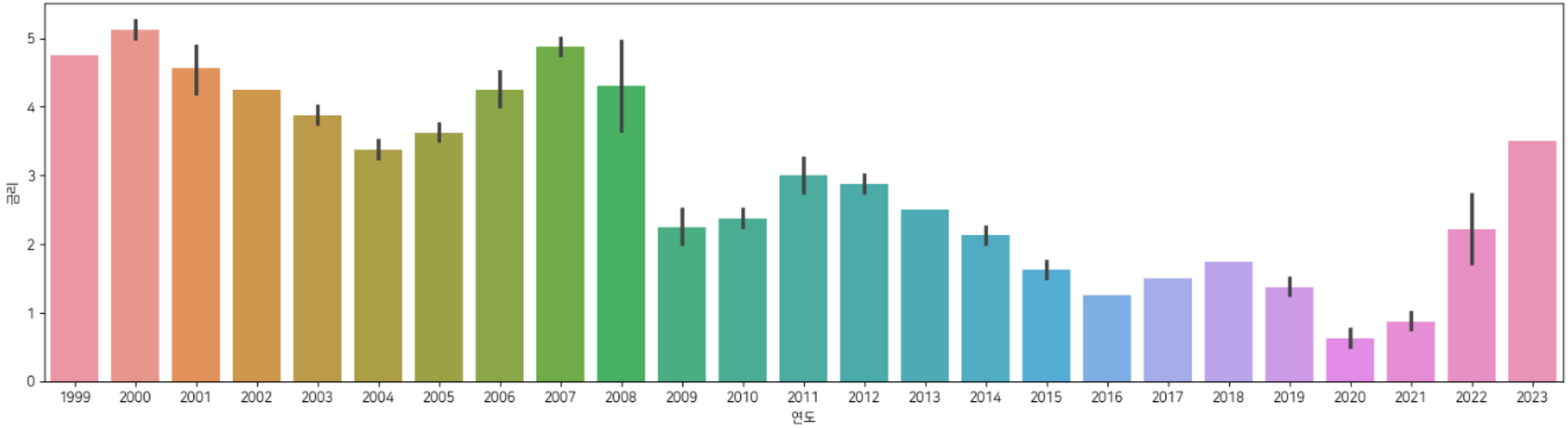
```
def make_date(row):
    month_day = row['월일'].replace('월 ', '-')
    month_day = month_day.replace('일', '')
    date = str(row['연도'])+ '-' + month_day
    return date

interest_rate['날짜'] = interest_rate.apply(lambda x: make_date(x), axis=1)
interest_rate['날짜'] = pd.to_datetime(interest_rate['날짜'])

for idx, row in tqdm(all_data.iterrows(), total = all_data.shape[0]):
    date = row['transaction_date']
    rate = interest_rate[interest_rate['날짜'] <= date].iloc[0]['금리']
    all_data.loc[idx, 'interest_rate'] = rate
```

	연도	월일	금리	날짜
0	2023	01월 13일	3.50	2023-01-13
1	2022	11월 24일	3.25	2022-11-24
2	2022	10월 12일	3.00	2022-10-12
3	2022	08월 25일	2.50	2022-08-25
4	2022	07월 13일	2.25	2022-07-13
5	2022	05월 26일	1.75	2022-05-26
6	2022	04월 14일	1.50	2022-04-14
7	2022	01월 14일	1.25	2022-01-14
8	2021	11월 25일	1.00	2021-11-25
9	2021	08월 26일	0.75	2021-08-26

Part 1

금리 데이터

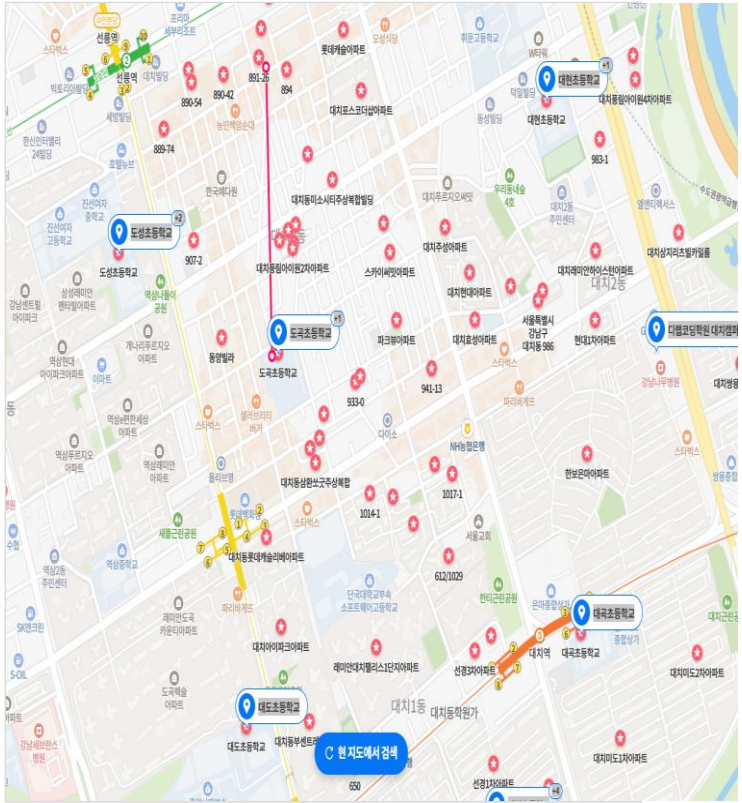
연도 별 금리 변화율



Part 1

Element school distance(m)

아파트 별 근접한 초등학교의 거리 (지도상 지번기준)



	jibun	element school distance(m)
0	670	165
1	986-14	370
2	988-4	460
3	511	250
4	891-6	650
...	...	...
5982	316	330
5983	316	330
5984	316	330
5985	902	590
5986	316	330

전세 데이터

대치동 전세 데이터 - 서울특별시 열린 데이터 광장

서울시 부동산 전월세가 데이터에서 대치동 전세 가져오기

```
junse_price = pd.read_csv('/project/datacamp/team3/t3user1/서울시 부동산 전월세가 정보.csv',encoding='cp949')
```

```
/tmp/ipykernel_702646/2207834238.py:1: DtypeWarning: Columns (18,19,20) have mixed types. Specify dtype option on import or set low_memory=False.
  junse_price = pd.read_csv('/project/datacamp/team3/t3user1/서울시 부동산 전월세가 정보.csv',encoding='cp949')
```

```
junse_price.info()
```

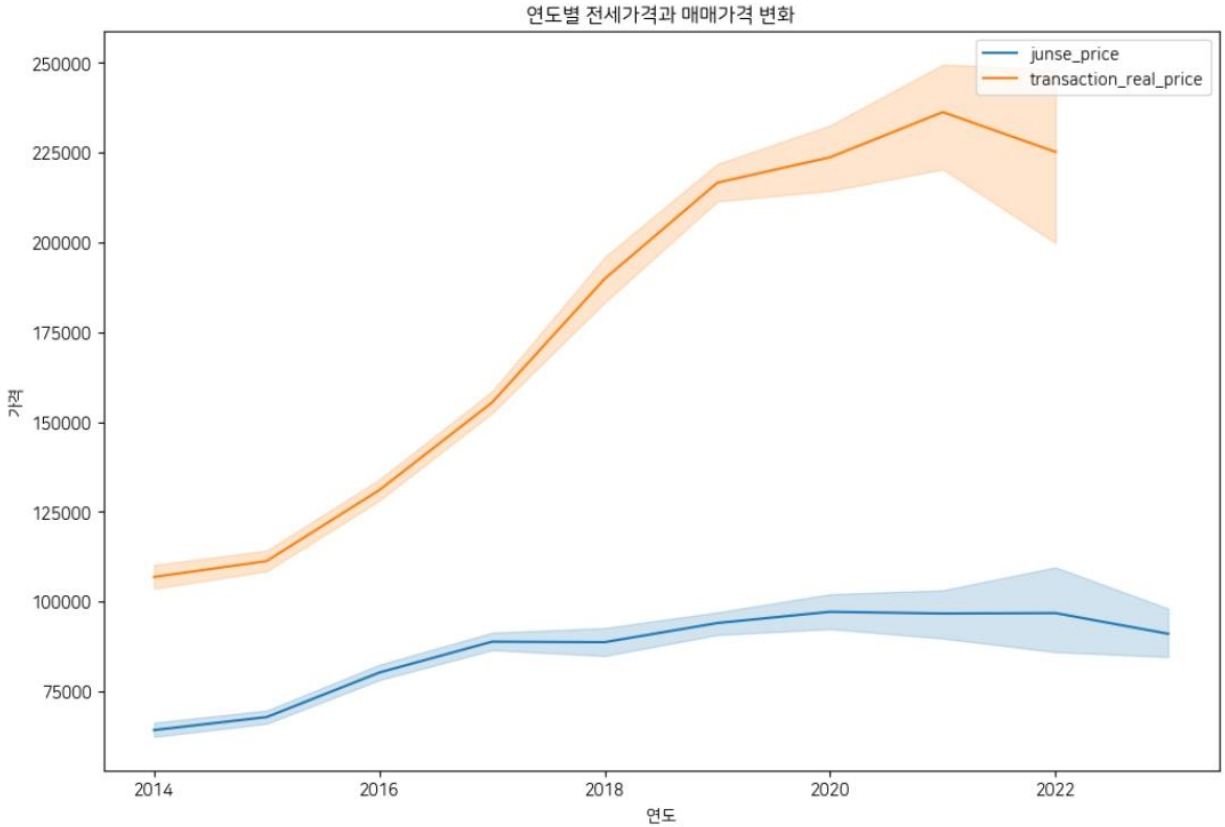
•••

```
junse_price = junse_price[(junse_price['법정동명']=='대치동')&(junse_price['건물용도']=='아파트')&(junse_price['전월세 구분']=='전세')]
junse_price = junse_price[junse_price['접수연도'] >= 2014]
junse_price = junse_price[['접수연도','본번','층','임대면적(㎡)','건물명','보증금(만원)']]
junse_price['층'] = junse_price['층'].astype('int')
```

Part 1

전세 데이터

대치동 전세 데이터 - 서울특별시 열린 데이터 광장



Part 1

시군구별 지가변동률

시군구별로 월마다 지가변동률 데이터 - 국가통계포털(KOSIS)

	시군구별(1)	시군구별(2)	2014.01	2014.02	2014.03	2014.04	2014.05	2014.06	2014.07	2014.08	...	2022.08	2022.09	2022.10	2022.11	2022.12	2023.01	2023.02	2023.03	2023.04	2023.05
0	시군구별 (1)	시군구별 (2)	평균	평균	평균	평균	평균	평균	평균	평균	...	평균	평균	평균	평균	평균	평균	평균	평균	평균	평균
1	서울특별시	강남구	0.211	0.300	0.510	0.311	0.211	0.283	0.276	0.283	...	0.312	0.255	0.100	0.030	0.021	0.054	0.064	0.102	0.093	0.248

2 rows × 115 columns

```
percent.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Columns: 115 entries, 시군구별(1) to 2023.05
dtypes: object(115)
memory usage: 1.9+ KB
```

```
percent.columns
```

```
Index(['시군구별(1)', '시군구별(2)', '2014.01', '2014.02', '2014.03', '2014.04',
       '2014.05', '2014.06', '2014.07', '2014.08',
       ...
       '2022.08', '2022.09', '2022.10', '2022.11', '2022.12', '2023.01',
       '2023.02', '2023.03', '2023.04', '2023.05'],
      dtype='object', length=115)
```

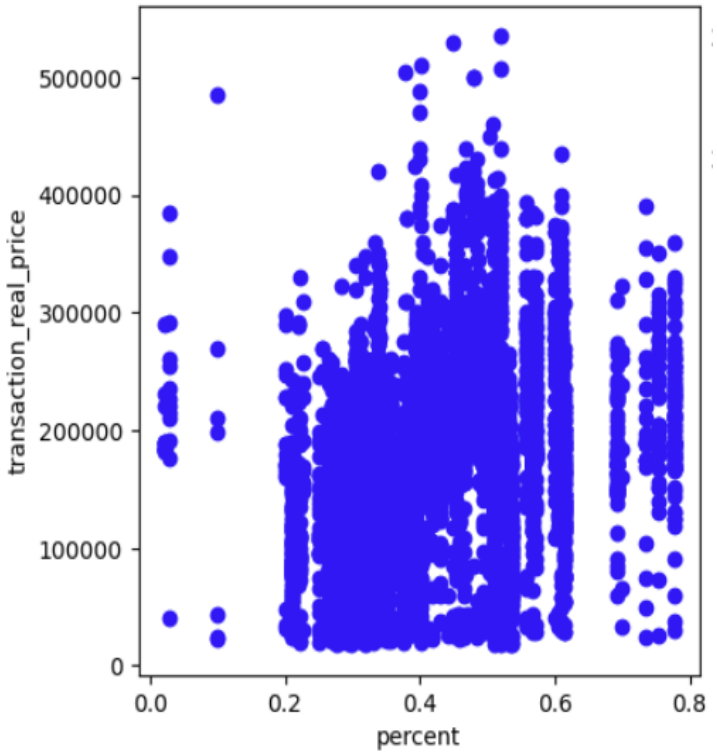
```
all_data[['transaction_year_month','percent']].head(10)
```

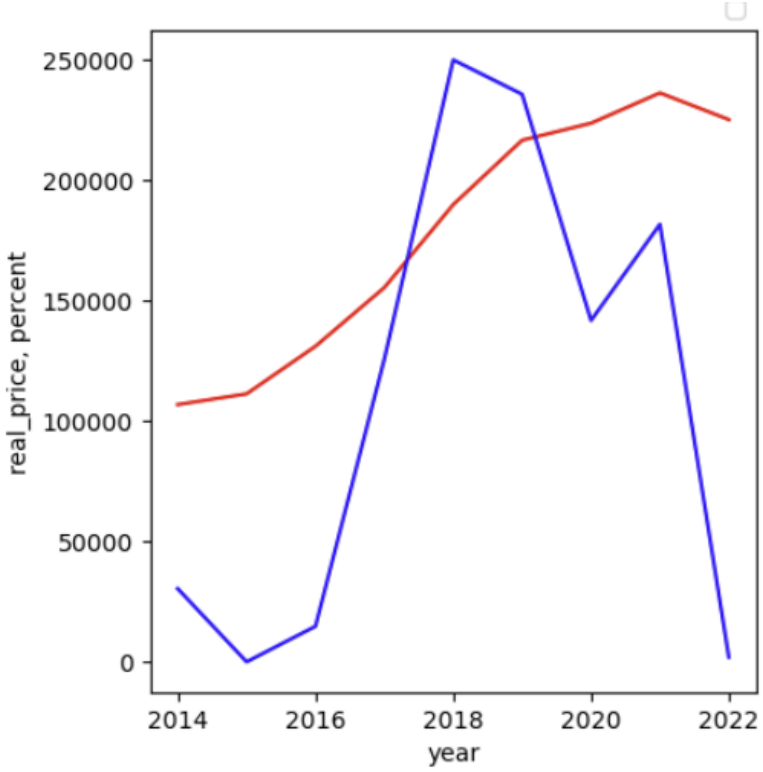
	transaction_year_month	percent
14	201401	0.211
26	201401	0.211
67	201401	0.211
116	201402	0.300
113	201402	0.300
122	201402	0.300
125	201402	0.300
148	201402	0.300
166	201402	0.300
173	201403	0.510

Part 1

시군구별 지가변동률

시군구별로 월마다 지가변동률 데이터 - 국가통계포털(KOSIS)





Part 2

데이터 전처리

Part 2

데이터 전처리와 feature 생성

데이터청년캠퍼스 3조

가격과 상관계수가 높으면 학습 결과가 좋을꺼야!

Part 2

주택규모 및 매매지수 전처리

데이터청년캠퍼스 3조

STEP 1		STEP 2		STEP 3
대치동 데이터에 주택규모와 매매지수 추가	>>	대치동 데이터 매매지수 결측치 평균으로 대체	>>	주택규모 별 층 별 평균 가격 칼럼 추가

Part 2

주택규모 및 매매지수 전처리

STEP 1 - 대치동 데이터에 주택규모와 매매지수 추가하기

all_data 매매지수 칼럼 추가

```
all_data
all_data["매매지수"]=0
```

```
for i,x in enumerate(all_data[['주택규모별','연도','월']].values):
    for ii, y in enumerate(korea_price[['주택규모별','연도','월']].values):
        if (x[0]==y[0]) and (x[1]==y[1]) and (x[2]==y[2]):
            all_data.iloc[i,-1]=korea_price.iloc[ii,-3]
        else:
            pass
```

대치동 데이터에 매매지수를 추가

Part 2

주택규모 및 매매지수 전처리

STEP 2 - 대치동 데이터의 매매지수 결측치 평균 값으로 대체

all_data 매매지수 칼럼 결측치 처리

```
sum((all_data['주택규모별']=='1')&(all_data['연도']<=2018)&(all_data['매매지수']==0))
```

```
136
```

주택규모가 1 인 아파트는 2017.11월 전까지 매매지수 데이터가 존재하지 않음을 알 수 있음

```
mean_value = round(all_data[all_data['매매지수'] != 0.0]['매매지수'].mean(),1)
all_data['매매지수'] = all_data['매매지수'].replace(0.0,mean_value)
```

매매지수가 0인 데이터들을 매매지수의 평균으로 대체

Part 2

주택규모 및 매매지수 전처리

STEP 3 - 주택규묘 별 층 별 칼럼 추가하기

주택규모별 층별 평균 가격

```
import numpy as np

# 아파트 별로 가격 평균값 구하기
data = all_data[all_data['train_test'] == 'train']
data
data['transaction_real_price']=data['transaction_real_price'].replace('.', '').astype('int')
data = data[['주택규모별','floor','transaction_real_price']]
data
```

•••

```
data['주택규모별floor'] = data['주택규모별'].astype('str') + data['floor'].astype('str')
size_floor_price = data[['transaction_real_price']].groupby(data['주택규모별floor']).mean().reset_index()
size_floor_price
```

•••

```
all_data['주택규모별floor'] = all_data['주택규모별'].astype('str') + all_data['floor'].astype('str')
all_data['size_floor_price'] = 0
```

```
for i,x in enumerate(all_data[['주택규모별floor','size_floor_price']].values):
    for ii, y in enumerate(size_floor_price[['주택규모별floor','transaction_real_price']].values):
        if x[0]==y[0] :
            all_data.iloc[i,-1]=size_floor_price.iloc[ii,-1]
        else:
            pass
```

주택 규모별 층별 평균 가격을 추가

Part 2

전세 데이터 전처리

데이터청년캠퍼스 3조

STEP 1		STEP 2
대치동 데이터에 전세 가격 칼럼 추가	>>	결측값에 대해 중앙값 으로 대체

Part 2

전세 데이터 전처리

STEP1 - 대치동 데이터에 전세 가격 칼럼 추가

```
junse_price['junse_id'] = junse_price['접수연도'].astype('str')  + junse_price['임대면적(㎡)'].astype('str') + junse_price['건물명'].astype('str')
```

```
junse_price_gr = junse_price.groupby('junse_id')['보증금(만원)'].mean().reset_index()
junse_price_gr
```

•••

```
all_data['junse_id'] = all_data['연도'].astype('str') + all_data['exclusive_use_area'].astype('str') + all_data['apt_name'].astype('str')
```

```
all_data['junse_price'] = 0
```

```
for i,x in enumerate(all_data[['junse_id','junse_price']].values):
    for ii, y in enumerate(junse_price_gr[['junse_id','보증금(만원)']].values):
        if x[0]==y[0] :
             all_data.iloc[i,-1]=junse_price_gr.iloc[ii,-1]
        else:
            pass
```

접수연도와 면적, 건물명을 공통 인자로 삼아 전세 ID를 생성
전세 ID 그룹별로 평균을 내서 대치동 데이터에 추가

전세 데이터 전처리

STEP 2 - 결측값에 대해 중앙값으로 대체

```
sum(all_data['junse_price'] == 0)
```

```
328
```

```
# 전세가 0인 값에 대해 중앙값 계산
median_junse_price = all_data.loc[all_data['junse_price'] > 0, 'junse_price'].median()

# 중앙값으로 채우기
all_data.loc[all_data['junse_price'] == 0, 'junse_price'] = median_junse_price
```

```
all_data.loc[all_data['junse_price'] > 0, 'junse_price'].median()
```

```
73900.0
```

```
sum(all_data['junse_price'] == 0 )
```

```
0
```

대치동 데이터에 0으로 나오는 328개의 값들을 전세 가격의 중앙값으로 대체

Part 2

K means cluster

주택규모별 가격 군집화

Dacon코드와 같은 아파트 이름
기준 가격 군집화
(k=20)

초등학교 거리별 가격 군집화

Part 2

K means cluster

```
from sklearn.cluster import KMeans
import numpy as np

# 아파트 별로 가격 평균값 구하기
train = all_data[all_data['train_test'] == 'train']
train['transaction_real_price'] = train['transaction_real_price'].str.replace(',', '').astype('int')
data = train[['apt_name', 'transaction_real_price']]

data = data.groupby('apt_name').mean()
arr = data['transaction_real_price'].to_numpy().reshape(-1, 1)

# 가격을 기준으로 아파트 군집화
k = 5
kmeans = KMeans(n_clusters=k, n_init=10)
kmeans.fit(arr)

# 가격을 기준으로 군집의 순서를 정렬하기 위해 인덱스를 추출
sort_order = np.argsort(kmeans.cluster_centers_.flatten())

# 군집화 결과를 가격 순서대로 재할당
labels = np.zeros_like(kmeans.labels_)
for i, cluster in enumerate(sort_order):
    labels[kmeans.labels_ == cluster] = i

# 군집화 결과와 가격을 데이터에 추가
data['cluster'] = labels
data = data.reset_index()
data = data[['apt_name', 'cluster']]
data
```

Size_sale

	주택규모별	size_sale
0	1	0
1	2	1
2	3	2
3	4	3
4	5	4

Trade_price_With_apt_name

	apt_name	trade_price_with_apt_name
0	개포우성1	15
1	개포우성2	16
2	국제	6
3	노빌리티빌리지	12
4	대우아이빌멤버스(891-26)	0

cluster

	element school distance(m)	cluster
0	100	6
1	135	3
2	155	1
3	160	1
4	165	6

Part 2

대치동 인구밀도 전처리

STEP 1		STEP 2		STEP 3
대치동 데이터의 아파트 이름 별 대치동 1동, 2동, 4동 분류	>>	대치동 데이터에 동별 인구밀도 데이터 추가	>>	대치동 데이터의 2023년도 동별인구밀도 2022년도 인구밀도로 결측치 대체

Part 2

대치동 인구밀도 전처리

STEP 1 - 대치동 데이터의 아파트 이름 별 대치동 1동, 2동, 4동 분류하기

```
daechi_1 = ['선경1차(1동-7동)', '선경3차', '선경2차(8동-12동)', '개포우성1', '개포우성2',
            '동부센트레빌', '래미안대치팰리스', '대치SKVIEW', '노빌리티빌리지', '대치삼성',
            '롯데캐슬리베', '대치하나빌', '롯데캐슬', '미도맨션3차', '삼성2차', '삼성3차',
            '청암빌라트']

daechi_2 = ['우정에쉐르', '대치동우정에쉐르1', '대치동우정에쉐르2(890-42)', '화인하이빌',
            '대치아이파크', '대치우성아파트1동,2동,3동,5동,6동,7동', '대치주성',
            '대치현대', '대치효성', '래미안대치하이스턴', '상지리츠빌카일룸(1009-4)', '은마',
            '풍림아이원3차(1007-1)', '풍림아이원4차(1007-2)', '하이캐슬', '하이캐슬102동',
            '한보미도맨션1', '한보미도맨션2', '한양팰리스', '해암프리존', '현대1', '세영팔레스타운',
            '(977-)']

daechi_4 = ['대우아이빌멤버스(891-26)', '대우아이빌명문가(891-23)', '대치동우정에쉐르1',
            '대치동우정에쉐르2(890-42)', '대치타워', '대치한신휴플러스', '동민맥스빌A동',
            '동민맥스빌B동', '동양', '메트로', '삼환SOGOOD', '선릉역풍림아이원레몬', '세연파크뷰',
            '스카이써밋아파트', '쌍용대치2', '쌍용대치아파트1동,2동,3동,5동,6동', '아름빌(889-74)',
            '월드빌', '청원', '테헤란로대우아이빌(891-6)', '포스코더샵', '풍림아이원1차101동(910-3)',
            '풍림아이원1차103동(910-5)', '풍림아이원2차201동', '풍림아이원2차202동', '한티(933-0)',
            '한티(933-35)', '현대썬앤빌테헤란', '신성미소시티', '풍림아이원아파트']
```

```
def daechi(x):
    if x in daechi_1:
        return 'daechi_1'
    elif x in daechi_2:
        return 'daechi_2'
    else:
        return 'daechi_4'
```

```
all_data['daechi'] = all_data['apt_name'].apply(daechi)
```

```
all_data[['apt_name', 'daechi']]
```

	apt_name	daechi
0	동부센트레빌	daechi_1
1	하이캐슬	daechi_2
2	대치효성	daechi_2
3	한보미도맨션1	daechi_2
4	테헤란로대우아이빌(891-6)	daechi_4
...	...	...
6178	포스코더샵	daechi_4
6179	한보미도맨션2	daechi_2
6180	개포우성2	daechi_1
6181	대치우성아파트1동,2동,3동,5동,6동,7동	daechi_2
6182	대치효성	daechi_2

Part 2

대치동 인구밀도 전처리

STEP 2 - 대치동 데이터에 동별인구밀도 데이터 추가하기

```
all_data[['train_test', '연도', 'daechi', '동별인구밀도']]
```

	train_test	연도	daechi	동별인구밀도
0	train	2014	daechi_1	25068.0
1	train	2014	daechi_2	21559.0
2	train	2014	daechi_2	21559.0
3	train	2014	daechi_2	21559.0
4	train	2014	daechi_4	30321.0
...	...	...	...	...
6178	test	2023	daechi_4	NaN
6179	test	2023	daechi_2	NaN
6180	test	2023	daechi_1	NaN
6181	test	2023	daechi_2	NaN
6182	test	2023	daechi_2	NaN

인구밀도 데이터에 2023년 데이터가 없어서 결측치 생성

Part 2

대치동 인구밀도 전처리

STEP 3 - 대치동 데이터의 2023년도 동별인구밀도 결측치 채우기

```
for i,x in enumerate(all_data[['연도','daechi']].values):
    for ii, y in enumerate(mm['시점'].values):
        if x[0]==int(y):
            if (x[1]=='daechi_1'):
                all_data.iloc[i,-1]=mm.iloc[ii,-4]
            elif (x[1]=='daechi_4'):
                all_data.iloc[i,-1]=mm.iloc[ii,-3]
            else:
                all_data.iloc[i,-1]=mm.iloc[ii,-2]
        elif x[0]==2023:
            if (x[1]=='daechi_1'):
                all_data.iloc[i,-1]=mm[mm['시점']=='2022'].iloc[0,-4]
            elif (x[1]=='daechi_4'):
                all_data.iloc[i,-1]=mm[mm['시점']=='2022'].iloc[0,-3]
            else:
                all_data.iloc[i,-1]=mm[mm['시점']=='2022'].iloc[0,-2]
all_data['동별인구밀도'] = all_data['동별인구밀도'].astype(int)
```

```
all_data[['train_test', '연도', 'daechi', '동별인구밀도']]
```

	train_test	연도	daechi	동별인구밀도
0	train	2014	daechi_1	25068
1	train	2014	daechi_2	21559
2	train	2014	daechi_2	21559
3	train	2014	daechi_2	21559
4	train	2014	daechi_4	30321
...	...	...	...	...
6178	test	2023	daechi_4	26700
6179	test	2023	daechi_2	18549
6180	test	2023	daechi_1	30185
6181	test	2023	daechi_2	18549
6182	test	2023	daechi_2	18549

결측치에 2022년 동별인구밀도 데이터로 대체

Part 2

금리데이터 전처리

데이터청년캠퍼스 3조

STEP 1		STEP 2		STEP 3
금리데이터에 날짜 변수 생성	>>	대치동 데이터의 날짜와 금리데이터의 날짜를 비교하여 대치동 데이터에 금리 추가	>>	날짜가 다를 경우 이전 날짜의 금리로 대체

Part 2

금리데이터 전처리

데이터청년캠퍼스 3조

```
def make_date(row):
    month_day = row['월일'].replace('월 ', '-')
    month_day = month_day.replace('일', '')
    date = str(row['연도'])+ '-' + month_day
    return date


interest_rate['날짜'] = interest_rate.apply(lambda x: make_date(x), axis=1)
interest_rate['날짜'] = pd.to_datetime(interest_rate['날짜'])

for idx, row in tqdm(all_data.iterrows(), total = all_data.shape[0]):
    date = row['transaction_date']
    rate = interest_rate[interest_rate['날짜'] <= date].iloc[0]['금리']
    all_data.loc[idx, 'interest_rate'] = rate
```

	transaction_date	interest_rate
14	2014-01-07	2.5
26	2014-01-09	2.5
67	2014-01-19	2.5
116	2014-02-10	2.5
113	2014-02-10	2.5
...	...	...
6155	2023-06-09	3.5
6169	2023-06-19	3.5
6173	2023-06-21	3.5
6179	2023-06-23	3.5
6054	2023-03-23	3.5

Part 2

시군구별 지가변동률 전처리

데이터청년캠퍼스 3조

STEP 1		STEP 2		STEP 3
지가변동률 데이터 **shape**를 대치동 데이터와 맞추기기	>>	지가변동률 데이터에 없는 **2023년 6월** 데이터를 지난 **2023년**간의 평균으로 추가하기	>>	지가변동률 데이터의 연월과 대치동 데이터의 연월이 같으면 지가 변동률 데이터를 삽입

Part 2

시군구별 지가변동률 전처리

데이터청년캠퍼스 3조

```
# 시군구별 지가변동률 전처리
import pandas as pd
percent = pd.read_csv('percent1.csv',encoding = 'cp949')
percent
```

	시군구별 (1)	시군구별 (2)	2014.01	2014.02	2014.03	2014.04	2014.05	2014.06	2014.07	2014.08	...	2022.08	2022.09	2022.10	2022.11	2022.12	2023.01	2023.02	2
0	시군구별 (1)	시군구별 (2)	평균	평균	평균	평균	평균	평균	평균	평균	...	평균	평균	평균	평균	평균	평균	평균	
1	서울특별시	강남구	0.211	0.300	0.510	0.311	0.211	0.283	0.276	0.283	...	0.312	0.255	0.100	0.030	0.021	0.054	0.064	

2 rows × 115 columns

```
p =percent.drop(['시군구별(1)','시군구별(2)'], axis = 1).transpose()
percent = p.drop(0,axis = 1)
percent.index = percent.index.str.replace('.','')
percent.rename(columns={1: '증감률'}, inplace = True)
percent = percent['증감률'].astype(float)
percent= percent.to_frame()
percent
```

	증감률
201401	0.211
201402	0.300
201403	0.510
201404	0.311
201405	0.211
...	...
202301	0.054
202302	0.064
202303	0.102
202304	0.093
202305	0.248

113 rows × 1 columns

Part 2

시군구별 지가변동률 전처리

데이터청년캠퍼스 3조

```
#nan 없애기
per = percent[percent.index.str.contains('2023')]
a = per['증감률'].mean()
new_index = '202306'
new_row = {'증감률': a}
percent.loc[new_index] = new_row

new_row
```

```
{'증감률': 0.11219999999999998}
```

percent

	증감률
201401	0.2110
201402	0.3000
201403	0.5100
201404	0.3110
201405	0.2110
...	...
202302	0.0640
202303	0.1020
202304	0.0930
202305	0.2480
202306	0.1122

114 rows × 1 colun

Result!

위 코드의 최종 결과는 `all_data` 에 지가변동률을 추가하고 데이터프레임의 **'transaction_year_month'** 값이 **'202306'**인 행은 **'2023'** 년도의 평균 증감률 값으로 변경되는 것

Part 2

recent_floor_price 전처리

데이터청년캠퍼스 3조

STEP 1

데이터프레임 `all_data`에서 인덱스 `idx`까지의 데이터를 임시 데이터프레임 `temp_df`로 생성.

>>

STEP 2

ID와 **floor**가 같은 아파트의 최근 시점의 **'transaction_real_price'** 값을 계산

결측치 제거

>>

STEP 3

`all_data` 데이터프레임의 각 행에 대해 **'recent_floor_price'** 열을 업데이트하는 반복문을 실행

Part 2

recent_floor_price 전처리

데이터청년캠퍼스 3조

```python
def get_recent_floor_price(idx, all_data):
    temp_df = all_data.loc[:idx]
    temp_df = temp_df[
        (temp_df['transaction_date'] < row['transaction_date']) &
        (temp_df['bucket_area'] == row['bucket_area'])
    ]
    if len(temp_df) == 0:
        temp_df = all_data[
            (all_data['transaction_date'] < datetime.strptime('2016-01-01', "%Y-%m-%d")) &
            (all_data['bucket_area'] == row['bucket_area'])
        ]


    recent_floor_price = temp_df[(temp_df['apartment_id'] == row['apartment_id'])]
    recent_floor_price =  temp_df[(temp_df['floor'] == row['floor'])]
    if len(recent_floor_price) == 0:
        recent_floor_price = temp_df[(temp_df['sigungu'] == row['sigungu'])]
        recent_floor_price = recent_floor_price.iloc[-1]['transaction_real_price']
    else:
        recent_floor_price = recent_floor_price['transaction_real_price'].mean()

    if recent_floor_price is None:
        recent_floor_price = temp_df['transaction_real_price'].mean() # 2019년 전체평균

    return recent_floor_price


for idx, row in tqdm(all_data.iterrows(), total = all_data.shape[0]):
    #if row['train_test'] == 'test':
        #continue
    all_data.loc[idx, 'recent_floor_price'] = get_recent_floor_price(idx, all_data)
```

Result!

`all_data` 데이터프레임의 'transaction_date', 'bucket_area', 'apartment_id', 'floor', 'sigungu', 'transaction_real_price' 열들을 활용하여 최근 시점의 가격 정보를 'recent_floor_price' 열에 추가.

Part 2

size_floor_price 전처리

데이터청년캠퍼스 3조

STEP 1		STEP 2		STEP 3
'주택규모별floor' 열을 생성하고 그룹별로 'transaction_real_price'의 평균 값을 계산하여 'size_floor_price' 데이터프레임을 생성	>>	all_data에 '주택규모별floor' 열을 만들고, 'size_floor_price' 열을 모두 0으로 초기화하여 'all_data'에 추가	>>	'all_data'의 각 행을 반복하며, '주택규모별floor' 값과 'size_floor_price' 데이터프레임의 값들을 비교하고 일치하는 값이 있으면 'transaction_real_price' 값을 할당

Part 2

size_floor_price 전처리

데이터청년캠퍼스 3조

```
import numpy as np

#train데이터 전처리
data = all_data[all_data['train_test'] == 'train']
data['transaction_real_price']=data['transaction_real_price'].replace('.', '').astype('int')
data = data[['주택규모별','floor','transaction_real_price']]
data
```

주택규모별, floor를 뽑아와서 하나로 합침

```
#(규모별+floor)별 transaction_real_price
data['주택규모별floor'] = data['주택규모별'].astype('str') + data['floor'].astype('str')
size_floor_price = data[['transaction_real_price']].groupby(data['주택규모별floor']).mean().reset_index()
size_floor_price
```

	주택규모별floor	transaction_real_price
0	110	27883.870968
1	111	25237.878788
2	112	26636.000000
3	113	26409.230769
4	114	27285.294118
...	...	...
173	65	222776.956522
174	66	212302.210526
175	67	209477.333333
176	68	255407.894737
177	69	226970.400000

Part 2

size_floor_price 전처리

데이터청년캠퍼스 3조

```
#all_data에 size_floor_price 넣기

all_data['주택규모별floor'] = all_data['주택규모별'].astype('str') + all_data['floor'].astype('str')
all_data['size_floor_price'] = 0
for i,x in enumerate(all_data[['주택규모별floor','size_floor_price']].values):
    for ii, y in enumerate(size_floor_price[['주택규모별floor','transaction_real_price']].values):
        if x[0]==y[0] :
             all_data.iloc[i,-1]=size_floor_price.iloc[ii,-1]
        else:
            pass
```

```
all_data[['주택규모별floor','transaction_real_price']].head()
```

	주택규모별floor	transaction_real_price
0	616	180000.0
1	35	68000.0
2	38	72600.0
3	65	155000.0
4	120	27000.0

Result!

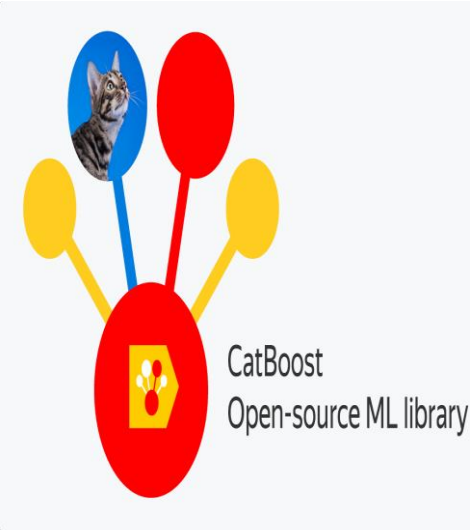
'주택규모별floor' 그룹별로 **'transaction_real_price'**의 평균 값을 계산하여 **'all_data'**에 주택규모별**floor** 평균 가격 정보를 추가

Part 3

학습모델/데이터 선택

Part 3

학습 모델



CatBoost

dmlc
XGBoost

XGBoost

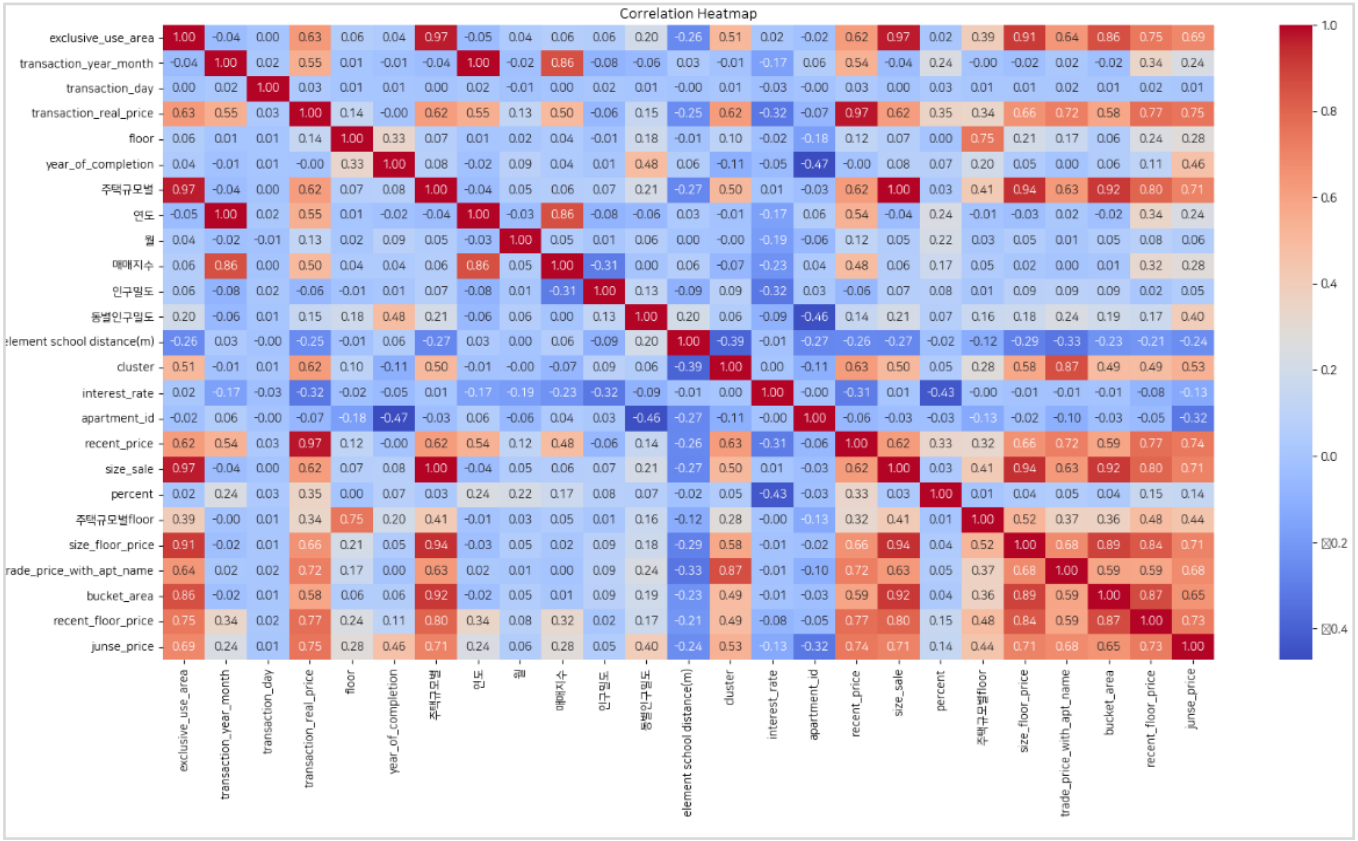
LGBM

Part 3

변수간의 상관계수

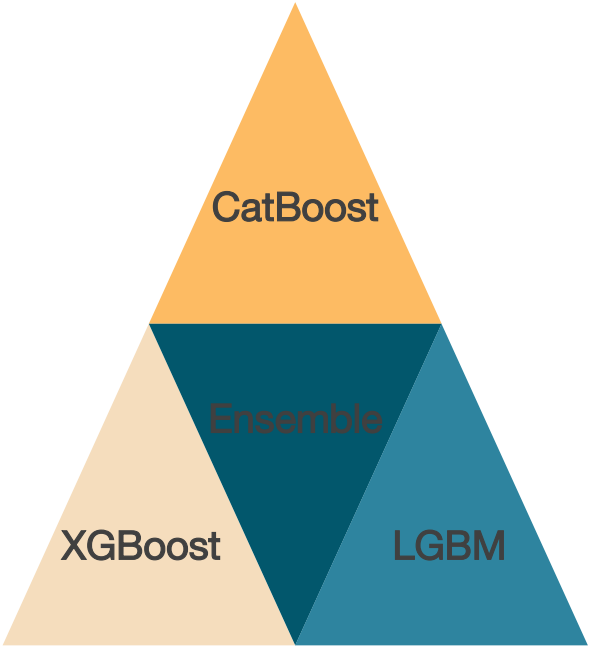
sns.heatmap(all_data_copy.corr(), cmap='coolwarm', annot=True, fmt='.2f')

Heatmap



Part 3

학습 모델/ 데이터 선택



변수를 바꿔가며 MAE로 평가..?

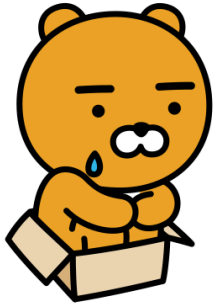
```
2018 년도 MAE_bp:  35087.013544189664
2018 년도 MAE_cb:  32284.32480979835
2018 년도 MAE_xbb:  41148.743394746554
2018 년도 MAE_lgbm:  32284.32480979835
-------------------------------------
2019 년도 MAE_bp:  24154.704497733826
2019 년도 MAE_cb:  25574.825622295542
2019 년도 MAE_xbb:  22671.31502495247
2019 년도 MAE_lgbm:  25574.825622295542
-------------------------------------
2020 년도 MAE_bp:  26772.169758475848
2020 년도 MAE_cb:  29554.731463998432
2020 년도 MAE_xbb:  21753.15654853465
2020 년도 MAE_lgbm:  29554.731463998432
-------------------------------------
2021 년도 MAE_bp:  37027.86823242886
2021 년도 MAE_cb:  38666.83330973121
2021 년도 MAE_xbb:  34554.596526731926
2021 년도 MAE_lgbm:  38666.83330973121
-------------------------------------
2022 년도 MAE_bp:  28318.121458304784
2022 년도 MAE_cb:  27314.11564929328
2022 년도 MAE_xbb:  33469.76689949682
2022 년도 MAE_lgbm:  27314.11564929328
-------------------------------------
```

```
2018 년도 MAE_bp:  35087.013544189664
2018 년도 MAE_cb:  32284.32480979835
2018 년도 MAE_xbb:  41148.743394746554
2018 년도 MAE_lgbm:  32284.32480979835
-------------------------------------
2019 년도 MAE_bp:  24154.704497733826
2019 년도 MAE_cb:  25574.825622295542
2019 년도 MAE_xbb:  22671.31502495247
2019 년도 MAE_lgbm:  25574.825622295542
-------------------------------------
2020 년도 MAE_bp:  26772.169758475848
2020 년도 MAE_cb:  29554.731463998432
2020 년도 MAE_xbb:  21753.15654853465
2020 년도 MAE_lgbm:  29554.731463998432
-------------------------------------
2021 년도 MAE_bp:  37027.86823242886
2021 년도 MAE_cb:  38666.83330973121
2021 년도 MAE_xbb:  34554.596526731926
2021 년도 MAE_lgbm:  38666.83330973121
-------------------------------------
2022 년도 MAE_bp:  28318.121458304784
2022 년도 MAE_cb:  27314.11564929328
2022 년도 MAE_xbb:  33469.76689949682
2022 년도 MAE_lgbm:  27314.11564929328
-------------------------------------
```

```
2018 년도 MAE_bp:  41472.64423934964
2018 년도 MAE_cb:  41515.08094759135
2018 년도 MAE_xbb:  41496.552542138284
2018 년도 MAE_lgbm:  41515.08094759135
-------------------------------------
2019 년도 MAE_bp:  22927.002484910154
2019 년도 MAE_cb:  23091.986671689785
2019 년도 MAE_xbb:  23879.76478582462
2019 년도 MAE_lgbm:  23091.986671689785
-------------------------------------
2020 년도 MAE_bp:  23133.45787369886
2020 년도 MAE_cb:  23218.59143097304
2020 년도 MAE_xbb:  23756.810880829016
2020 년도 MAE_lgbm:  23218.59143097304
-------------------------------------
2021 년도 MAE_bp:  35695.84838181231
2021 년도 MAE_cb:  36433.700353701744
2021 년도 MAE_xbb:  34871.73104919679
2021 년도 MAE_lgbm:  36433.700353701744
-------------------------------------
2022 년도 MAE_bp:  28068.36844528472
2022 년도 MAE_cb:  27843.5884623973
2022 년도 MAE_xbb:  31958.808626853814
2022 년도 MAE_lgbm:  27843.5884623973
-------------------------------------
```

Part 3, 학습 모델/ 데이터 선택

모델

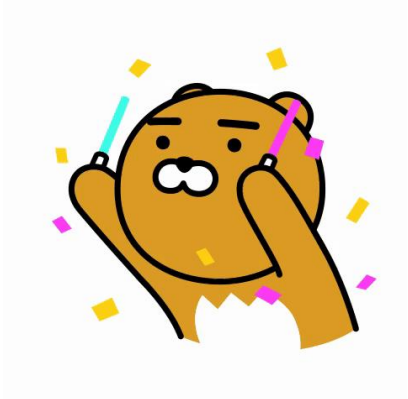


MAE값으로 최적의 모델 판단 불가능

↓

DACON Public score로 판단

변수



상관계수가 높은 변수 조합의 MAE가 높음

↓

MAE값이 전체적으로 낮아지는 변수 조합 선택!

Part 3, 학습 데이터 선택

all_data_copy.corr()['transaction_real_price'].sort_values(ascending=False)

```
transaction_real_price          1.000000
recent_price                    0.973714
recent_floor_price              0.774865
junse_price                     0.753817
trade_price_with_apt_name       0.718743
size_floor_price                0.656549
exclusive_use_area              0.627561
cluster                         0.624279
주택규모별                           0.615145
size_sale                       0.615145
bucket_area                     0.582424
transaction_year_month          0.549810
연도                            0.548266
매매지수                            0.497436
percent                         0.349821
주택규모별floor                      0.341173
동별인구밀도                          0.148173
floor                           0.138266
월                               0.125457
transaction_day                 0.029999
year_of_completion             -0.000849
인구밀도                           -0.063937
apartment_id                   -0.069737
element school distance(m)     -0.252039
interest_rate                  -0.321855
```

항목 1	항목 2
recent_price	0.973714
recent_floor_price	0.774865
junse_price	0.753817
trade_price_with_apt_name	0.718743
size_floor_price	0.656549
exclusive_use_area	0.627561
cluster	0.624279
주택규모별	0.615145
size_sale	0.615145

상관계수 상위 9개의 변수 중 평가 결과가 좋은 변수 : 6개
총 사용 변수 : 6개+3개(연도,월,interest_rate)=9개

Part 4

학습결과

Part 4, 학습결과

Public Score

```
# 예측
pred_ls = list()
pred_ls_cb = list()
pred_ls_xgb = list()
pred_ls_LGBM = list()

now_df = all_data.loc[all_data['train_test'] == 'train']
test = all_data.loc[all_data['train_test'] == 'test']

for idx, row in tqdm(test.iterrows(), total = test.shape[0]):
    now_df = pd.concat([now_df, test.loc[[idx]]])
#     test_x.loc[idx, 'recent_price'] = get_recent_price(idx, now_df)

    # 예측
    pred_cb_ls = model_cb.predict(test_x.loc[idx:idx])
#     pred_rf_ls = model_rf.predict(test_x.loc[idx:idx])
    pred_xgb_ls = model_xgb.predict(test_x.loc[idx:idx])
    pred_LGBM_ls = model_xgb.predict(test_x.loc[idx:idx])

    blended_prediction = (pred_cb_ls + pred_xgb_ls + pred_LGBM_ls)/3

    now_df.loc[idx, 'transaction_real_price'] = blended_prediction
    pred_ls.append(blended_prediction[0])
    pred_ls_cb.append(pred_cb_ls[0])
    pred_ls_xgb.append(pred_xgb_ls[0])
    pred_ls_LGBM.append(pred_LGBM_ls[0])
```

앙상블 → 17279

CatBoost → 12414

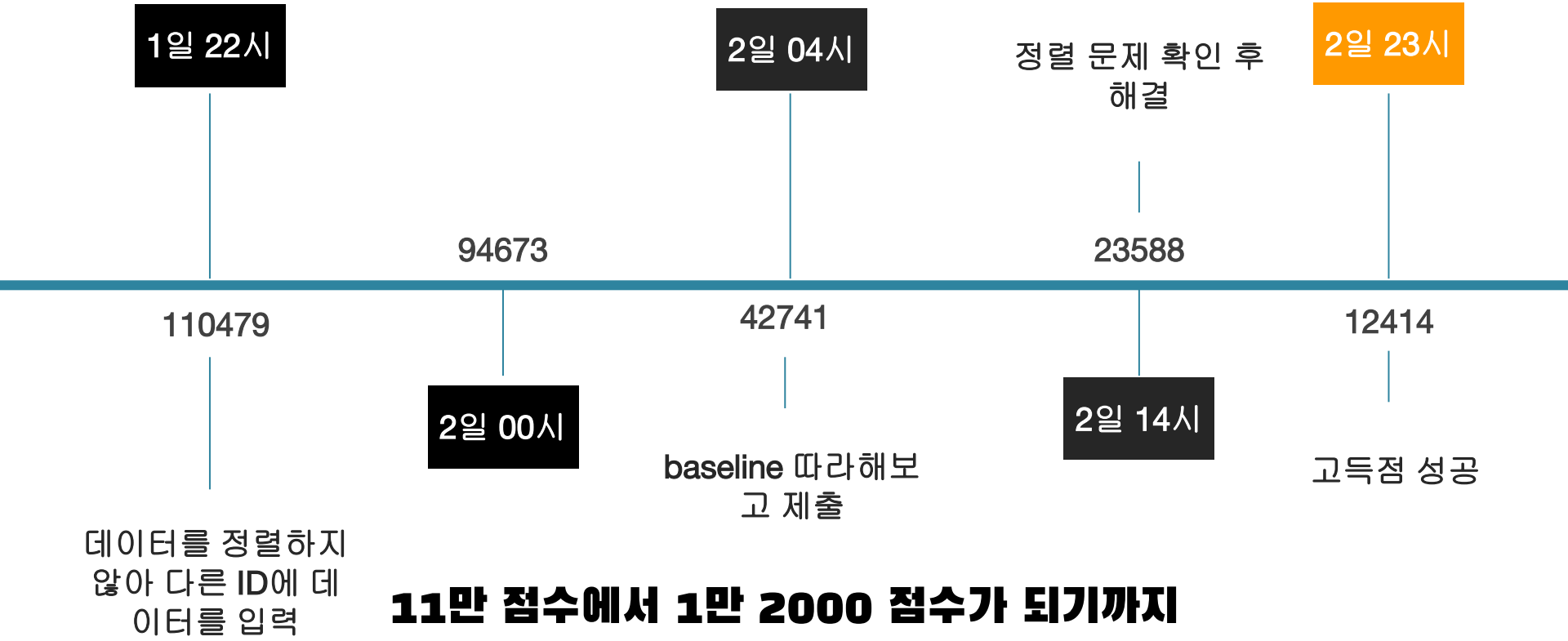
XGBoost → 22702

LGBM → 22702

Part 4,

학습결과

Public Score



11만 점수에서 1만 2000 점수가 되기까지

Part 4, 외부 데이터 출처

https://kosis.kr/statHtml/statHtml.do?orgId=408&tblId=DT_40803_N0003&vw_cd=MT_ZTITLE&list_id=C14_001&scrId=&seqNo=&lang_mode=ko&obj_var_id=&itm_id=&conn_path=MT_ZTITLE&path=%252FstatisticsList%252FstatisticsListIndex.do

수도권 규모별 매매가격지수
[2014 ~ 2023.06]

서울 열린데이터 광장

http://data.seoul.go.kr/dataList/OA-21276/S/1/datasetView.do

서울시 부동산 전월세가 정보
(2014년 ~ 2023년)

DATA 공공데이터포털 .GO.KR

https://data.seoul.go.kr/dataList/10790/S/2/datasetView.do

대치동 인구밀도 데이터(2012년 ~2022년)

https://map.naver.com/

네이버 지도 거리계산 기능

감사합니다