Analysing Various Price Factors of the Real Estate Market in Seoul, 2020

Content

1. Introduction

1.1 The Purpose of the Analysis

By analysing various price factors affecting the real estate market in Seoul, this report is aiming to deliver a deeper insights on the recent surging property prices. The analysis consist the three points below:

1) Locational influence on real property values;

2) The qualities of the building; and

3) Demographic influence

1.2 The Content of the Analysis

Part 1. Locational influence on real property values

- Education

- Transportation
- Satisfaction Score by Residents

Part 2. The qualities of the building

- Year built
- Area of the property
- The brand value of the certain apartment

Part 3. Demographic influence

- Demographic information on the certain region

2. Data Processing and Explanatory Data Analysis (EDA)

2.1. Data Gathering

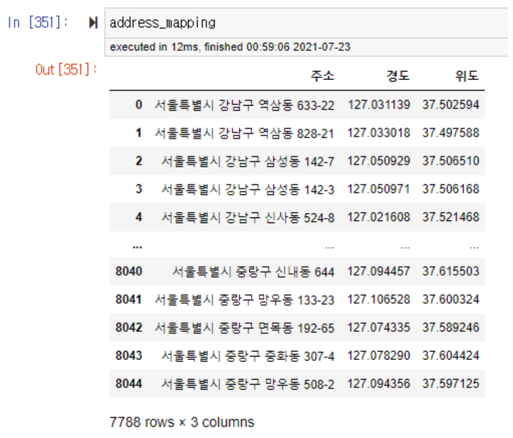
- 2020년 서울시 부동산 실거래가 정보(http://data.seoul.go.kr/dataList/OA-15548/S/1/datasetView.do)
- 서울부동산정보광장_실거래가정보(https://land.seoul.go.kr:444/land/rtms/transactionInfo.do)
- 서울시 구 경계, 법정동 경계, 행정동 경계 shp (http://data.nsdi.go.kr/dataset/15144)
- 서울시 지하철 위경도 데이터(https://observablehq.com/@taekie/seoul_subway_station_coordinate)
- 서울교통공사 노선별 지하철역 정보(http://data.seoul.go.kr/dataList/OA-15442/S/1/datasetView.do)
- 행정동별 생활인구 데이터(http://data.seoul.go.kr/dataList/OA-14991/S/1/datasetView.do)
- 자치구별 생활만족도 data(http://data.seoul.go.kr/dataList/218/S/2/datasetView.do)
- 서울시 학교 데이터(https://www.data.go.kr/tcs/dss/selectStdDataDetailView.do)
- 네이버 부동산 건축년도 크롤링 데이터(https://new.land.naver.com/)

2.2. Data Preprocessing

2.2.1 Adding a geographical location determined by latitude and longitude by using *vworld api*

▼ code

```
authorkey= 'key'#vworld에서 제공하는 api 권한 key값 입력
vworld_apikey = authorkey

for i in tqdm(range(len(address_mapping))):
    try:
        addr = address_mapping['주소'][i]

        # 지번일 경우 url 부분에서type = PARCEL, 도로명일 경우 url 부분에서 type= ROAD
        url = "http://api.vworld.kr/req/address?service=address&request=getCoord&type=PARCEL&refine=false&key=%s&" % (vworld_apikey) +

        request = Request(url)
        response = urlopen(request)
        rescode = response.getcode()
        response_body = response.read().decode('utf-8')
        jsonData = json.loads(response_body)

        address_mapping['위도'][i] = float(jsonData['response']['result']['point']['y'])
        address_mapping['경도'][i] = float(jsonData['response']['result']['point']['x'])

    except:
        pass
```



```
In [351]:  address_mapping
executed in 12ms, finished 00:59:06 2021-07-23
Out[351]:
```

	주소	경도	위도
0	서울특별시 강남구 역삼동 633-22	127.031139	37.502594
1	서울특별시 강남구 역삼동 828-21	127.033018	37.497588
2	서울특별시 강남구 삼성동 142-7	127.050929	37.506510
3	서울특별시 강남구 삼성동 142-3	127.050971	37.506168
4	서울특별시 강남구 신사동 524-8	127.021608	37.521468
...	...	...	...
8040	서울특별시 중랑구 신내동 644	127.094457	37.615503
8041	서울특별시 중랑구 망우동 133-23	127.106528	37.600324
8042	서울특별시 중랑구 면목동 192-65	127.074335	37.589246
8043	서울특별시 중랑구 중화동 307-4	127.078290	37.604424
8044	서울특별시 중랑구 망우동 508-2	127.094356	37.597125

7788 rows × 3 columns

2.2.2 Adding the district and neighbourhood information

▼ code

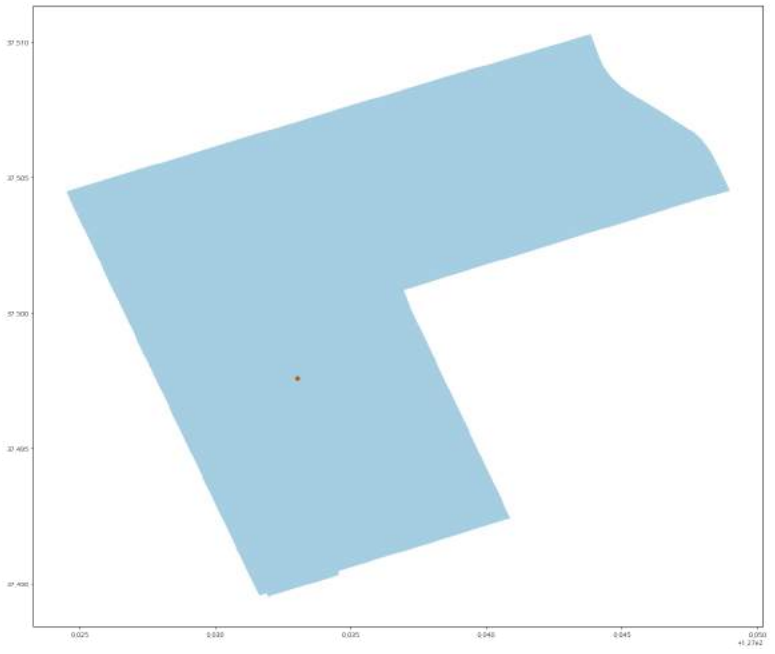
```
import pandas as pd
import geopandas as gpd
import numpy as np
from tqdm import tqdm
from shapely.geometry.multipolygon import MultiPolygon
from shapely.geometry import Point, Polygon, LineString

# 구경계, 행정동 경계 의 data를 불러옴 geojson file이라 geopandas를 활용하여 불러옴
gu = gpd.read_file('./data/구경계_espg4326.geojson')
dong2 = gpd.read_file('./data/행정동경계__espg4326.geojson')

# 코드 예시
# geometry 계산이 가능하도록 위경도를 Point로 묶어주는 작업
df['geometry'] = df.apply(lambda row : Point([row['경도'], row['위도']]), axis=1)

def define_gu(df):
    df['구'] = np.nan
    for i in tqdm(range(len(df))):
        for j in range(len(gu)):
            try:
                # contains method를 이용하여 면적에 점이 포함되는지 여부 확인
                # 포함이 되면 True 출력, 포함 안될시 False
                if gu['geometry'][j].contains(df['geometry'][i]):
                      df['구'][i] = gu['SGG_NM'][j]
            except:
                pass
```

▼ 예시

아래 그림과 같이 면적(역삼1동 행정동 shape)을 가지는 공간에 빨간색 점(역삼동 아파트)이 포함 여부 확인 가능

이와 같은 방식으로 구와 행정동을 구분

```
#시각화 확인 코드
import geopandas as gpd
gpd.GeoSeries([dong2.geometry[365], apt.geometry[1]]).plot(cmap='Paired',figsize=(20,20))
```



2.2.3 Adding the null values of Year Built by web crawling

▼ code

```
import time
from selenium import webdriver
import numpy as np
import time
import requests
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.keys import Keys


driver = webdriver.Chrome()  # Optional argument, if not specified will search path.
url = 'https://land.naver.com/'
driver.get(url)
time.sleep(2)

# 검색용 리스트 생성
arr = ['래미안 장위포레카운티',
 '롯데캐슬 클라시아 ',
 '서울역센트럴자이',
 '래미안장위퍼스트하이',
 '신정뉴타운1-1구역 아이파크위브',
 '용산 센트럴파크 해링턴 스퀘어',
 '상계역센트럴푸르지오',
 '래미안 리더스원',
 '휘경 해모로 프레스티지']


jungong_list = []
x = []
y = []
l = []

for i in range(arr_len):
    jungong_list = []
    driver.get('https://land.naver.com/')
    time.sleep(1)
    danjiname = arr[i]
    elem_search = driver.find_element_by_class_name("search_input")
    elem_search.clear()
    elem_search.send_keys(danjiname)
    elem_search.send_keys(Keys.RETURN)

    try:
        time.sleep(1)
        button = driver.find_element_by_class_name("complex_link").click()
        time.sleep(1)
        detail = driver.find_element_by_id("detailContents1")
        deep1 = detail.find_elements_by_class_name('table_th')
        deep2 = detail.find_elements_by_class_name('table_td')

        jungong_list.append(deep2[2].text)
        jungong_num = jungong_list[0]
        x = [arr[i], jungong_num]
        #print(x)

        l.append(x)
        print(l[-1])

    except :
        y = [arr[i], "정보검색안됨"]
        l.append(y)
        print(l[-1])
```

2.2.4 Adding demographic information

▼ code

```
import pandas as pd
import numpy as np


어린이 = ['남자0세부터9세생활인구수','여자0세부터9세생활인구수']
청소년 = ['남자10세부터14세생활인구수',
       '남자15세부터19세생활인구수','여자10세부터14세생활인구수', '여자15세부터19세생활인구수']
청년 = ['남자20세부터24세생활인구수', '남자25세부터29세생활인구수',
       '남자30세부터34세생활인구수', '남자35세부터39세생활인구수', '여자20세부터24세생활인구수',
       '여자25세부터29세생활인구수', '여자30세부터34세생활인구수', '여자35세부터39세생활인구수']
중년 = ['남자40세부터44세생활인구수',
       '남자45세부터49세생활인구수', '남자50세부터54세생활인구수', '남자55세부터59세생활인구수','여자40세부터44세생활인구수', '여자45세부터49세생활인구수', '여자50
       '여자55세부터59세생활인구수']
장년 = ['남자60세부터64세생활인구수', '남자65세부터69세생활인구수', '남자70세이상생활인구수','여자60세부터64세생활인구수', '여자65세부터69세생활인구수',
       '여자70세이상생활인구수' ]

population = pd.read_csv('./data/LOCAL_PEOPLE_DONG_202012.csv')

# 0시부터 6시까지의 시간대만 추출
df = population[(population['시간대구분']>=0) & (population['시간대구분']<=6)]

# 행정동 코드별로 groupby후 평균
new_df = df.groupby('행정동코드').mean().reset_index()

#필요없는 column 삭제
new_df.drop(['기준일ID', '시간대구분'], axis=1, inplace=True)

# 연령에 따라 어린이, 청소년, 청년, 장년 4가지 카테고리로 구분
new_df['어린이'] = new_df[어린이[0]] +new_df[어린이[1]]
new_df['청소년'] = new_df[청소년[0]] + new_df[청소년[1]] + new_df[청소년[2]] + new_df[청소년[3]]
new_df['청년'] = new_df[청년[0]] + new_df[청년[1]]+new_df[청년[2]]+new_df[청년[3]]+new_df[청년[4]]+new_df[청년[5]]+new_df[청년[6]]+new_df[청년[7
new_df['중년'] = new_df[중년[0]] + new_df[중년[1]]+new_df[중년[2]]+new_df[중년[3]]+new_df[중년[4]]+new_df[중년[5]]+new_df[중년[6]]+new_df[중년[7
new_df['장년'] = new_df[장년[0]] + new_df[장년[1]]+new_df[장년[2]]+new_df[장년[3]]+new_df[장년[4]]+new_df[장년[5]]

#필요한 column만 추출
new_df2 = new_df[['행정동코드','총생활인구수', '어린이', '청소년', '청년', '중년', '장년']]

# 비율 계산
new_df2['어린이비율(%)'] = new_df2['어린이'] /  new_df2['총생활인구수']* 100
new_df2['청소년비율(%)'] = new_df2['청소년'] /  new_df2['총생활인구수']* 100
new_df2['청년비율(%)'] = new_df2['청년'] /  new_df2['총생활인구수']* 100
new_df2['중년비율(%)'] = new_df2['중년'] /  new_df2['총생활인구수']* 100
new_df2['장년비율(%)'] = new_df2['장년'] /  new_df2['총생활인구수']* 100

#정리된 파일 저장
new_df2.to_csv('./행정동별인구비율.csv', index=False)
```

2.2.5 Obtaining the distance between the property and public transportation (metro) using *haversine*

Haversine Docs = https://www.kite.com/python/docs/haversine.haversine

▼ code

```
import pandas as pd
import numpy as np
import geopandas as gpd
import folium
import time
import json
from shapely.geometry.multipolygon import MultiPolygon
from shapely.geometry import Point, Polygon, LineString
import shapely
from tqdm import tqdm
from haversine import haversine


subway = pd.read_csv('./new_data/서울시지하철(new).csv')
bd = pd.read_csv('./new_data/실거래가_좌표추가.csv')

# 위도, 경도를 하나로 괄호안에 묶어주는 작업 시행
bd['위경도'] = bd.apply(lambda row : ([row['위도'], row['경도']]), axis=1)
subway['위경도'] = subway.apply(lambda row : ([row['위도'], row['경도']]), axis=1)

# 중복된 건물 제외(서로 다른 건물이 같은 경도나 위도를 가지는 경우가 없어, 경도를 기준으로 중복된 것을 제거)
bd2 = bd.drop_duplicates(subset='경도', keep='first')
bd2.reset_index(drop=True, inplace=True)

# 계산된 정보를 저장할 Dataframe와 data를 담아줄 list 생성
new_df = pd.DataFrame(columns = ['건물주소', '가장인근지하철역', '거리'])
address_lst = []
station_lst = []
lonlat_lst=[]
distance_lst = []

# 각 건물과 모든 지하철역의 거리를 계산 후, 최소값을 저장.
# 최소값이 가지는 data를 저장하여 dataframe에 넣어줌
for i in tqdm(range(len(bd2))):
    lst = []
    for j in range(len(subway)):
        lst.append(haversine(bd2['위경도'][i], subway['위경도'][j]))
        tmp = min(lst)
        index = lst.index(tmp)
    address_lst.append(bd2['주소'][i])
    lonlat_lst.append(bd2['위경도'][i])
    station_lst.append(subway['역이름'][index])
    distance_lst.append(tmp)
new_df['건물주소'] = address_lst
new_df['위경도'] = lonlat_lst
new_df['가장인근지하철역'] = station_lst
new_df['거리'] = distance_lst
```

2.2.6 Determining the public transportation acces

▼ code

```
# 역세권의 범위를 500m라고 설정하였으나,
# 아파트 주소의 위경도가 모든 아파트의 면적을 고려하지는 못하기 때문에
# 550m 를 역세권 여부를 판단하는 기준으로 설정

apt['역세권'] = np.nan
for i in tqdm(range(len(apt3))):
    if apt['subway거리'][i] < 0.55:
        apt['역세권'][i] = 1
    else:
        apt['역세권'][i] = 0
```

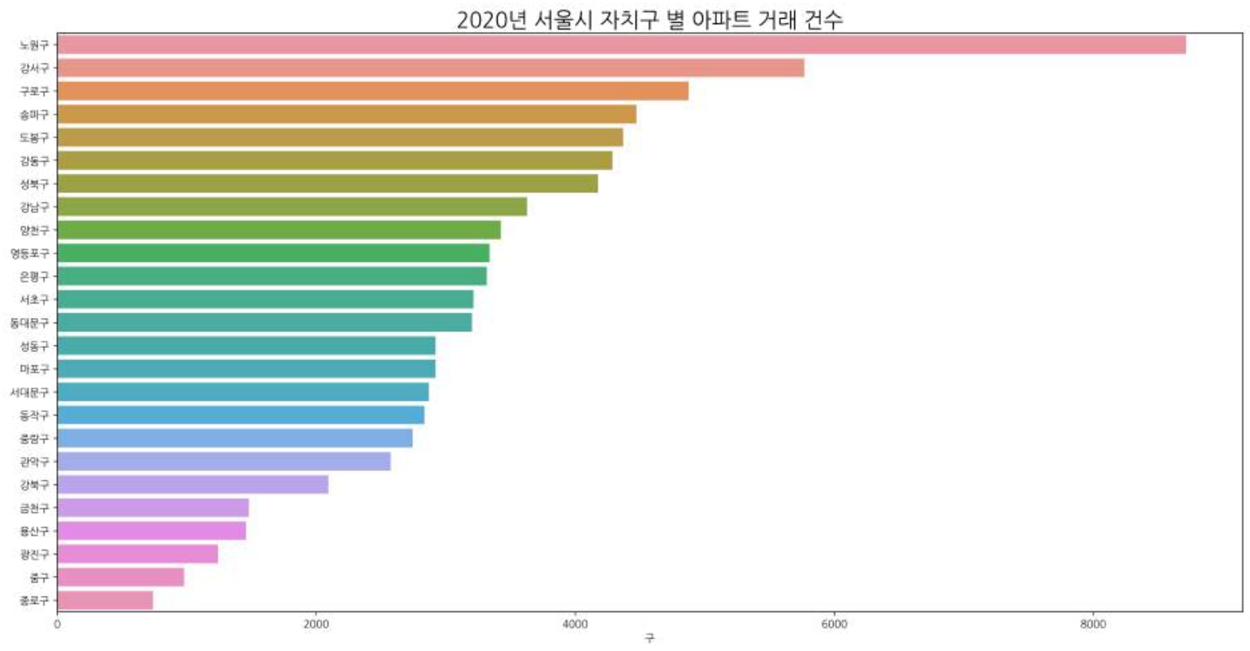
2.3 Explanatory Data Analysis (EDA)

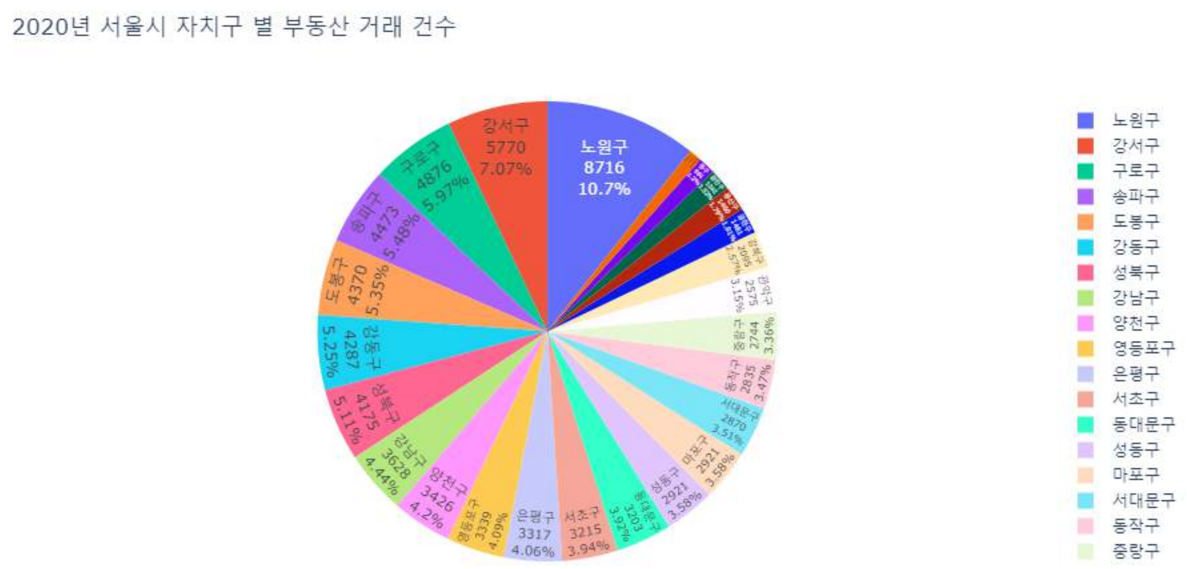
2.3.1 Visualisation of locational influence on real estate values

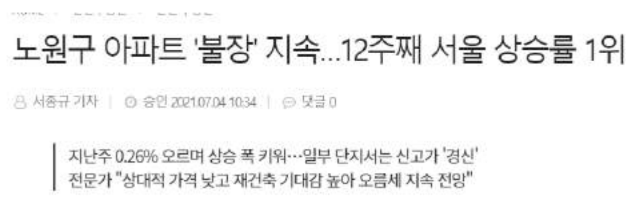
1) The number of transaction by the district, Seoul, 2020

▼ **code**

```
▶구별거래량 확인
apt.구.value_counts()

▶그래프
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)

plt.xticks(rotation=0)
sns.barplot(y = apt.구.value_counts().index, x = apt.구.value_counts())
ax.set_title("2020년 서울시 자치구 별 아파트 거래 건수", fontweight='bold',fontsize = 20)
plt.show()

import plotly.express as px
counts_gu = pd.DataFrame(apt.구.value_counts())
counts_gu = counts_gu.reset_index()
counts_gu.columns =['gu', 'counts']

fig = px.pie(counts_gu, values='counts', names='gu', title='2020년 서울시 자치구 별 부동산 거래 건수')
fig.update_traces(textposition='inside', textinfo='percent+label+value')
fig.show()
```






노원구 아파트 '불장' 지속...12주째 서울 상승률 1위

서종규 기자 | 승인 2021.07.04 10:34 | 댓글 0

지난주 0.26% 오르며 상승 폭 키워…일부 단지서는 신고가 '경신'
전문가 "상대적 가격 낮고 재건축 기대감 높아 오름세 지속 전망"

> 뉴스기사 中 (https://www.shinailbo.co.kr/news/articleView.html?idxno=1431485)

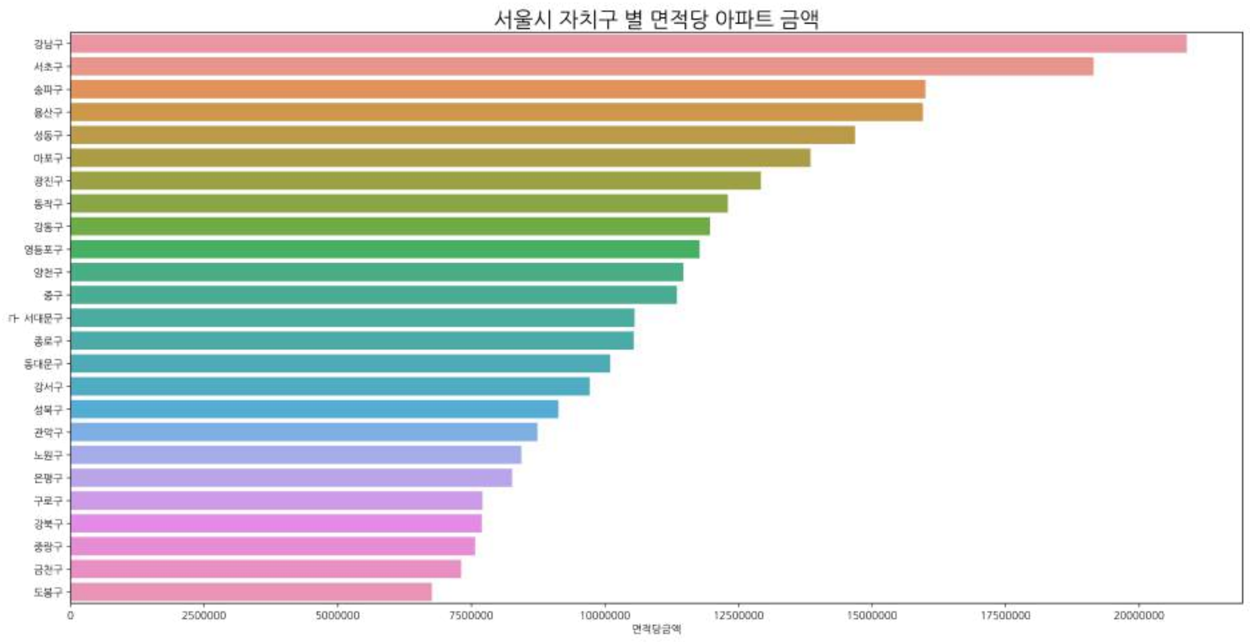
💡 Based on this graph, more than 10% of the real estate transaction have been located in Nowon District. According to the recent report, it can be understood as a combination of its relatively low price and the expectation of renovation.

2) The market value of real estate and the satisfaction scores of residents by the district of Seoul

a. Bar graph of the market value of real estate by the district of Seoul

▼ code

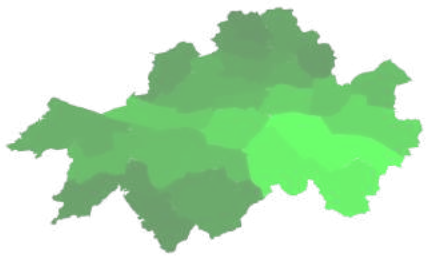
```
면적df = apt.groupby('구').mean()[['면적당금액']].reset_index()

면적df = 면적df.sort_values('면적당금액', ascending=False)

▶그래프
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)
plt.xticks(rotation=0)
sns.barplot(data=면적df, y='구', x='면적당금액')
ax.set_title("서울시 자치구 별 면적당 아파트 금액", fontweight='bold',fontsize = 20)
ax.get_xaxis().get_major_formatter().set_scientific(False)
plt.show()
```



b. Mapping the market value of real estate by the district of Seoul

▼ code

```
import pydeck as pdk
import pandas as pd
▶ rename 해주기
gu.rename(columns= {'SGG_NM':'구'}, inplace=True)
gu2 = pd.merge(gu, 면적df, on='구')

▶ 정규화된 값을 색상에 반영하기 위함
gu2['정규화'] = gu2['면적당금액'] / max(gu2['면적당금액'])

▶
gu2['coordinates'] = gu2['geometry'].apply(multipolygon_to_coordinates)

▶시각화
layer_gu = pdk.Layer( 'PolygonLayer',
                     gu2,
                     get_polygon='coordinates',
                     get_fill_color='[0,255*정규화,0]',
                     pickable=True,
                     extrude=True,
                     opacity=0.3,
                     stroked=False,
                     get_elevation = '정규화',
                     elevation_scale = 100,
                     elevation_range=[0,100],
                     auto_highlight=True)

center = [126.96216879726751, 37.50663783146973]
view_state = pdk.ViewState(
    longitude=center[0],
    latitude=center[1],
    zoom=10,
    bearing=15,
    pitch =45
)

tooltip = {"html": "<b>자치구:</b> {구} <br /><b>면적당 금액:</b> {면적당금액}"}

# Render
r = pdk.Deck(layers=[layer_gu], initial_view_state=view_state,
              #map_style='mapbox://styles/mapbox/outdoors-v11',
             map_style='mapbox://styles/mapbox/dark-v10',
             mapbox_key = 'map_box key값입력',
             tooltip=tooltip
            )
r.to_html()
```

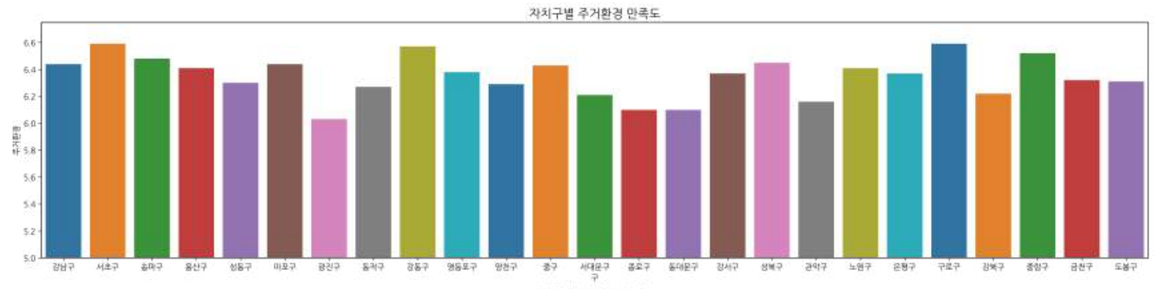


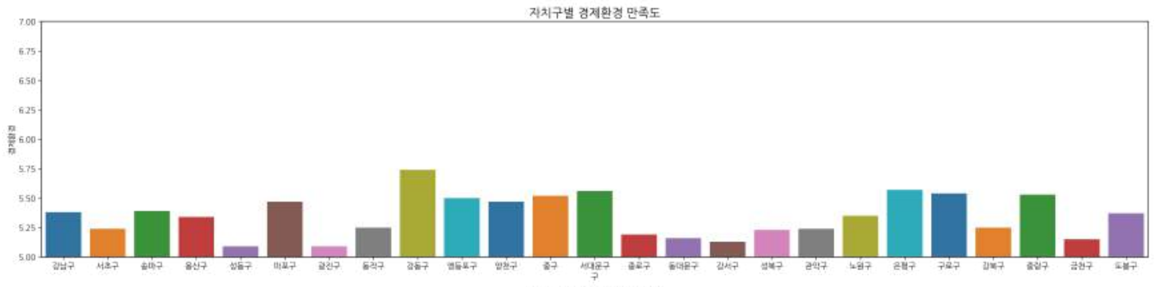
c. Bar graph of the satisfaction scores of residents by the district of Seoul

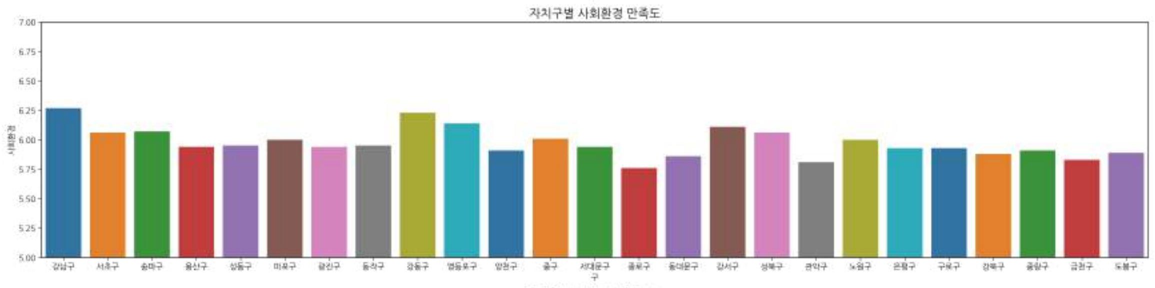
💡 Questions: We would like to ask how much you are satisfied of residing in this district.
(Very satisfied: 5, Satisfied: 4, Neutral: 3, Dissatisfied: 2, Very dissatisfied: 1)
- Converted the 5-point Likert scale to 10

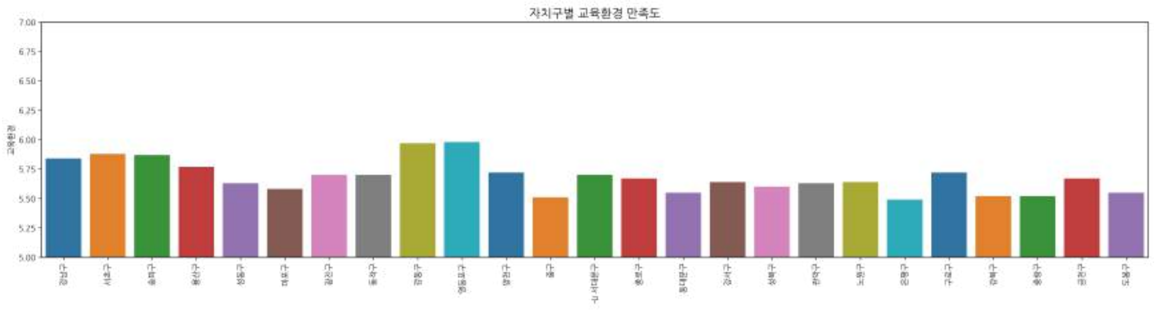
▼ code

```
▶전처리 작업
satis.drop(['기간', '대분류'], axis=1, inplace=True)
satis.rename(columns = {'분류':'구'}, inplace=True)
▶금액 비교를 위해 데이터 병합
new_면적df = pd.merge(면적df, satis, on='구')

▶구별 그래프(주거환경, 경제환경, 사회환경, 교육환경 비교)
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4,1,figsize=(20, 20), dpi=300)
ax1.set_title("자치구별 주거환경 만족도", fontweight='bold',fontsize = 15)
plt.xticks(rotation=90)
sns.barplot(data = new_면적df, x='구', y ='주거환경', palette='tab10',ax=ax1)
ax1.set_ylim([5,6.75])

ax2.set_title("자치구별 경제환경 만족도", fontweight='bold',fontsize = 15)
sns.barplot(data = new_면적df, x='구', y ='경제환경', palette='tab10', ax=ax2)
ax2.set_ylim([5,7])

ax3.set_title("자치구별 사회환경 만족도", fontweight='bold',fontsize = 15)
sns.barplot(data = new_면적df, x='구', y ='사회환경',  palette='tab10',ax=ax3)
ax3.set_ylim([5,7])

ax4.set_title("자치구별 교육환경 만족도", fontweight='bold',fontsize = 15)
sns.barplot(data = new_면적df, x='구', y ='교육환경' ,  palette='tab10',ax=ax4)
ax4.set_ylim([5,7])

plt.tight_layout()
plt.show()
```







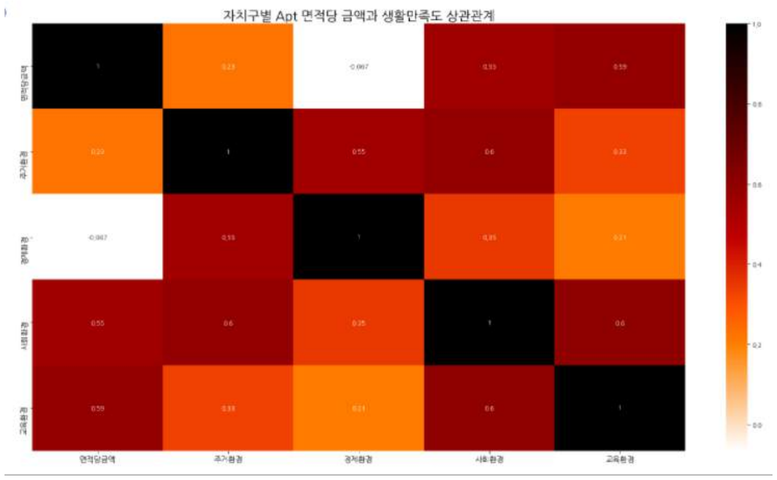


💡 This survey was conducted by City of Seoul in 2020, which contains satisfaction on living area, economic environment, social influence, and education.

d. Determining correlation between the market price and satisfaction scores

▼ code

```
new_면적df.corr()
▶히트맵 그래츠 추출
fig, ax1 = plt.subplots()
fig.set_size_inches(18,10)
sns.heatmap(new_면적df.corr(), annot=True, cmap='gist_heat_r', edgecolors='white')
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.title('자치구별 Apt 면적당 금액과 생활만족도 상관관계', fontsize=20, fontweight='bold')
plt.tight_layout()
plt.show()
```



💡 According to this heatmap, the price per m2 has more correlation with the satisfaction of social environment and education. Therefore, this analysis will proceed with the factors of determining social environment and education, for example, public transportation and education system.

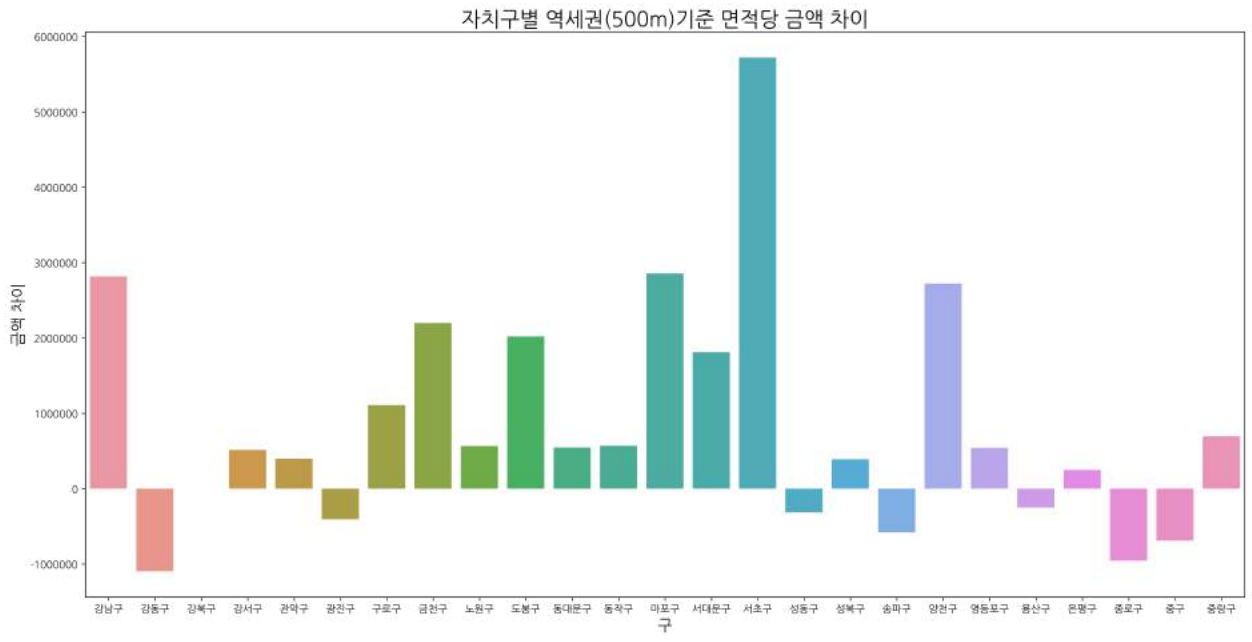
2.3.2 Locational influence

1) **Determining the influence of the public transportation access**

*price per m2 determined by public transportation access by district = (price per m2 by the public-transportation-access area) - (price per m2 by the **non**-public-transportation-access area)*

▼ code

```
▶구별비교
구별역세권df = apt.groupby(['역세권', '구']).mean().reset_index()[['역세권', '구', '면적당금액']]
구별역세권df[구별역세권df['구']=='강남구']
▶금액확인
구별역세권df[구별역세권df['구']=='강남구']['면적당금액'].iloc[1]
gu_lst = 구별역세권df.구.tolist()
diff_gu_subway = pd.DataFrame(columns=['구', '금액차이'])

diff_ = []
for gu in gu_lst:
    df = 구별역세권df[구별역세권df['구']==gu]
    diff_.append(df['면적당금액'].iloc[1]-df['면적당금액'].iloc[0])
diff_gu_subway['구'] = gu_lst
diff_gu_subway['금액차이'] = diff_

▶bar그래프로 시각화 표현하기
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)
sns.barplot(x = diff_gu_subway['구'], y=diff_gu_subway['금액차이'])
ax.set_title("자치구별 역세권(500m)기준 면적당 금액 차이", fontweight='bold',fontsize = 20)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.set_xlabel('구', fontsize=15)
ax.set_ylabel('금액 차이', fontsize=15)
plt.show()
```

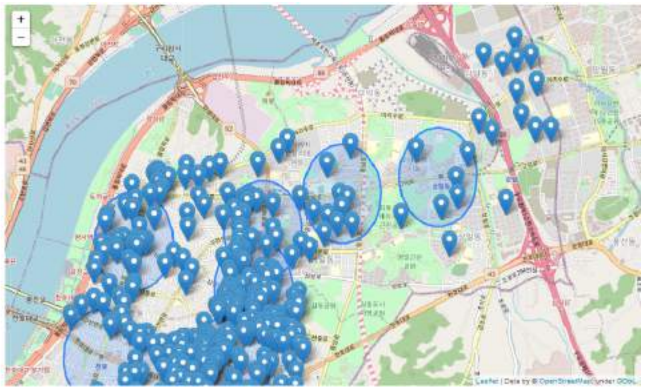


💡 It is to obtain the *price per m2 determined by public transportation access by district.* For example, in Seocho District, there is a huge difference between the apartments **within 500 m from the metro** and the others. However, Gangdong and Jongno District show the opposite trends than we expected, so it needs a further analysis.

2) **Gangdong District**

▼ code

```
▶경도 위도를 이용하여 위치 구하기
subway['buffer500'] = np.nan
for i in range(len(subway)):
    subway['buffer500'][i] = Point(subway.loc[i,'경도'],subway.loc[i,'위도']).buffer(0.005)
▶강동구
gangdong_subway = subway[subway['구']=='강동구']
gangdong_subway.reset_index(drop=True, inplace=True)
▶
gangdong_apt = apt[apt['구']=='강동구']
gangdong_apt.reset_index(drop=True, inplace=True)
▶
gangdong_apt = gangdong_apt.drop_duplicates('건물명', keep='first')
gangdong_apt.reset_index(drop=True, inplace=True)

▶ folium을 이용한 map 그래프 시각화
map = folium.Map(location = [37.5,127], zoom_start =12)
for i in range(len(gangdong_subway)):
    folium.GeoJson(gangdong_subway.buffer500[i]).add_to(map)

for j in range(len(gangdong_apt)):
    folium.Marker((gangdong_apt['위도'][j], gangdong_apt['경도'][j])).add_to(map)

map
```



▼ click

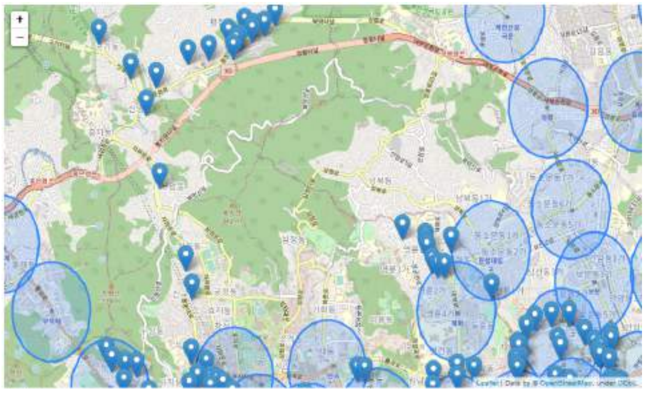
💡 In this district, even there is more than 500 m of distance to access the public transportation, the price per m2 is relatively higher. In this residential area, the apartment composes the biggest village, which is assumedly not affected by the public transportation access.

3) **Jongno District**

▼ code

```
jongro_subway = subway[subway['구']=='종로구']
jongro_subway.reset_index(drop=True, inplace=True)
jongro_apt = apt[apt['구']=='종로구']
jongro_apt = jongro_apt.drop_duplicates('건물명', keep='first')
jongro_apt.reset_index(drop=True, inplace=True)
▶folium 이용한 시각화
map2 = folium.Map(location = [37.5,127], zoom_start =12)
for i in range(len(subway)):
    folium.GeoJson(subway.buffer500[i]).add_to(map2)

for j in range(len(jongro_apt)):
    folium.Marker((jongro_apt['위도'][j], jongro_apt['경도'][j])).add_to(map2)

map2
```

▼ click



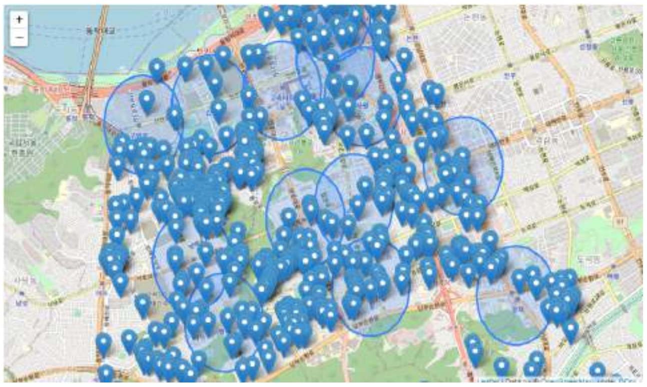
💡 This area is surrounded by mountain, which indicates the general usage of cars, rather than metro.

4) **Seocho District**

▼ code

```
seocho_subway = subway[subway['구']=='서초구']
seocho_subway.reset_index(drop=True, inplace=True)
seocho_apt = apt[apt['구']=='서초구']
seocho_apt = seocho_apt.drop_duplicates('건물명', keep='first')
seocho_apt.reset_index(drop=True, inplace=True)

▶folium을 이용한 map 그래프 확인
map2 = folium.Map(location = [37.5,127], zoom_start =12)
for i in range(len(seocho_subway)):
    folium.GeoJson(seocho_subway.buffer500[i]).add_to(map2)

for j in range(len(seocho_apt)):
    folium.Marker((seocho_apt['위도'][j], seocho_apt['경도'][j])).add_to(map2)

map2
```

▼ click



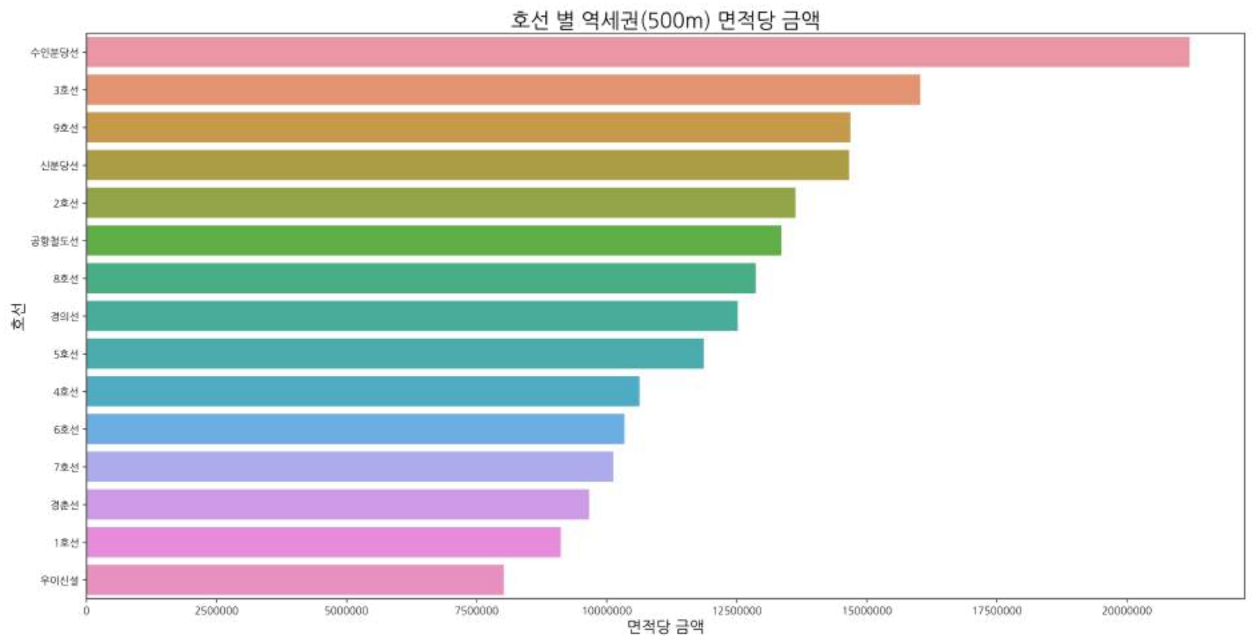
💡 As indicated above, in Seocho District, there is a huge price difference between the apartments **within 500 m from the metro** and the others.

2.3.3 Comparing price per m2 by metro lines

▼ code

```
▶glob 함수 불러오기
from glob import glob
lines = glob('./data/line/*.csv')

line1 = pd.read_csv(lines[0])
line1.면적당금액.mean()

line_name = ['1호선', '2호선','3호선', '4호선', '5호선', '6호선', '7호선', '8호선', '9호선', '경의선', '경춘선', '공항철도선', '수인분당선', '신분당선', '
line_면적당금액=[]

line_df = pd.DataFrame(columns =['호선', '면적당금액'])
for line in lines:
    df = pd.read_csv(line)
    line_면적당금액.append(df.면적당금액.mean())


line_df['호선'] = line_name
line_df['면적당금액'] = line_면적당금액
```

```
line_df.sort_values('면적당금액', ascending=False, inplace=True)

▶호선별 면적당 금액 bar그래프로 나타내기
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)
sns.barplot(x=line_df['면적당금액'], y=line_df['호선'])
ax.set_title("호선 별 역세권(500m) 면적당 금액", fontweight='bold',fontsize = 20)
ax.get_xaxis().get_major_formatter().set_scientific(False)
ax.set_xlabel('면적당 금액', fontsize=15)
ax.set_ylabel('호선', fontsize=15)
plt.show()
```



Pydeck layer 사용한 그래프

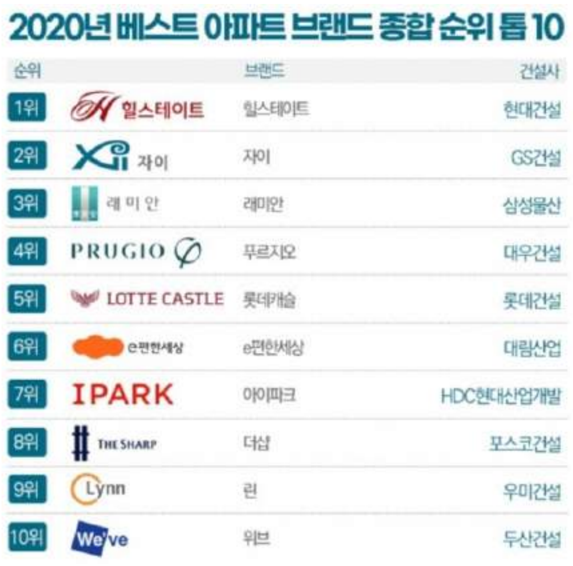
▼ code

```
column_layer1 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[0]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[13,54,146],
    pickable=True,
    auto_highlight=True,
)
column_layer2 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[1]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[51,162,61],
    pickable=True,
    auto_highlight=True,
)

column_layer3 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[2]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[254,91,16],
    pickable=True,
    auto_highlight=True,
)

column_layer4 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[3]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[0,162,209],
    pickable=True,
    auto_highlight=True,
)

column_layer5 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[4]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[139,80,164],
    pickable=True,
    auto_highlight=True,
)

column_layer6 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[5]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[197,92,29],
    pickable=True,
    auto_highlight=True,
)

column_layer7 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[6]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[84,100,13],
    pickable=True,
    auto_highlight=True,
)
column_layer8 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[7]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[241,46,130],
    pickable=True,
    auto_highlight=True,
)
column_layer9 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[8]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[170,152,114],
    pickable=True,
    auto_highlight=True,
)

column_layer10 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[9]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[115,119,166],
    pickable=True,
    auto_highlight=True,
)
column_layer11 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[10]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[50,198,166],
    pickable=True,
    auto_highlight=True,
)

column_layer12 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[11]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[54,129,183],
    pickable=True,
    auto_highlight=True,
)

column_layer13 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[12]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[255,140,0],
    pickable=True,
    auto_highlight=True,
)
column_layer14 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[13]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[200,33,39],
    pickable=True,
    auto_highlight=True,
)

column_layer15 = pdk.Layer(
    "ColumnLayer",
    data=pd.read_csv(lines[14]),
    get_position=["경도", "위도"],
    get_elevation="면적당금액",
    elevation_scale=0.00005,
    radius=50,
    get_fill_color=[69,81,9],
    pickable=True,
    auto_highlight=True,
)


# Set the viewport location
center = [126.96216879726751, 37.50663783146973]
view_state = pdk.ViewState(
    longitude=center[0],
    latitude=center[1],
    zoom=10,
    bearing=15,
    pitch =45
)

# Render
r = pdk.Deck(layers=[column_layer1, column_layer2, column_layer3, column_layer4, column_layer5, column_layer6, column_layer7,
                     column_layer8, column_layer9,column_layer10, column_layer11, column_layer12, column_layer13,column_layer14,column_
              #map_style='mapbox://styles/mapbox/outdoors-v11',
             map_style='mapbox://styles/mapbox/dark-v10',
             mapbox_key = "pk.eyJ1IjoiamNsYXJhODExIiwiYSI6ImNrZzF4bWNhdTBpNnEydG54dGpxNDEwajAifQ.XWxOKQ-2HqFBVBYa-XoS-g"
            )
r.to_html('서울시노선별면적당금액.html')
```

▼ click

2.3.4 The property's own traits - Brand & Year Built

- Top 10 The Most Famous Brands in Korea



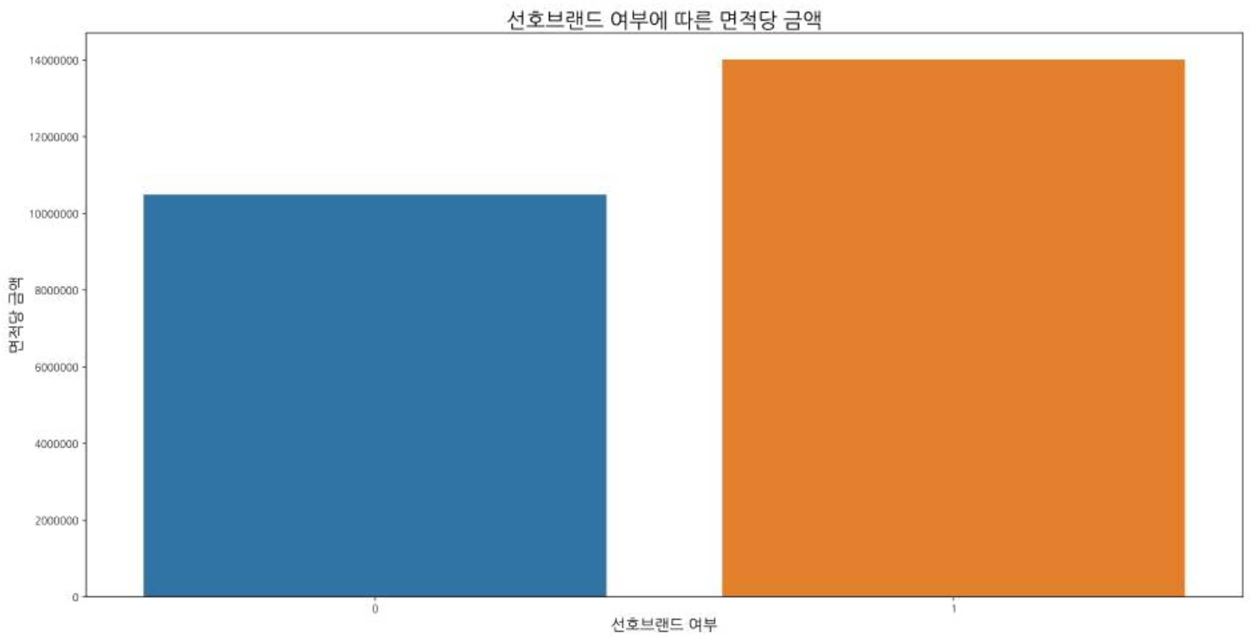
2020년 베스트 아파트 브랜드 종합 순위 톱10

순위	브랜드	건설사
1위	힐스테이트	현대건설
2위	자이	GS건설
3위	래미안	삼성물산
4위	푸르지오	대우건설
5위	롯데캐슬	롯데건설
6위	e편한세상	대림산업
7위	아이파크	HDC현대산업개발
8위	더샵	포스코건설
9위	린	우미건설
10위	위브	두산건설

1) **Using dummy variables (if the brand is in Top 10, then 1, else 0)**

▼ code

```
apt_lst = apt.건물명.unique().tolist()
searching_words = ['아이파크', '자이', '힐스테이트', '래미안', '푸르지오', '편한', '더샵', '위브', '롯데캐슬']
brand_lst = []
for name in apt_lst:
    for word in searching_words:
        if word in name:
            brand_lst.append(name)

brand_df = pd.DataFrame(columns=['건물명', '선호브랜드'])
brand_df.건물명 = brand_lst

nobrand_lst = list(set(apt.건물명.unique().tolist()) - set(brand_lst))

nobrand_df = pd.DataFrame(columns=['건물명', '선호브랜드'])
nobrand_df.건물명 = nobrand_lst
nobrand_df['선호브랜드'] = 0

brand = pd.concat([brand_df, nobrand_df])

apt = pd.merge(apt, brand, on='건물명')

선호브랜드df = apt.groupby(['선호브랜드']).mean().reset_index()[['면적당금액']]

▶선호브랜드를 1 , 해당하지 않는 브랜드를 0으로 표현한 시각화
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)

sns.barplot(x=선호브랜드df.index, y=선호브랜드df['면적당금액'])
ax.set_title("선호브랜드 여부에 따른 면적당 금액", fontweight='bold',fontsize = 20)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.set_xlabel('선호브랜드 여부', fontsize=15)
ax.set_ylabel('면적당 금액', fontsize=15)
plt.show()
```



💡 According to this dummy variable, it is understood the market price is influenced by the brand of apartment. In detail, the average price of the brands is higher by 3,500,000 KRW.

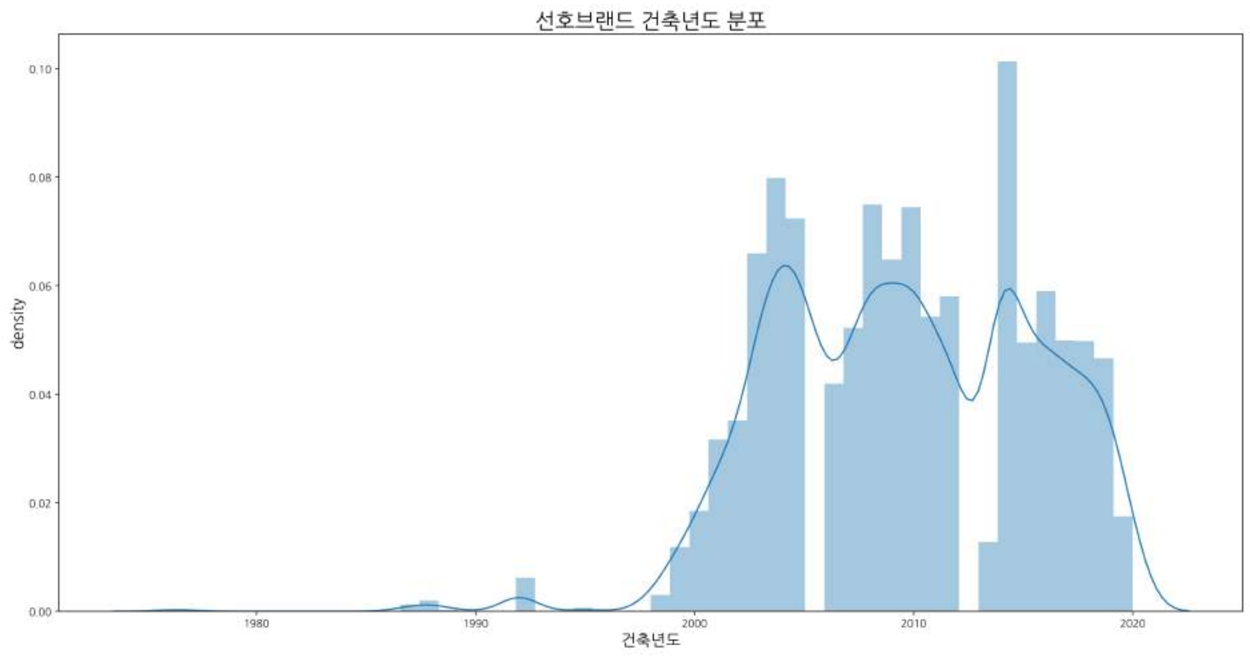
2) **Newly built?**

Combining the analysis we had with the Top 10 Brands, it is to determine whether they are built recently.

- **Year built of the Top 10 brands apartments**

▼ code

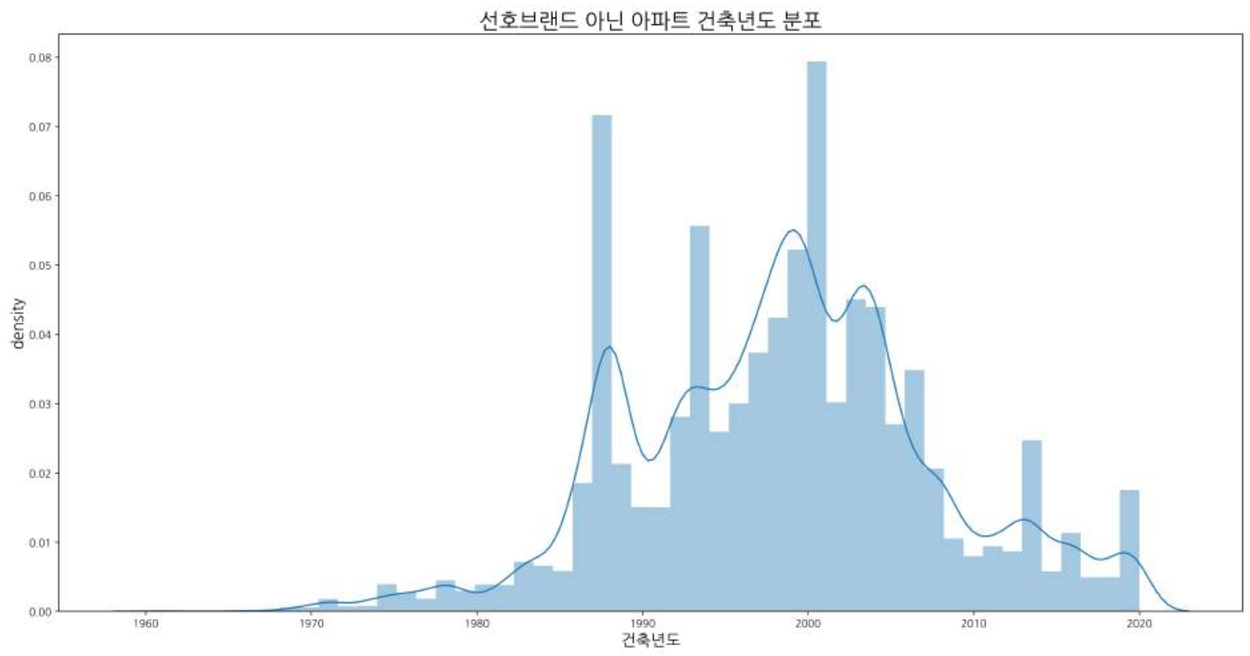
```
apt[apt['선호브랜드']==1].건축년도.describe()

▶ 선호브랜드 건축년도 분포
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)

sns.distplot(apt[apt['선호브랜드']==1].건축년도)
ax.set_title("선호브랜드 건축년도 분포", fontweight='bold',fontsize = 20)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.set_xlabel('건축년도', fontsize=15)
ax.set_ylabel('density', fontsize=15)
plt.show()
```



- **Year built of the Non-Top 10 brands apartments**

▼ code

```
apt[apt['선호브랜드']==0].건축년도.describe()

▶선호브랜드 아닌 아파트 건축년도 분포
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)

sns.distplot(apt[apt['선호브랜드']==0].건축년도)
ax.set_title("선호브랜드 아닌 아파트 건축년도 분포", fontweight='bold',fontsize = 20)
ax.get_yaxis().get_major_formatter().set_scientific(False)
ax.set_xlabel('건축년도', fontsize=15)
ax.set_ylabel('density', fontsize=15)
plt.show()
```

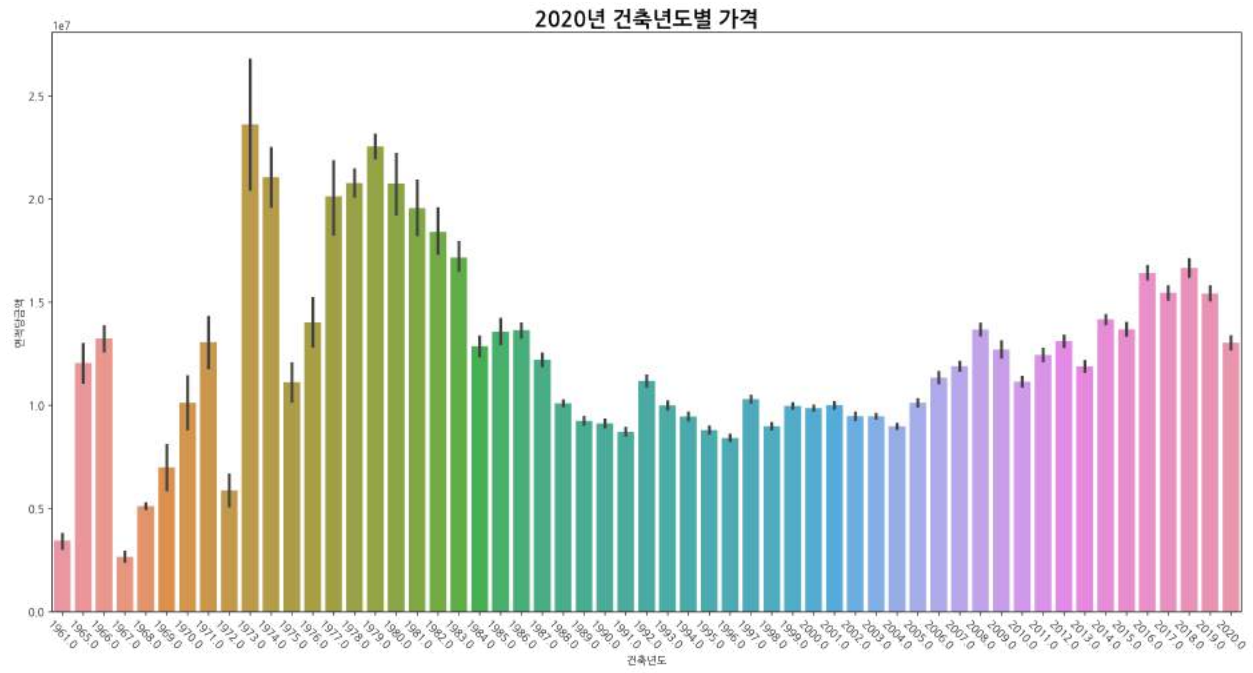


💡 The Top 10 Brands apartments are relatively recently built comparing to the non-top-10.

- **The price per m2 by year built**

▼ code

```
fig, ax = plt.subplots(1,figsize=(20, 10), dpi=300)
plt.xticks(rotation=-45)
sns.barplot(y = apt['면적당금액'], x = apt['건축년도'])
ax.set_title("2020년 건축년도별 가격", fontweight='bold',fontsize = 20)
plt.show()
```



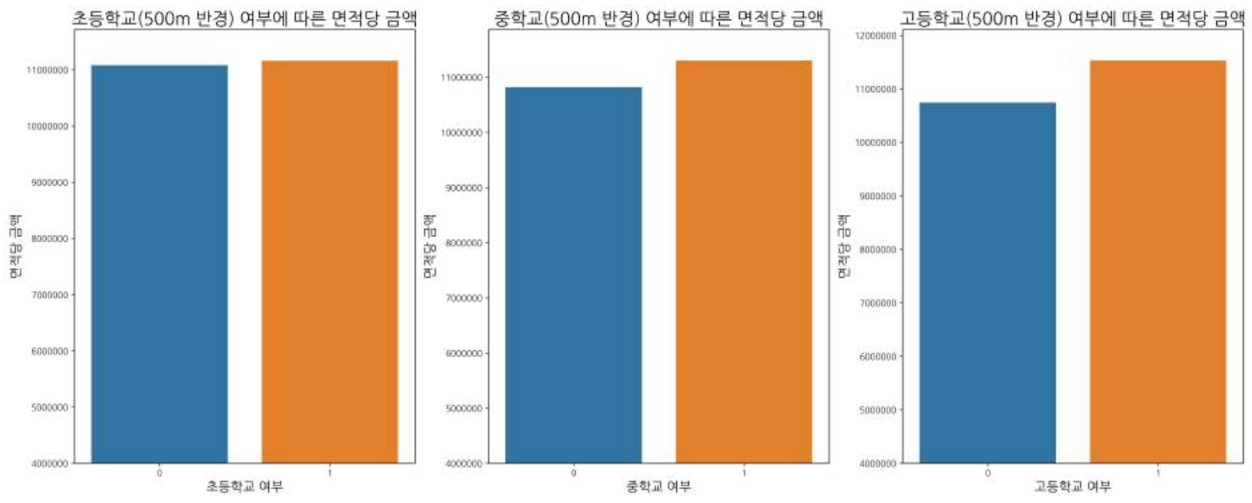
2.3.5 Locational Influence - Education

As a locational influence factor, it is to determine the influence of the public school access (within 500 m) to the price per m2.

Primary school, Middle school and High school

▼ code

```
apt[['면적당금액', 'elementary거리']].corr()

apt[['면적당금액', 'middleschool거리']].corr()

apt[['면적당금액', 'highschool거리']].corr()
```



The price difference by education access

- Primary school: 78,482.48 KRW
- Middle school : 485,302.68 KRW
- High school : 788,418.36 KRW

According to this analysis, it is seen that the access to public schools affects the price per m2 of property. However, in cases of primary schools, it is regulated by the government to have more than 2 public schools within 1,500 m. This is why the difference is higher in the access to middle schools and high schools.

3. Modeling

3.1 Features

The research question is:

What are the determining factors of skyrocketing market price of real estate in 2020?

By far, this report has analysed the locational influence, including access to public schools, public transportation and the satisfaction score by residents, the qualities of the building itself, using year built, area and the brand value of the apartments, and the demographic information.

Now, using the highly correlated variables suggested below, this report plans to sheds some lights on the prediction of market price of apartments in Seoul in 2021.

Variables

Aa Variable	Name of Variable	Description

Variable	Name of Variable	Description
Target Variable	아파트 가격	Price per m2 (KRW)
Explanatory Variables	지하철역 거리	Distance between the property and public transportation (meter)
Untitled	건축년도	Year Built
Untitled	건물면적	Area (m2)
Untitled	선호 브랜드 더미	Dummy variable; if it's Top 10 brands, 1, else 0
Untitled	초등학교 거리	Distance between the property and primary school (km)
Untitled	중학교 거리	Distance between the property and middle school (km)
Untitled	고등학교 거리	Distance between the property and high school (km)
Untitled	어린이 비율	Proportion of children in the district (%)
Untitled	청소년 비율	Proportion of youth in the district (%)
Untitled	중장년 비율	Proportion of adults in the district (%)
Untitled	노인 비율	Proportion of the elders in the district (%)
Untitled	구, 행정동	Name of district

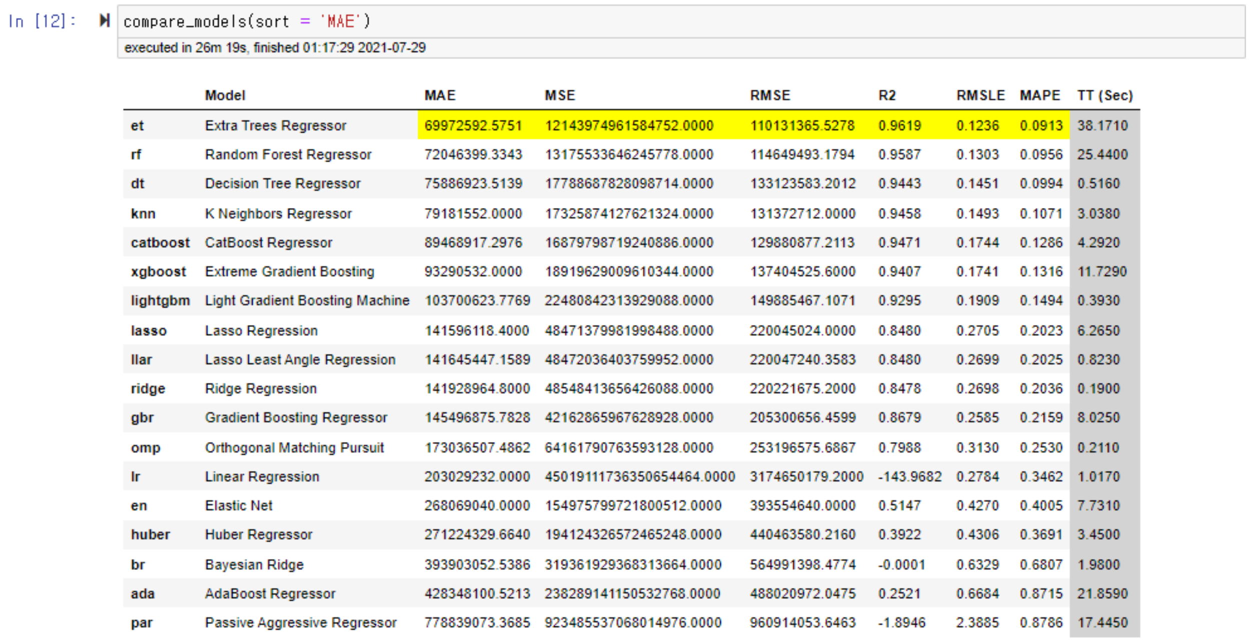
3.2 Model Selection

- Using Auto ML, we analysed the various models and its efficiency
- Using Pycaret, comparing regressive model

▼ code

```
# pycaret의 회귀 모델을 import
from pycaret.regression import *

# pycaret 환경 설정(train data와 target 설정, normalize 여부, normalization 방법 설정, categorical_column 지정 등)
setup(train_data, target='amount', train_size = 0.7, normalize = True, normalize_method='zscore', categorical_features=cat_columns2)

# 환경설정 된 DATA를 원하는 metric을 기반으로 모델 성능 비교
compare_models(sort='MAE')

# 원하는 model 생성, compare_models에서 기본적으로 cross_validation 10 fold로 비교해주기 때문에 false로 생성해줌
et=create_model('et', cross_validation=False)
```

1) **Comparing Models after Z-score Normalisation**



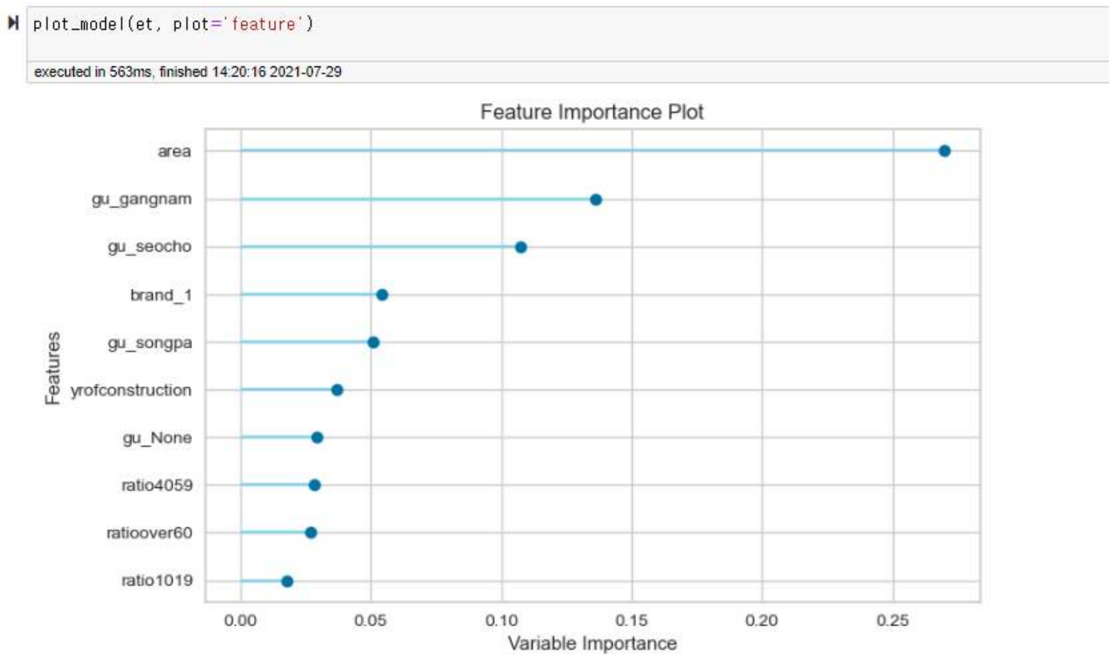
```
In [12]:  compare_models(sort = 'MAE')
executed in 26m 19s, finished 01:17:29 2021-07-29
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	69972592.5751	12143974961584752.0000	110131365.5278	0.9619	0.1236	0.0913	38.1710
rf	Random Forest Regressor	72046399.3343	13175533646245778.0000	114649493.1794	0.9587	0.1303	0.0956	25.4400
dt	Decision Tree Regressor	75886923.5139	17788687828098714.0000	133123583.2012	0.9443	0.1451	0.0994	0.5160
knn	K Neighbors Regressor	79181552.0000	17325874127621324.0000	131372712.0000	0.9458	0.1493	0.1071	3.0380
catboost	CatBoost Regressor	89468917.2976	16879798719240886.0000	129880877.2113	0.9471	0.1744	0.1286	4.2920
xgboost	Extreme Gradient Boosting	93290532.0000	18919629009610344.0000	137404525.6000	0.9407	0.1741	0.1316	11.7290
lightgbm	Light Gradient Boosting Machine	103700623.7769	22480842313929088.0000	149885467.1071	0.9295	0.1909	0.1494	0.3930
lasso	Lasso Regression	141596118.4000	48471379981998488.0000	220045024.0000	0.8480	0.2705	0.2023	6.2650
llar	Lasso Least Angle Regression	141645447.1589	48472036403759952.0000	220047240.3583	0.8480	0.2699	0.2025	0.8230
ridge	Ridge Regression	141928964.8000	48548413656426088.0000	220221675.2000	0.8478	0.2698	0.2036	0.1900
gbr	Gradient Boosting Regressor	145496875.7828	42162865967628928.0000	205300656.4599	0.8679	0.2585	0.2159	8.0250
omp	Orthogonal Matching Pursuit	173036507.4862	64161790763593128.0000	253196575.6867	0.7988	0.3130	0.2530	0.2110
lr	Linear Regression	203029232.0000	4501911173635065464.0000	3174650179.2000	-143.9682	0.2784	0.3462	1.0170
en	Elastic Net	268069040.0000	154975799721800512.0000	393554640.0000	0.5147	0.4270	0.4005	7.7310
huber	Huber Regressor	271224329.6640	194124326572465248.0000	440463580.2160	0.3922	0.4306	0.3691	3.4500
br	Bayesian Ridge	393903052.5386	319361929368313664.0000	564991398.4774	-0.0001	0.6329	0.6807	1.9800
ada	AdaBoost Regressor	428348100.5213	238289141150532768.0000	488020972.0475	0.2521	0.6684	0.8715	21.8590
par	Passive Aggressive Regressor	778839073.3685	923485537068014976.0000	960914053.6463	-1.8946	2.3885	0.8786	17.4450

💡 **Extra Trees Regressor shows the highest explanation with 69,970,000 KRW of MAE (mean absolute error), and the R2 score is the highest with 96%.**

💡 **Random Forest Regressor follows with 95% of R2 score.**

2) Feature importance plot of Extra Trees Regressor

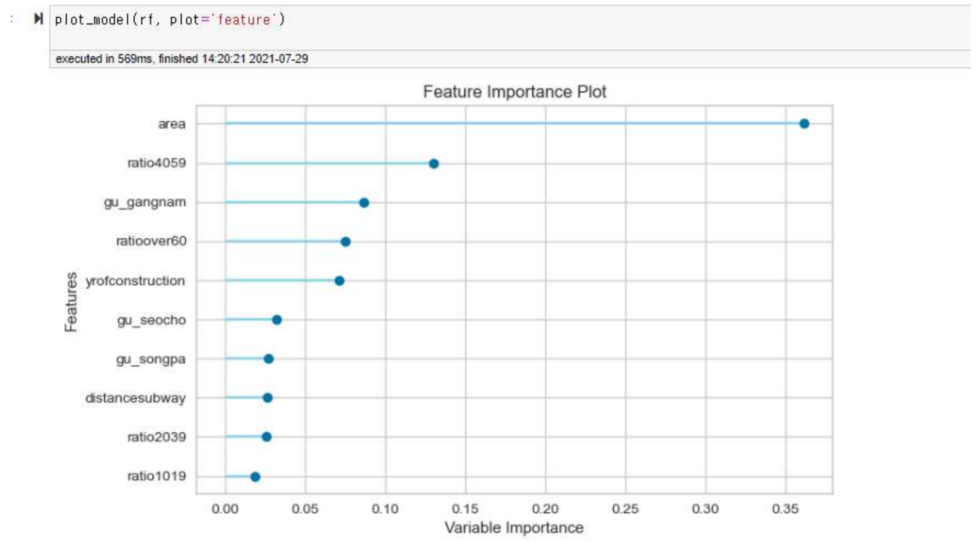


```
plot_model(et, plot='feature')
```

```
executed in 563ms, finished 14:20:16 2021-07-29
```

참고) gu_None = Yongsan Gu

3) Feature Importance plot of Random Forest Regressor



```
plot_model(rf, plot='feature')
```

```
executed in 569ms, finished 14:20:21 2021-07-29
```

💡 According to the **Extra Trees Regressor,** among the variables, area and the districts of Gangnam, Seocho, Songpa and Yongsan are the most important ones. The brand of the apartment is also an important factor of market values of real estates.

💡 In **Random Forest Regressor** also indicates 'area' as the most important feature determining the price of real estate, but most interestingly, the proportion of adults in age 40-59 also shows its significance.

4. Conclusion

This analysis is to answer the question of

What would be the determining factors of current surging market price of real estate in Seoul?
Through machine learning, would it be possible to predict the transaction price of apartments?

According to the Extra Trees Regressor, this report suggests *the area* and *locational factors* as the most important factors of determining 2020 real estate market price. However, our assumption of correlation of the price per m2 and access to public transportation or public schools has not been confirmed.

Then, what insights can we attain from this analysis?

First, the public transportation access has relatively less affected the price per m2 of apartments in Seoul, rather it varies by districts.

Second, the districts of Gangnam, Seocho, Songpa and Yongsan can be the most important factors of determining the market price of Seoul in 2020. It indicates the so-called **Gangnam 3 districts** (the districts of Gangnam, Seocho, Songpa) market power still exists.

Also, the top 10 brands of the apartment can also be an important factor of market values of real estates.

Still, there are several limitations requiring further research:

- More various variables can be used: the area of parking lot, the green area, concert halls etc.
- It could have been time series analysis: The analysis was limited in the year of 2020, however, the panel analysis would be more powerful to forecast future price of real estate in Seoul.