

# 맛집 데이터와 공공데이터를 이용한 서울시 자치구별 맛집 상권분석

데이터시각화 프로젝트

2조

---

김승규  
정현진  
김경민  
정서연  
이선희

# 구성원 및 역할

## 팀장 김승규

1. 구의 인구수와 맛집 수 관계
2. 구별 맛집 가격대 점수
  - 2-1. 구별 메뉴 가격대 점수
3. 구의 소득세와 맛집 가격 관계
  - 3-1. 구의 소득세와 메뉴가격 관계

## 팀원 정서연

1. 구별 소득세와 맛집 평균가격의 관계
2. 구별 GDP와 맛집 가격점수의 관계

## 팀원 김경민

1. 인구수와 리뷰수의 상관관계
2. 구 별 GDP와 맛집의 평균가격 상관관계
3. 구 별 GDP와 가격점수의 상관관계

## 팀원 이선희

가격대별 맛집 만족도 비교

- 1)가격대별 맛 평가(맛있다,괜찮다, 별로다) 비율
- 2)가격대별 맛집의 평균 평점 비교

## 팀원 정현진

1. 구 별 맛집 수
2. 음식 종류 별 맛집 수
3. 가격대 별 맛집 수
4. 서울 지도에 맛집의 좌표 값을 이용해 마커를 찍고  
마커 안에 간단한 정보를 삽입
5. 구 별 맛집 수 단계구분도 표현
6. 구별 맛집 수와 사업체 종사자수 간의 상관 관계
7. 구별 맛집 수와 사업체 수 간의 상관 관계
8. 인접역에서의 거리에 따른 맛집 수
9. 인접대학교에서의 거리에 따른 맛집 수
10. 구의 소득세와 맛집 가격 관계 단계구분도
11. 구의 소득세와 메뉴가격 관계 단계구분도

# CONTENTS

## 01

### 프로젝트 개요

1-1 프로젝트 기획 배경 및  
목표

1-2 구성원 및 역할

---

## 02

### 데이터 프로세싱

2-1 데이터 수집

2-2 데이터 전처리

---

## 03

### 데이터 시각화

3-1 데이터 시각화

3-2 시각화 결과를 통한  
인사이트 도출

---

## 04

### 기대효과 및 후기

4-1 향후 개선사항 및  
기대효과

4-2 느낀점 및 후기

---

# 01 프로젝트 개요

# 01 프로젝트 개요

## 1-1 프로젝트 기획 배경



출처: 행정안전부 / 분석 및 제공: 상가정보연구소

# 01 프로젝트 개요

## 1-2 프로젝트 목표

01

서울시 다양한 공공데이터 활용 및 크롤링을 통한  
맛집데이터 수집

02

맛집 데이터와 공공데이터 간의 다양한  
상관관계 분석

03

서울시 맛집 상권분석에 필요한 다양한 인사이트  
도출 및 창업 아이디어 제안

## 02 데이터 프로세싱

## 02 데이터 프로세싱

### 2-1 데이터 수집



서울 열린데이터 광장  
SEOUL OPEN DATA PLAZA

<https://data.seoul.go.kr/>

#### 공공데이터 수집 List

1. 서울시 자치구 별 인구수
2. 서울시 자치구 별 소득세
3. 서울시 자치구 별 GDP
4. 서울시 자치구 별 사업체 수
5. 서울시 자치구 별 사업체 종사자 수
6. 서울시 소재 지하철역 주소
7. 서울시 소재 대학교 주소



## 02 데이터 프로세싱

### 2-2 웹 크롤링

1. 망고 플레이트 서비스에서 서울시 구별로 필요한 데이터의 목록을 빈리스트로 생성

데이터 수집 사이트



<https://www.mangoplate.com/>

크롤링에 사용한 프레임워크



*Selenium*

## 02 데이터 프로세싱

### 2-2 웹 크롤링

#### 2. 서울시 25개 구 리스트 생성



#서울시 구 리스트

```
gu_list=['도봉구','노원구','강북구','은평구','종로구','성북구','동대문구','중랑구','서대문구',\
        '마포구','중구','용산구','성동구','광진구','강동구','송파구','강남구',\
        '강서구','양천구','영등포구','동작구','서초구','관악구','금천구','구로구']
```

## 02 데이터 프로세싱

### 2-2 웹 크롤링

#### 3. 크롤링 진행

```
xpath = '//*[@id="main-search"]'
g = driver.find_element(By.XPATH, xpath)

for u in gu_list:
    base_url = "https://www.mangoplate.com/search/"+u+"?keyword="+u+"&page="
    for n in range(1,2):
        urls = base_url + str(n)
        driver.get(urls)
        time.sleep(7)
        for i in range(1,11):
            for j in range(1,3):
                xpath='/html/body/main/article/div[2]/div/div/section/div[3]/ul/li[{0}]/div[{1}]/figure/figcaption/div/a'.format(i,j)
                time.sleep(7)
                q=driver.find_element(By.XPATH, xpath)
                driver.execute_script("arguments[0].click();",q)
                time.sleep(7)

            #식당명

        try:
            restaurant_name.append(driver.find_element(By.XPATH, '//h1').text)
        except:
            restaurant_name.append(' ')
```

## 02 데이터 프로세싱

### 2-2 웹 크롤링

#### 3. 데이터 정제

##### ■ 불필요한 반복문자 삭제처리

```
df1['주소']=df1['주소'].str.replace('주소','')
df1['구']=df1['구'].str.replace('서울시','')
df1['음식종류']=df1['음식종류'].str.replace('음식 종류','')
df1['가격대']=df1['가격대'].str.replace('가격대','')
```

##### ■ 데이터 병합 후 인덱스 재설정

```
df=pd.concat([df1,df2,df3,df4,df5])
df.reset_index(drop=True,inplace=True)
```

##### ■ 데이터 밀리는 부분 수정 전처리

```
wrong = df['음식종류'].str.contains("가격대")
wrong_df = df[wrong]
index = wrong_df.index.tolist()
```

```
for i in index:
    a = df.loc[i, '음식종류']

    df.loc[i, '가격대'] = a
    df.loc[i, '음식종류'] = 0
```

# 02 데이터 프로세싱

## 2-2 웹 크롤링

### 4. 데이터프레임 생성

	식당명	별점	구	주소	음식종류	가격대	리뷰날짜	리뷰평가	리뷰수	맛있다수	괜찮다수	별로다수
0	이코이	4.4	도봉구	서울특별시 도봉구 도봉로112길 20	돈부리 / 일본 카레 / 벤토	만원-2만원	2022-05-29	맛있다	37	32.0	4.0	1.0
1	패엠베이커리	4.3	도봉구	서울특별시 도봉구 도봉로116길 4	카페 / 디저트	만원 미만	2022-04-18	맛있다	7	7.0	0.0	0.0
2	스마일닭강정	4.3	도봉구	서울특별시 도봉구 도봉로110아길 7 1F	닭 / 오리 요리	만원 미만	2022-03-19	맛있다	7	7.0	0.0	0.0
3	코노하카레	4.2	도봉구	서울특별시 도봉구 우이천로20길 7 103동상가 1F	돈부리 / 일본 카레 / 벤토	만원-2만원	2022-02-07	맛있다	38	32.0	6.0	0.0
4	버거브라더	4.2	도봉구	서울특별시 도봉구 마들로 657 1F	브런치 / 버거 / 샌드위치	만원 미만	2022-06-30	맛있다	8	7.0	1.0	0.0

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   식당명      5000 non-null   object
1   별점        3417 non-null   float64
2   구          5000 non-null   object
3   주소        5000 non-null   object
4   음식종류    5000 non-null   object
5   가격대      5000 non-null   object
6   리뷰날짜    5000 non-null   object
7   리뷰평가    5000 non-null   object
8   리뷰수      5000 non-null   int64
9   맛있다수    4931 non-null   float64
10  괜찮다수   4931 non-null   float64
11  별로다수    4931 non-null   float64
dtypes: float64(4), int64(1), object(7)
```

열 개수: 12개  
행 개수: 5000개

## 03 데이터 시각화

## 03 데이터 시각화

**#1**

기본 시각화

**#2**

인구

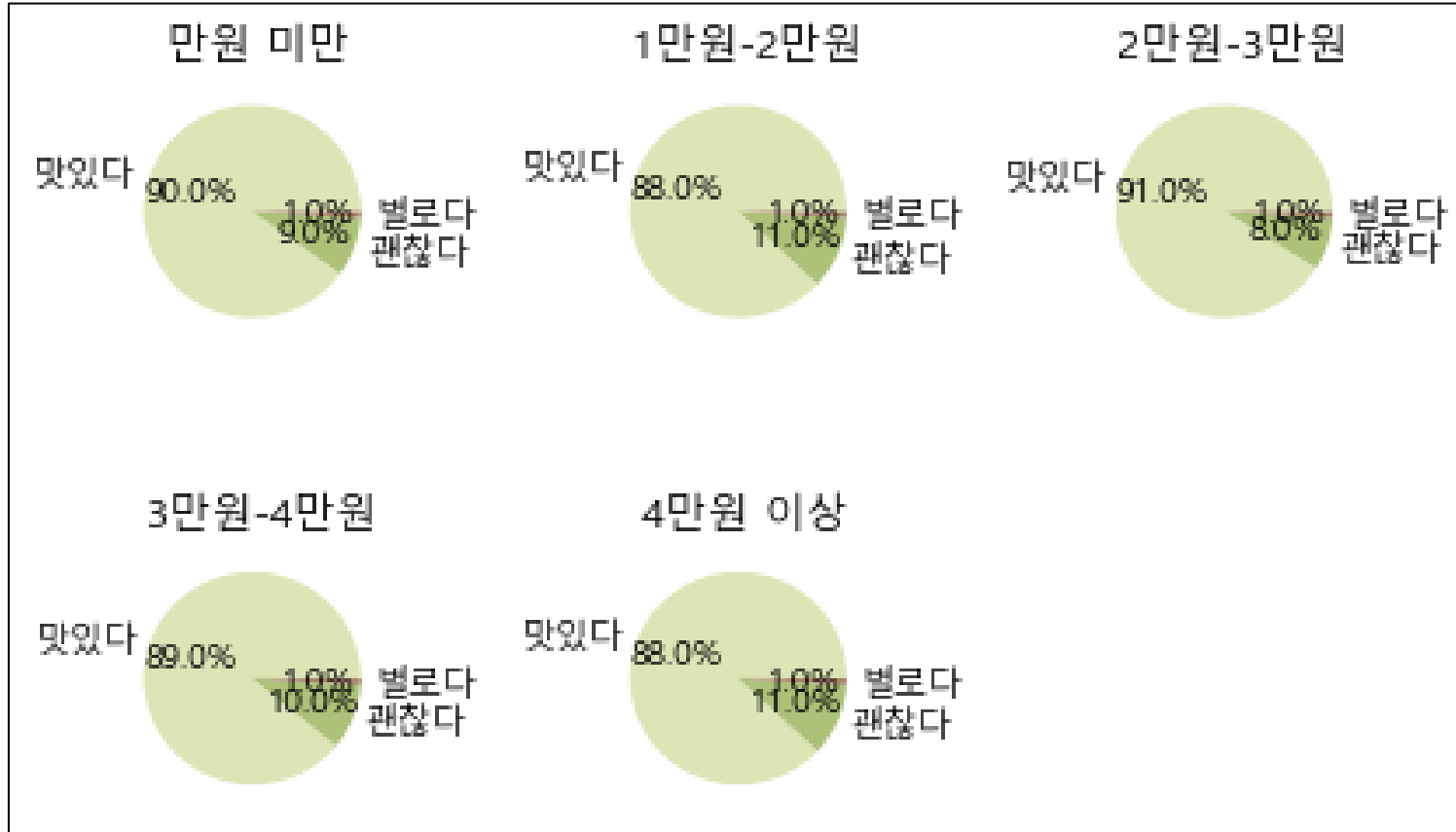
**#3**

활동인구/  
사업체/인프라

**#4**

소득(GDP)

## 03 데이터 시각화



■가중치를 적용하여 새로운 평점 산출

(별점\*맛있다 수 - 2\*별점\*별로다 수)/전체 리뷰수

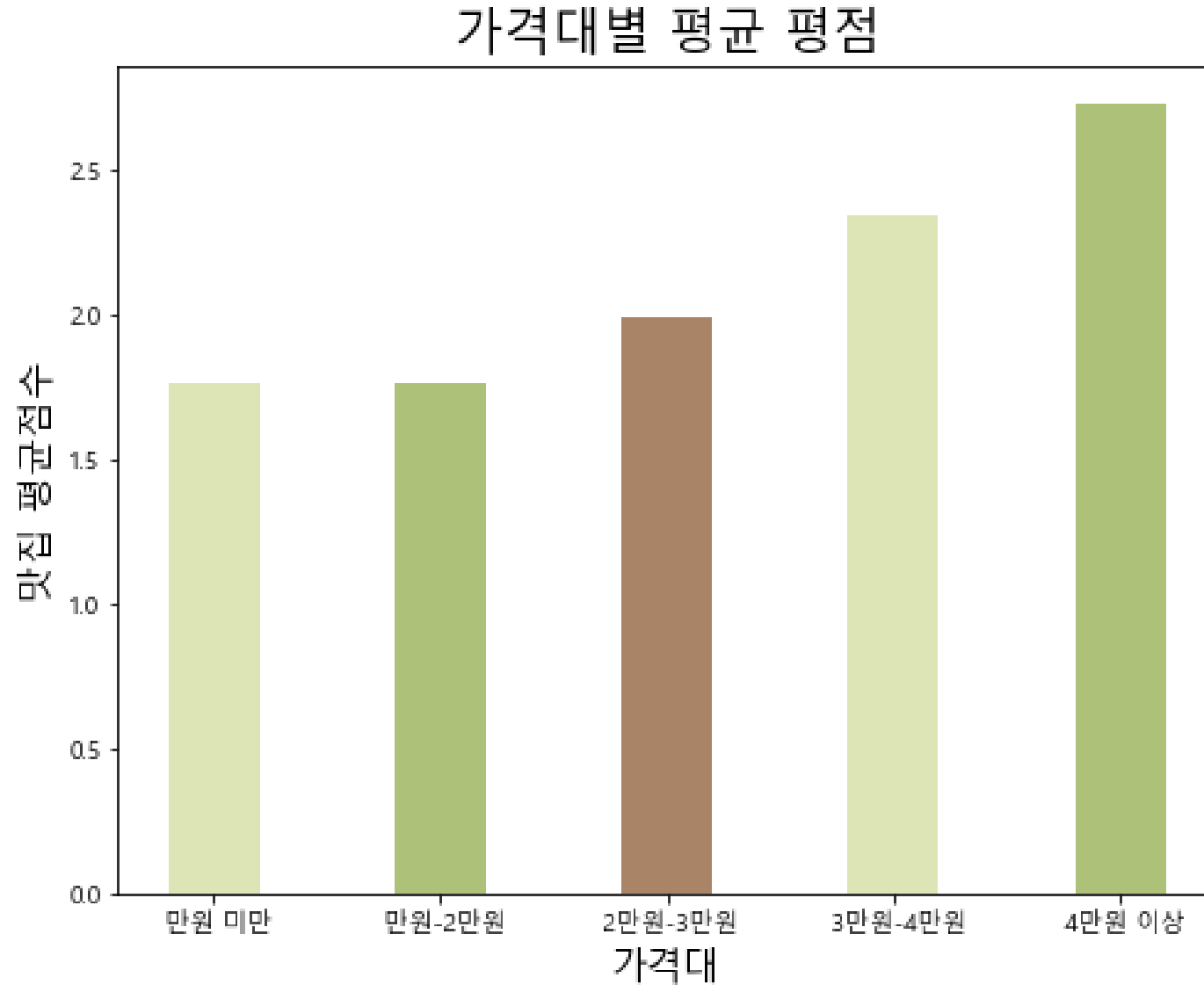
< 가격대별 맛 평가 (맛있다,괜찮다,별로다) 비율>



## 03 데이터 시각화

	식당명	별점	구	주소	음식종류	가격대	리뷰날짜	리뷰평가	리뷰수	맛있다수	괜찮다수	별로다수	최종평점
0	도이칠란드 박	4.6	성북구	서울특별시 성북구 숭례로6길 30-15 1F	다국적음식	만원-2만원	2022-07-04	맛있다	21.0	19.0	0.0	2.0	3.285714
1	쁘르께노 스페니쉬비스트로	4.6	성북구	서울특별시 성북구 동소문로6길 4-21 1F	다국적음식	만원-2만원	2022-07-22	맛있다	7.0	7.0	0.0	0.0	4.600000
2	공푸	4.5	성북구	서울특별시 성북구 삼선교로24길 29	중식	만원 미만	2022-07-07	별로	91.0	76.0	11.0	4.0	3.362637
3	성북동집	4.5	성북구	서울특별시 성북구 성북로24길 4	한식	만원-2만원	2022-07-28	맛있다	41.0	35.0	6.0	0.0	3.841463
4	계모임	4.5	성북구	서울특별시 성북구 보문로30라길 5-10 1F	한식	만원-2만원	5 일 전	맛있다	10.0	10.0	0.0	0.0	4.500000

## 03 데이터 시각화



가격대



맛집 평균점수



## 03 데이터 시각화

### ■ 가격점수 도출 이유

기존에 수집한 가격대별 맛집수의 절대적인 데이터에 가중치를 주어 상대적인 값(가격점수)로 변환한 뒤 데이터 분석에 활용

	10000원 미만	20000원 미만	30000원 미만	40000원 미만	40000원 이상	맛집수	가격 점수
구							
강남구	30	59	34	22	55	200	3.065000
강동구	88	69	19	5	2	183	1.710383
강북구	95	67	14	3	1	180	1.600000
강서구	82	78	21	3	10	194	1.871134
관악구	91	79	18	1	2	191	1.659686
광진구	102	76	12	4	4	198	1.646465
구로구	68	90	27	3	1	189	1.830688
금천구	87	71	11	2	4	175	1.657143

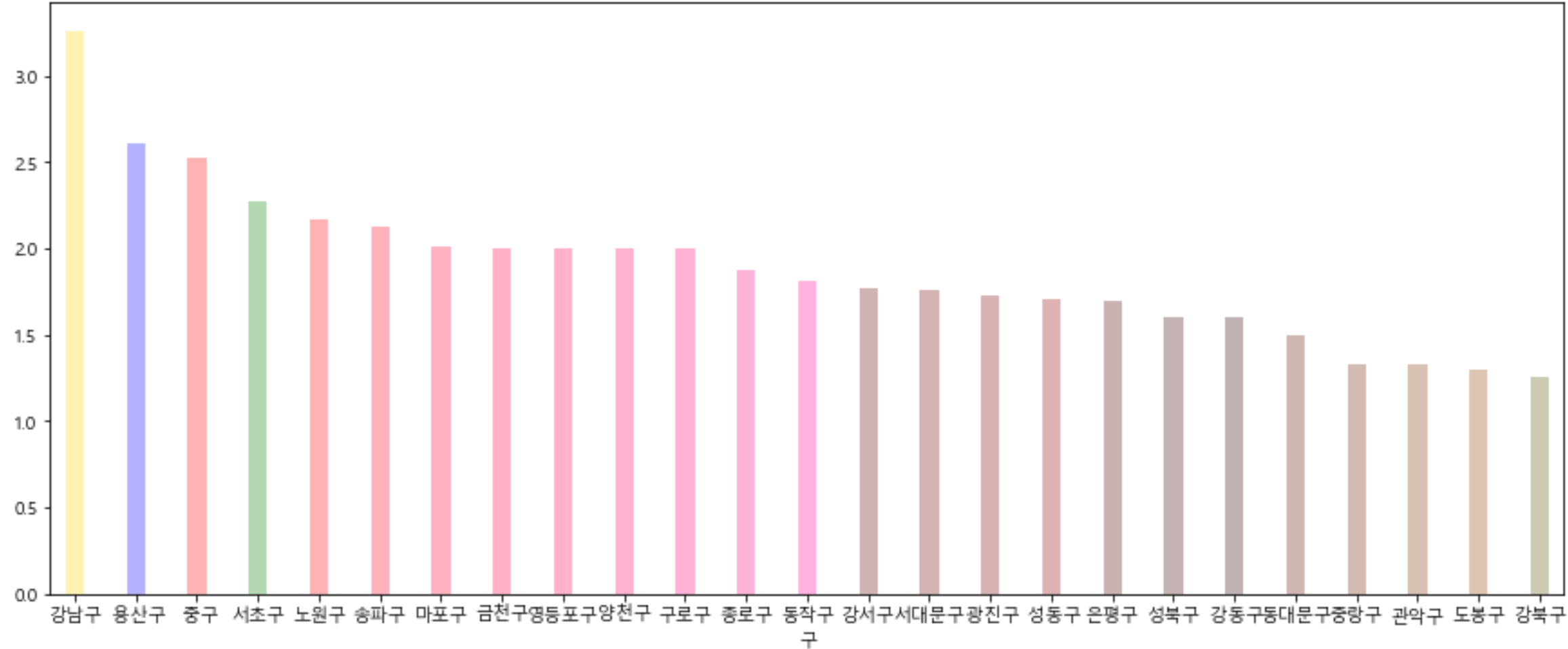
### ■ 가격점수 계산식

```
Fin_df['가격 점수'] = (Fin_df['10000원 미만'] + 2*Fin_df['20000원 미만'] \
+ 3*Fin_df['30000원 미만'] + 4*Fin_df['40000원 미만'] + 5*Fin_df['40000원 이상']) / Fin_df['맛집수']
```

# 03 데이터 시각화

<구 별 맛집 가격대 비교>

구별 맛집 가격대 비교



## 03 데이터 시각화

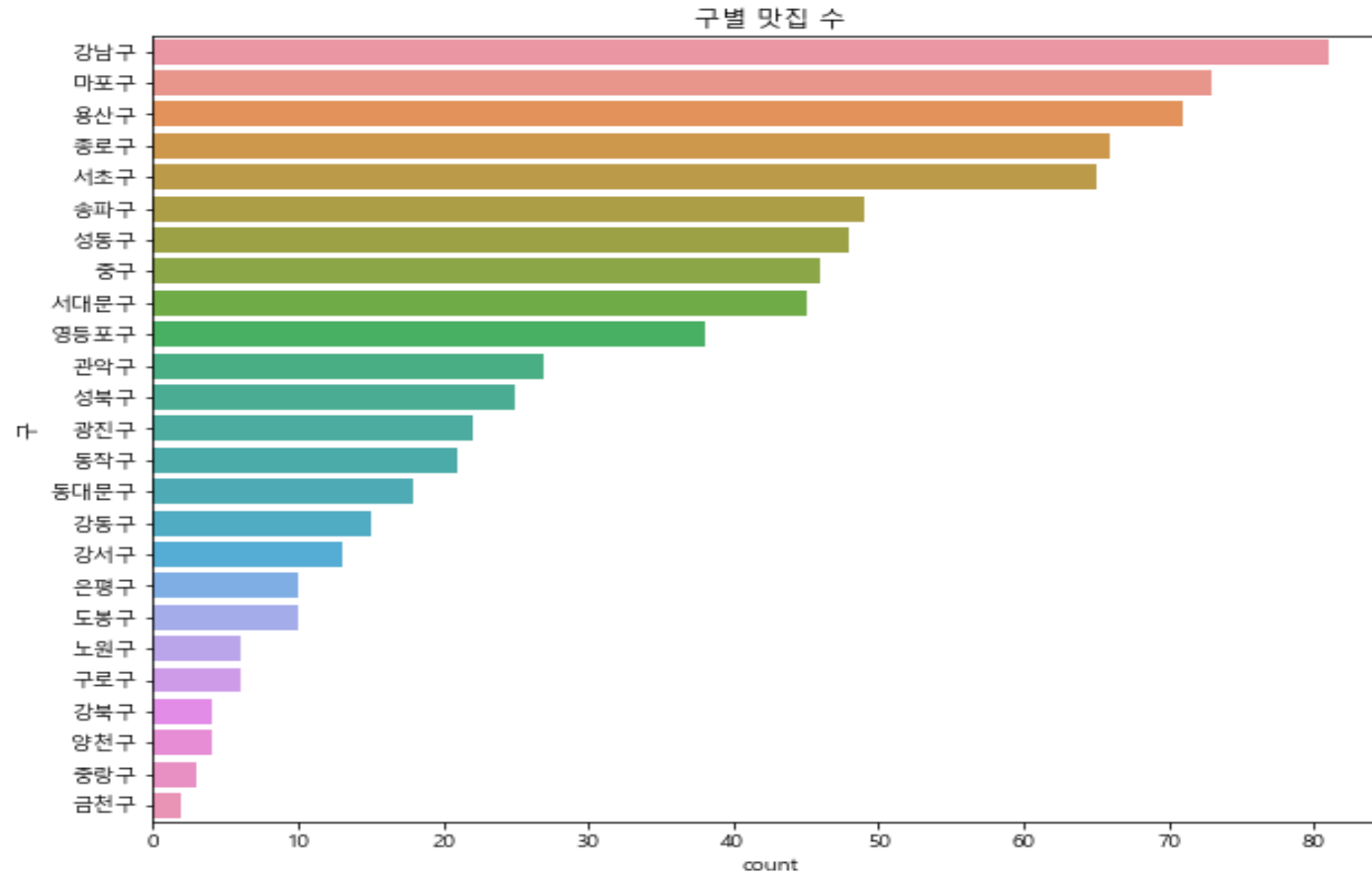
**#1**

기본 시각화

# #1

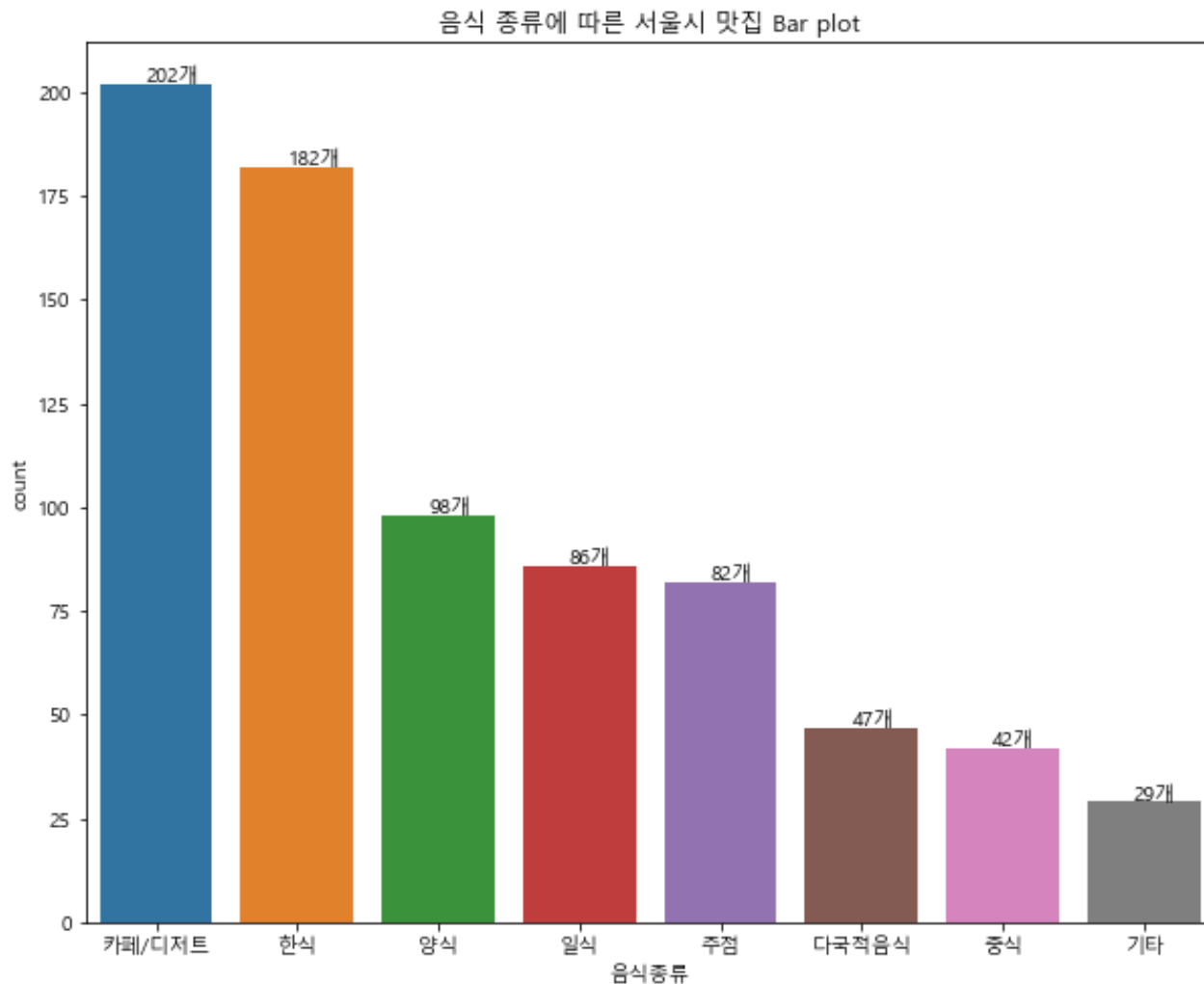
## 기본시각화

<서울시 구별 맛집 개수 분포>



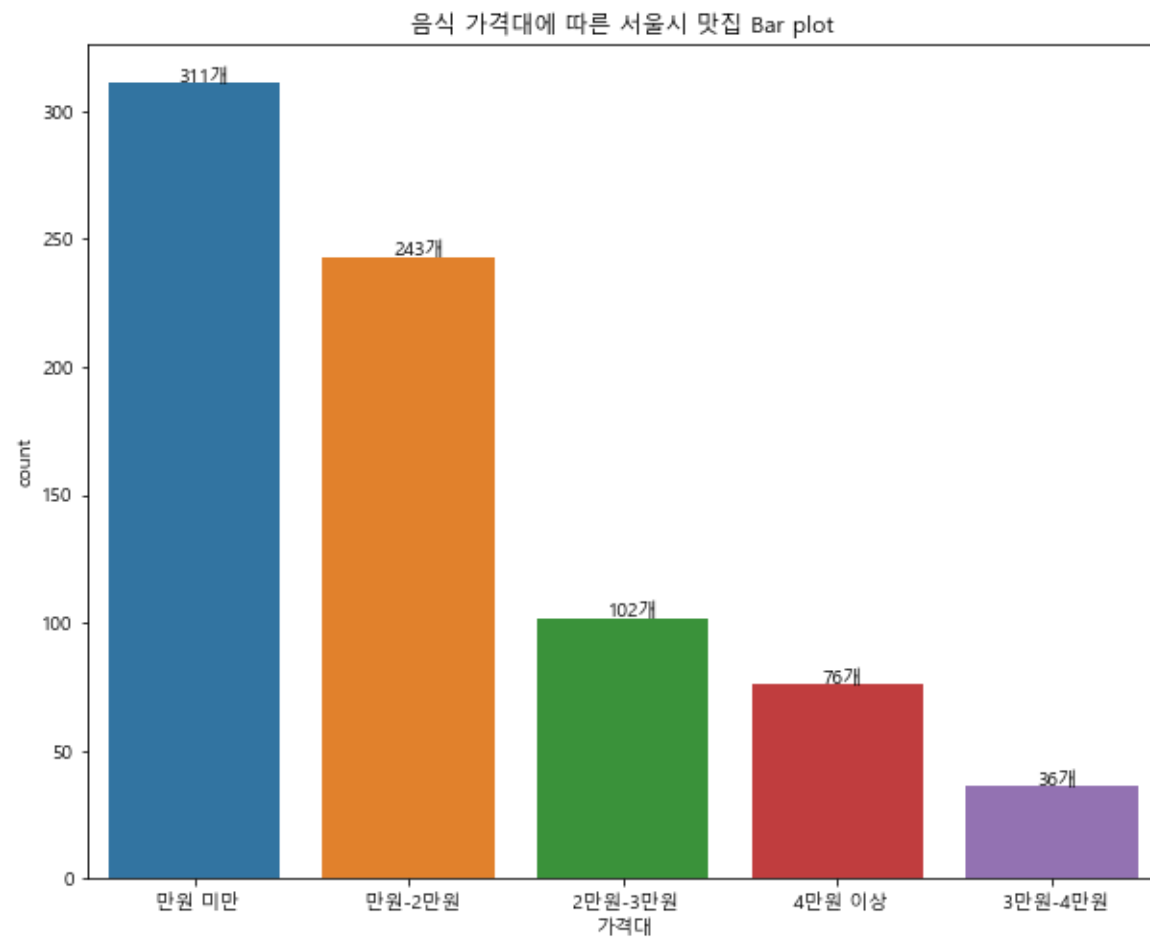
# #1 기본시각화

## <음식 종류에 따른 서울시 구별 맛집 개수>



# #1 기본시각화

## <음식 가격대별 서울시 맛집 개수>





# #1 기본시각화

## <지도에 맛집정보,위치 마커 표시>

### ■ 주소를 위경도 값으로 반환하는 함수

```
def geocoding(address):  
    try:  
        geo = geo_local.geocode(address)  
        x_y = [geo.latitude, geo.longitude]  
        return x_y  
    except:  
        return [0,0]
```

### ■ folium.Map활용해 지도에 맛집 표현

```
map = folium.Map(location=[37.5502, 126.982], zoom_start=11)  
  
for n in df.index:  
    folium.Marker(  
        [latitude[n], longitude[n]],  
        radius = 10,  
        color='#3186cc',  
        fill_color='#3186cc',  
        fill=True,  
        tooltip = ('<b>- 지역</b>: ' + df['구'][n] + '<br>' +  
                  '<b>- 상호명</b>: ' + df['식당명'][n] + '<br>' +  
                  '<b>- 평점</b>: ' + str(round(df['평점'][n], 2)) + '<br>' +  
                  '<b>- 음식분류</b>: ' + df['음식종류'][n] + '<br>' +  
                  '<b>- 가격대</b>: ' + df['가격대'][n])  
    ).add_to(map)  
map
```



# #1 기본시각화

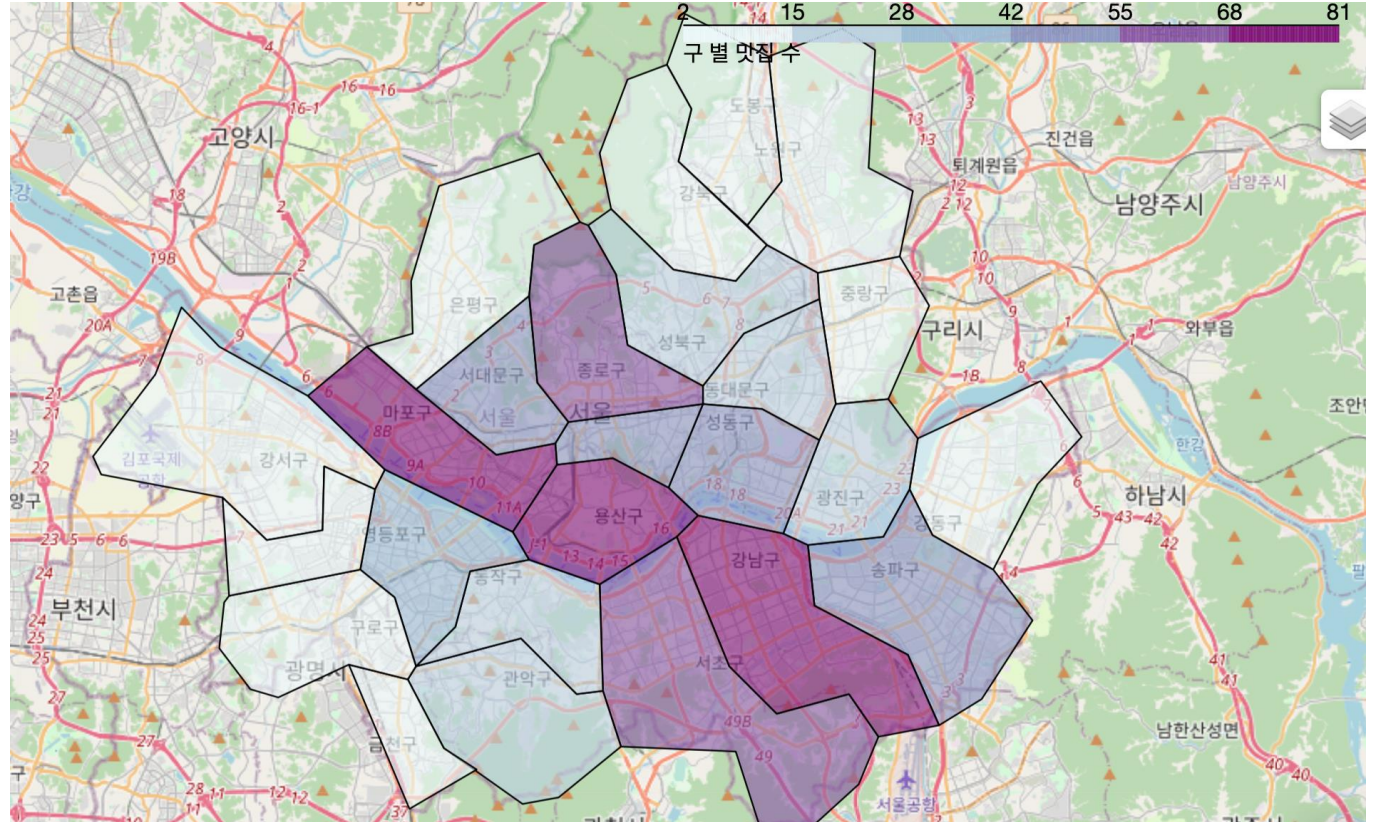
## <서울시 구별 맛집 수 단계구분도>

### ■ 단계구분도 표현 코드

```
geo_path='./data/02. skorea_municipalities_geo_simple.json'
geo_str = json.load(open(geo_path, encoding = 'utf-8'))
# geo_str

#서울을 중심으로 기본 지도 출력
#위경도[37.5502, 126.982]
map=folium.Map(location=[37.5502, 126.982],
                zoom_start=11,)
                #tiles='Stamen Terrain')

f_test=view
map.choropleth(geo_data=geo_str,
               data=f_test,
               columns=[f_test.index, '가격 점수'],
               key_on = 'feature.id',
               fill_color='BuPu',
               legend_name='구 별 가격점수')
folium.LayerControl().add_to(map)
map
```



## 03 데이터 시각화



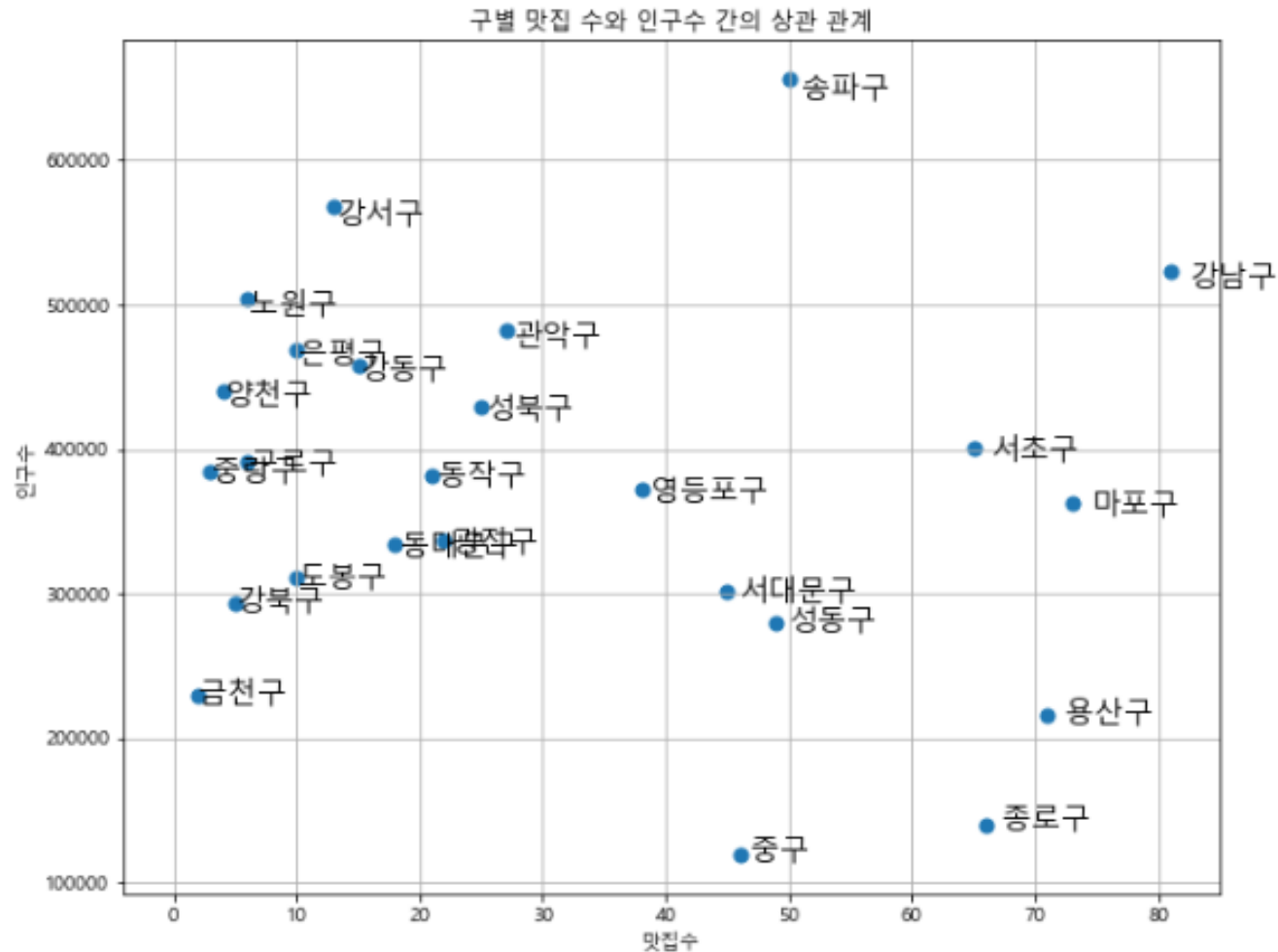
**#2**

인구

## #2

## 인구

### < 구 별 맛집 수와 인구 수 간의 상관관계 >



#### ■ 산점도 (Scatter)

- X축:맛집수
- Y축:인구수

#### ■ 결과

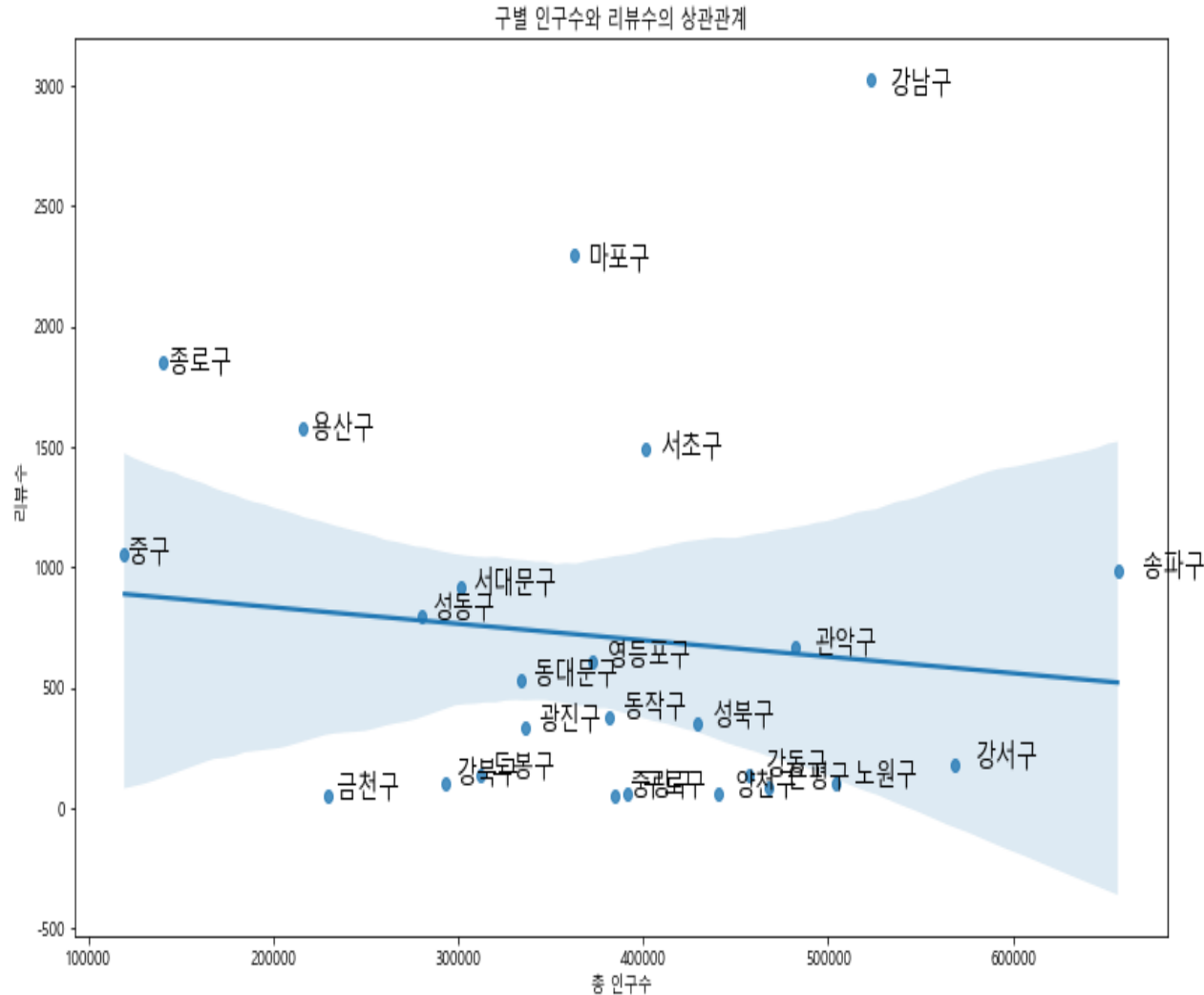
- 인구 수가 많다고 해서 꼭 맛집 수가 많은 것은 아니다



## #2

## 인구

### <구 별 인구 수와 리뷰 수의 상관관계>



#### ■ Seaborn 패키지 regplot()

- 산점도와 선형회귀분석에 의한 회귀선을 함께 나타낸다.
- X축: 총 인구수
- Y축: 리뷰수

#### ■ 결과

- 구별 인구수와 리뷰수는 서로 상관관계가 없다

## 03 데이터 시각화

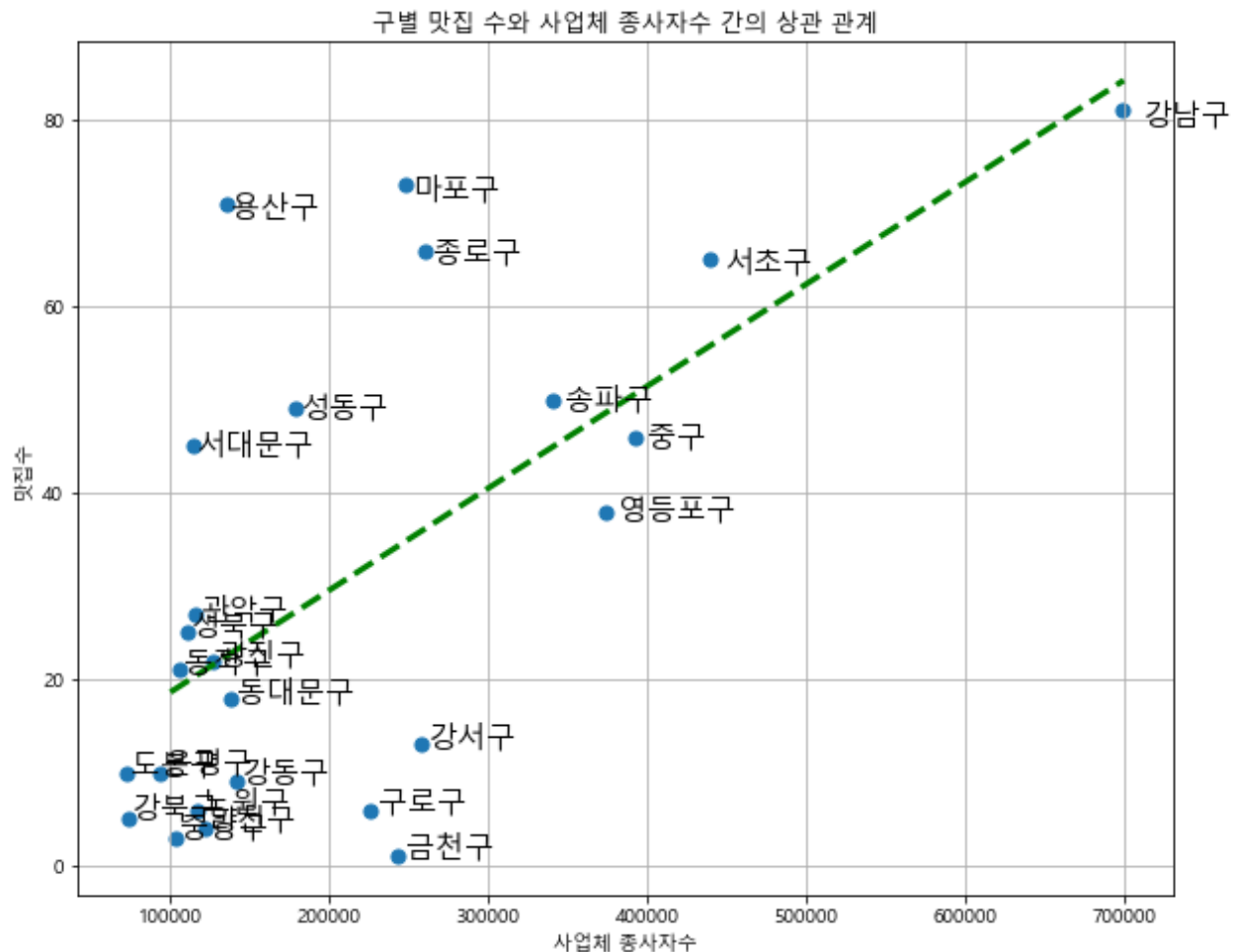
**#3**

활동인구 /  
사업체 / 인프라

# #3

## 활동인구 / 사업체 / 인프라

### < 구 별 맛집 수와 사업체 종사자 수 간의 상관관계 >



```
print(pearsonr(population['맛집수'], population['사업체종사자수']))
```

(0.6251416857877897 0.0008339021081624285)

#### ■ 결과

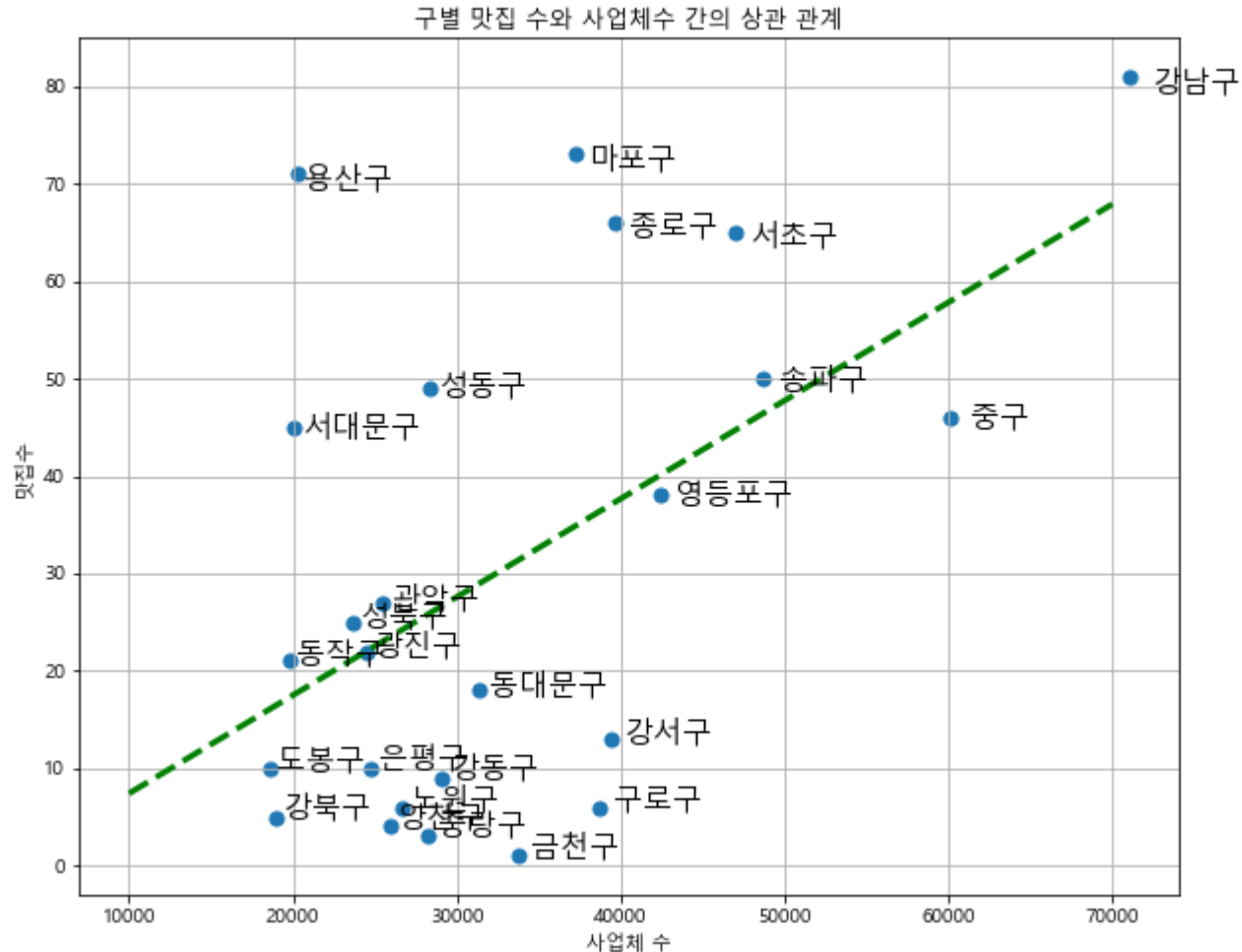
- 맛집수와 사업체종사자수는 강한 상관관계를 가지고 있다.

->사업체 종사자 수가 적을수록 맛집 수 또한 적다.

# #3

## 활동인구 / 사업체 / 인프라

### <구 별 맛집 수와 사업체 수 간의 상관관계>



```
print(pearsonr(company['맛집수'], company['사업체수']))
```

0.5190069873532723 0.007850065973888647)

#### ■ 결과

- 맛집 수와 사업체 수는  
꽤 큰 상관관계를 가지고 있다

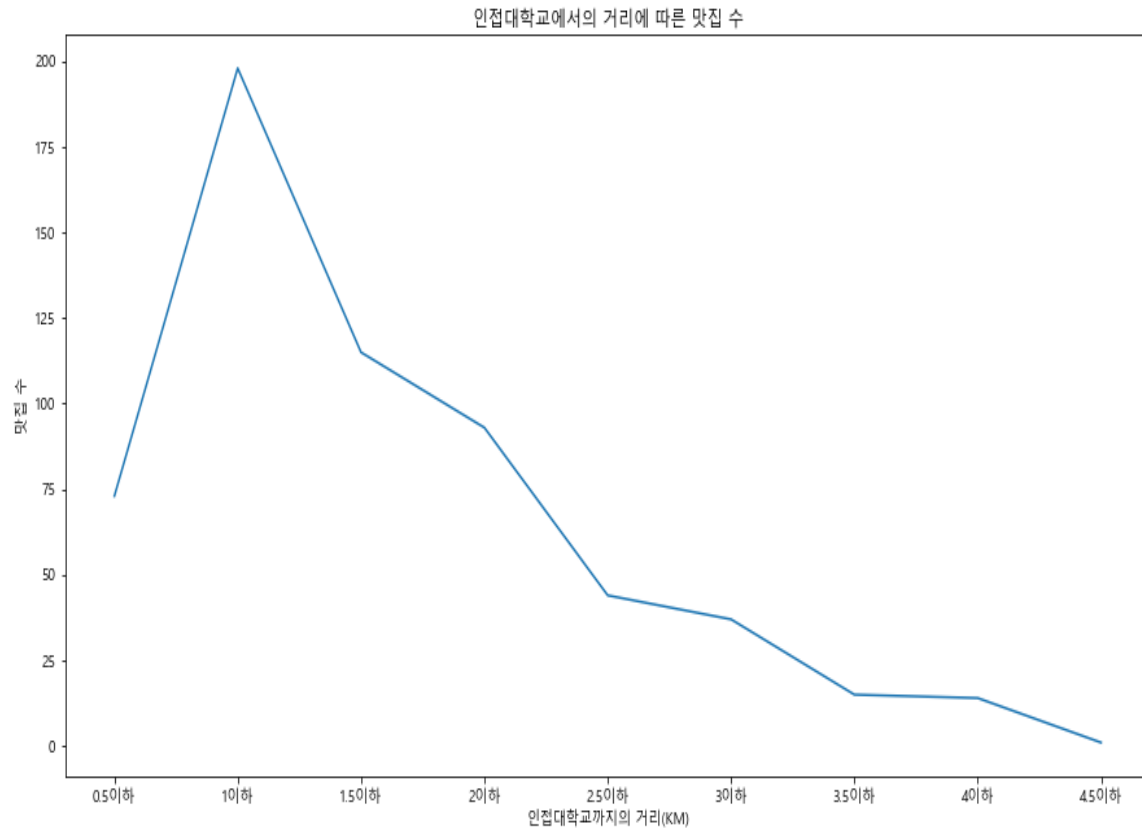
-> 보통 사업체 수가 많은 곳에  
맛집도 많이 분포한다.



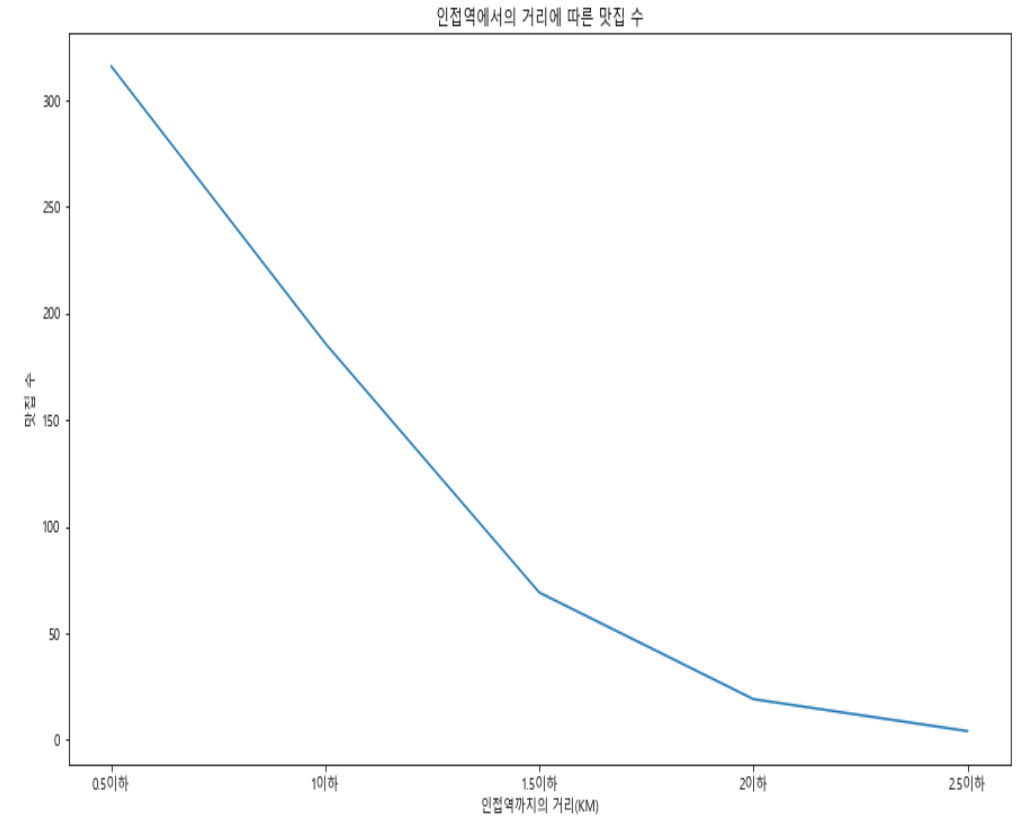
# #3

## 활동인구 / 사업체 / 인프라

### < 인접 대학교에서의 거리에 따른 맛집 수 >



### < 인접 지하철역에서의 거리에 따른 맛집 수 >



# #3 활동인구 / 사업체 / 인프라

## Haversine란?

동근 지구표면에 있는 두 좌표 사이의 거리 구하는 패키지

```
from haversine import haversine
distance = []
for i in range(0, len(res)):
    for j in range(0, len(df_trans)):
        start = (float(df_trans.loc[j, '위도']), float(df_trans.loc[j, '경도']))
        goal = (float(res.loc[i, '위도']), float(res.loc[i, '경도']))
        distance.append(haversine(start, goal))

res.loc[i, '인접역'] = df_trans.loc[distance.index(min(distance)), '역명']
res.loc[i, '인접역까지의거리'] = min(distance)
distance = []
```

맛집 데이터와 공공데이터를 이용한  
서울시 자치구별 맛집 상권분석

```
from collections import Counter
a = []
b = []
result = Counter(cat_list)

for key in result:
    a.append(key)
    b.append(result[key])

df_cam = pd.DataFrame(zip(a, b), columns = ['거리', '수'])
df_cam = df_cam.drop(0, axis = 0)
```

```
ex1 = pd.DataFrame(campus.인접대학교까지의거리.value_counts())
ex1.rename(columns = {'인접대학교까지의거리' : '인접대학수'}, inplace = True)
ex1['인접대학교까지의거리'] = ex1.index
```

```
ex1 = ex1.sort_values('인접대학교까지의거리')
ex1.reset_index(inplace = True, drop=True)
tmp = ex1[ex1['인접대학수'] == 7].index
ex1 = ex1.drop(tmp)
```

```
dis = ex1['인접대학교까지의거리'].to_list()
data = dis
bins = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5]
labels = ['0.5이하', '1이하', '1.5이하', '2이하', '2.5이하', '3이하', '3.5이하', '4이하', '4.5이하']
cats = pd.cut(data, bins, labels = labels)
cat_list = list(cats)
```

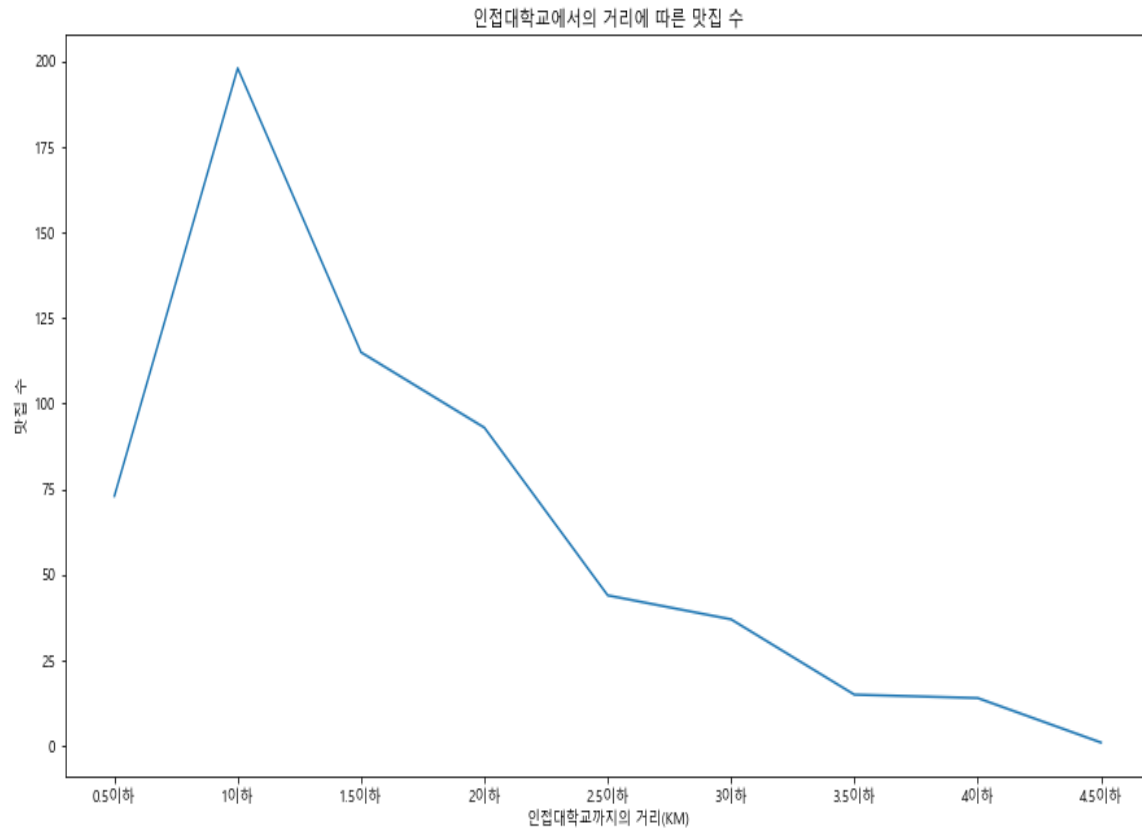
## ■지도 상 좌표간의 거리 구하는 코드

- Geocode로 모든 주소 위경도 좌표로 반환
- Haversine메소드로 맛집과의 거리를 구해 리스트에 저장
- 리스트 내의 가장 작은 값 구한 후 해당 Index(대학교 or 지하철역 이름)을 저장
- 거리에 따른 맛집의 수 구하기 :Cut() 사용
- 일정 거리별 맛집의 수 구하기:Counter() 사용

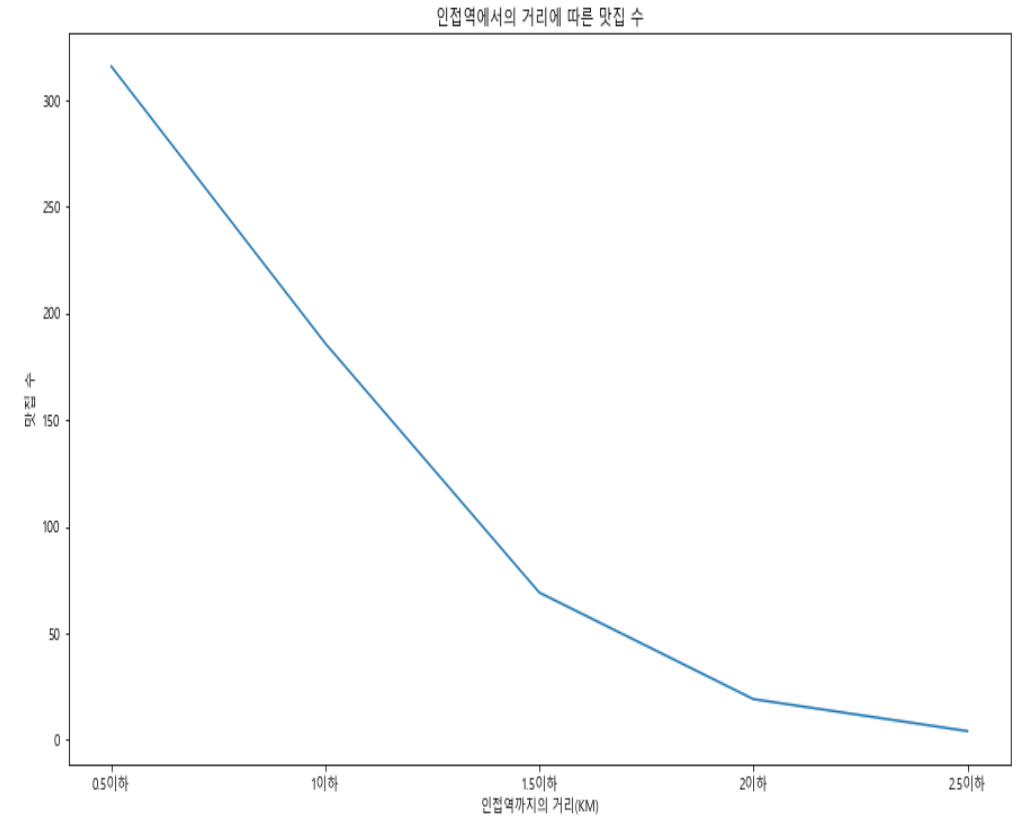
# #3

## 활동인구 / 사업체 / 인프라

### < 인접 대학교에서의 거리에 따른 맛집 수 >



### < 인접 지하철역에서의 거리에 따른 맛집 수 >



## 03 데이터 시각화

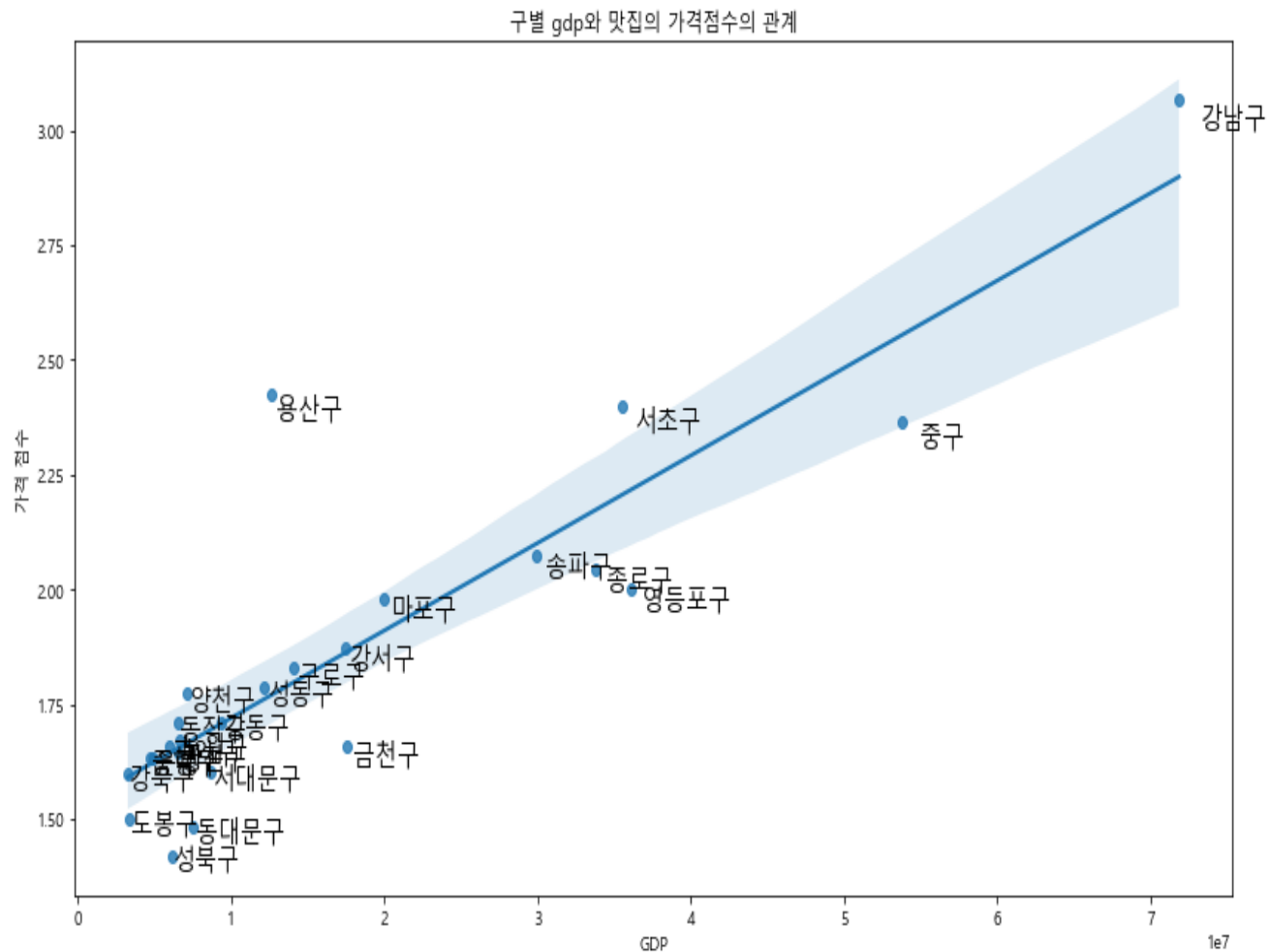
**#4**

**소득(GDP)**

# #4

## 소득(GDP)

### < 구 별 GDP ↔ 맛집 가격점수 >



```
print(pearsonr(df['GDP'], df['가격 점수']))  
(0.8789649046326994, 7.429293031312685e-09)
```

#### ■ 가격점수

- ({가격대 상대점수(각 1,2,3,4,5점)} \*  
{자치구의 가격대별 맛집 수})/자치구별 맛집 수

#### ■ 결과

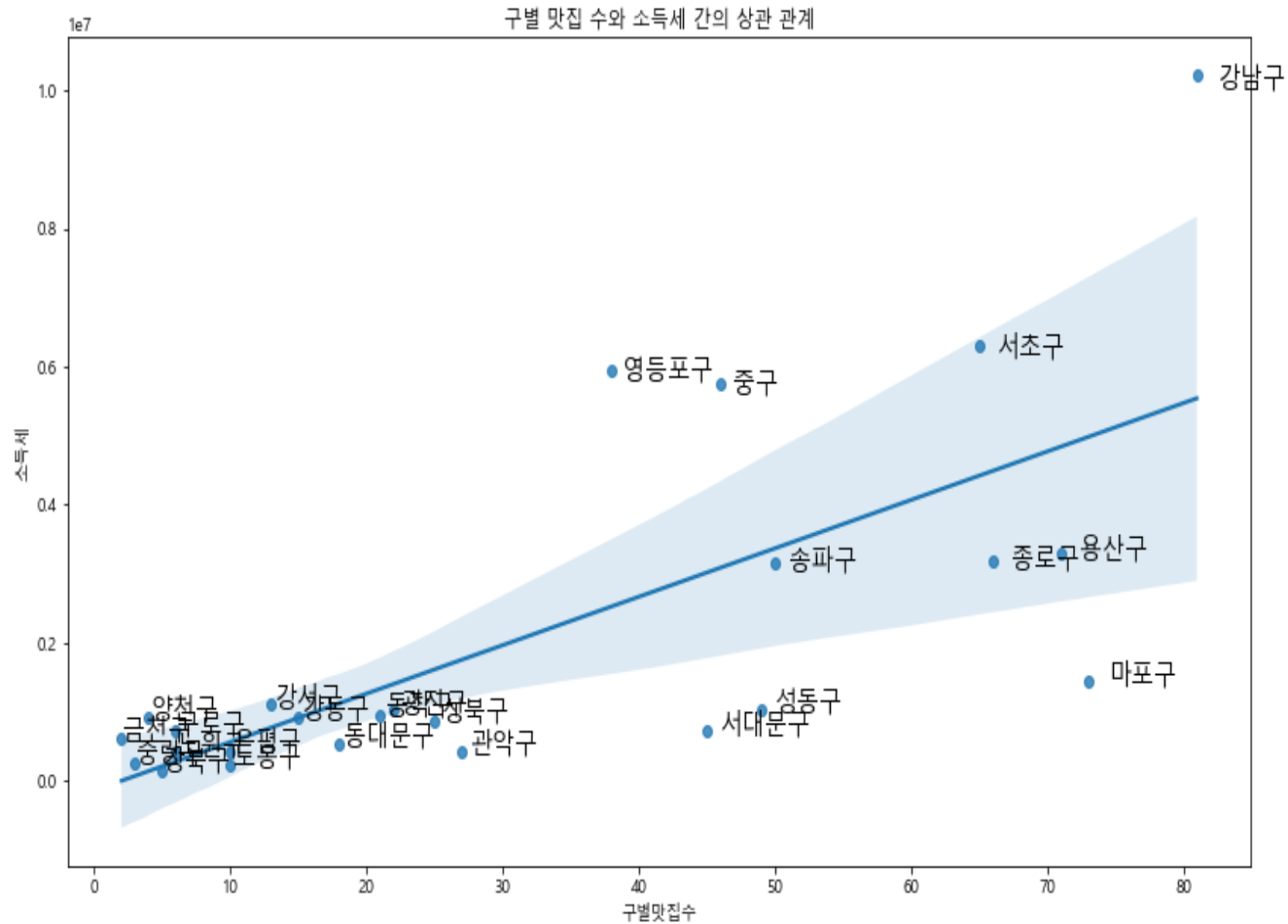
- GDP가 낮은 쪽이 가격점수도  
낮은 걸로 나타나므로  
두는 강한 상관관계를 가진다.

-> 총생산이 높을수록 맛집 평균  
가격대가 올라간다.

# #4

## 소득(GDP)

### < 구 별 맛집 수 ↔ 소득세 >



```
print(pearsonr(df_tax['구별맛집수'], df_tax['소득세']))  
(0.7081861588087626, 7.467343220764394e-05)
```

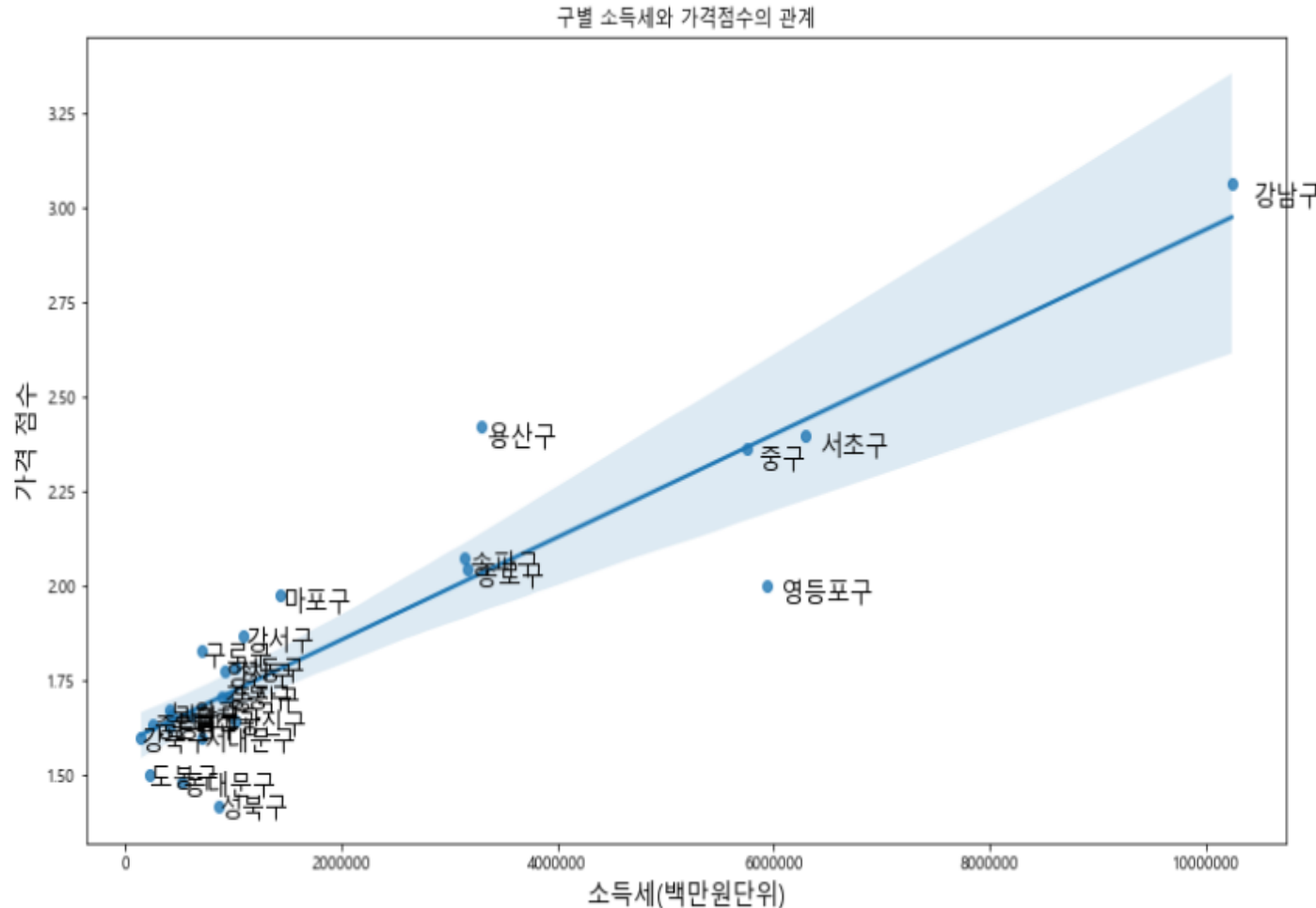
#### ■ 결과

- 소득세를 많이 내는 구일수록 맛집의 수가 많다

# #4

## 소득(GDP)

### < 구 별 소득세 ↔ 가격점수 >



```
print(pearsonr(tax['소득세'], tax['가격 점수']))  
(0.9144750319019931, 1.6386374859486384e-10)
```

#### ■ 결과

- 소득세가 가격 점수에 비례한다.
- > 소득세를 많이 내는 구일수록 가격 점수가 높다.

## 04 결과 후기 및 느낀점



## 04 결과 후기 및 느낀점

### 아쉬웠던 점

- 웹크롤링: 웹사이트(망고플레이트)의 속도 문제로 인한 데이터 수집의 한계
- 예상했던 결과와 다른 데이터 분석, 시각화 결과물
  - > 평균적인 생각과 실제의 차이 가능성 확인

### 기대효과

- 다양한 니즈에 따른 정보 확인 가능
  - 식당 창업 시 고려할 정보
  - 가격대 데이터를 통해 음식의 적정가 책정에 도움
  - 가고싶은 맛집의 위치 및 정보
- 추후 해당 데이터 분석 결과를 바탕으로 경쟁력 있는 음식 종류, 가격대 등을 이용하여 맛집 추천 시스템 개발로의 발전 가능성

감사합니다 😊