반려동물 셀프목욕방 창업을 위한 서울시 동단위 입지 분석

Drop D Bit 정사라 박태영 김원 김지우 이의준

목차

Drop D bit

반려동물 셀프목욕방 창업을 위한 서울시 동단위 입지 분석

반려가구 약 25%*

반려동물 관련업 창업, 가장 큰 고민은?

*한국방송광고공사, 2024

자영업 창업을 앞둔 소상공인의

자금마련 외 **가장 큰 고민**

입지 선정
(2위)

적절한
정보의 획득
(3위)

중소기업청(現중소벤처기업부)의 소상공인을 대상으로 한 조사에 따르면
자영업 창업 과정에서 가장 큰 애로사항은 자금마련이었으며,
입지의 선정, 업종의 선택, 적절한 정보의 획득 등이 그 뒤를 따르는 것으로 확인된다.

(중소기업청, 2013)

상업시설 입지 결정에 관한 이론적 모델

(Turhan et al. 2013)

•••

인구 구조

경제적 요인

경쟁 관계

포화 수준

점포 특성

집객시설 등

반려동물 관련 서비스업의 현재 분포상
경쟁관계를 고려할만큼의 포화수준에 이르지 않았다는 판단 하,
인구구조와 점포 특성, 집객 시설 등에 집중

공공데이터를 활용한

서울시 동 단위 입지 분석

반려 인구 수,
동물 등록 수,
인구 구조

+

관련 점포,
동반가능
시설정보

타겟

반려동물관련 창업을
준비하는 **예비 소상공인**

목적

사업안정성을 위한
빅데이터 입지분석서비스

기대효과

타당한 **자료 및 분석을 제공**하여
소상공인들 **창업의 안정적 시작**에 도움

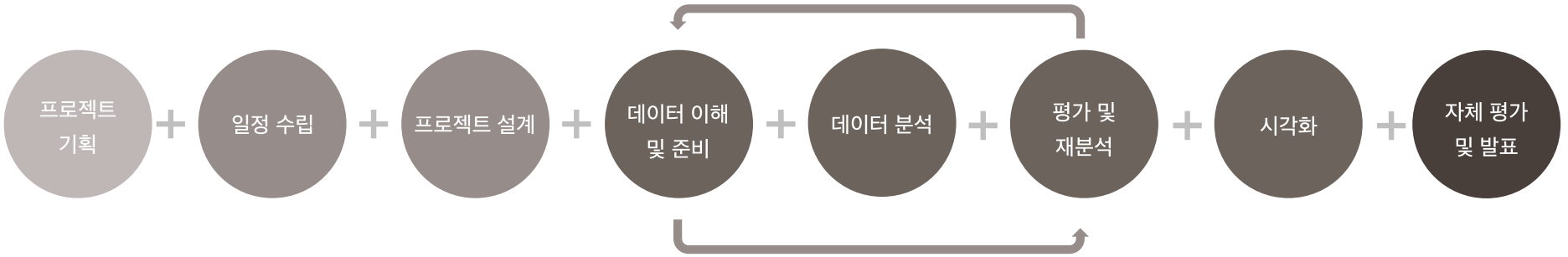
1. 프로젝트 개요

② 구성원 및 역할

정사라	**(조장) 프로젝트 기획, 일정 관리, 과업 수행 관리, 커뮤니케이션 데이터 모델링, 데이터분석(자료수집/전처리/분석/시각화), 발표자료 제작, 발표**
박태영	**(부조장) 데이터 통합, 데이터 전처리 취합, 데이터 모델링 데이터 분석(자료수집/전처리/분석/시각화), 대시보드 구현, 발표**
김원	**비즈니스 기획, 아이디에이션 데이터 분석(자료수집/전처리/분석/시각화), 발표**
김지우	**데이터 모델링 및 최적화, 데이터분석(자료수집/전처리/분석/시각화), 발표**
이의준	**데이터 전처리 정보 수집, 데이터분석(자료수집/전처리/분석/시각화), 발표**

2. 프로젝트 설계

① 수행 절차 및 방법



프로젝트 개요

비즈니스 목적과 기획을
데이터 분석을 위한 업무로 이해

프로젝트 설계

일정 계획 및
업무 분담

프로젝트 내용

기초적인 통계 분석과 시각화로 데이터 특성을 이해
모델 개발 과정 중 재탐색 및 개선 작업 진행

피드백 및 향후 계획

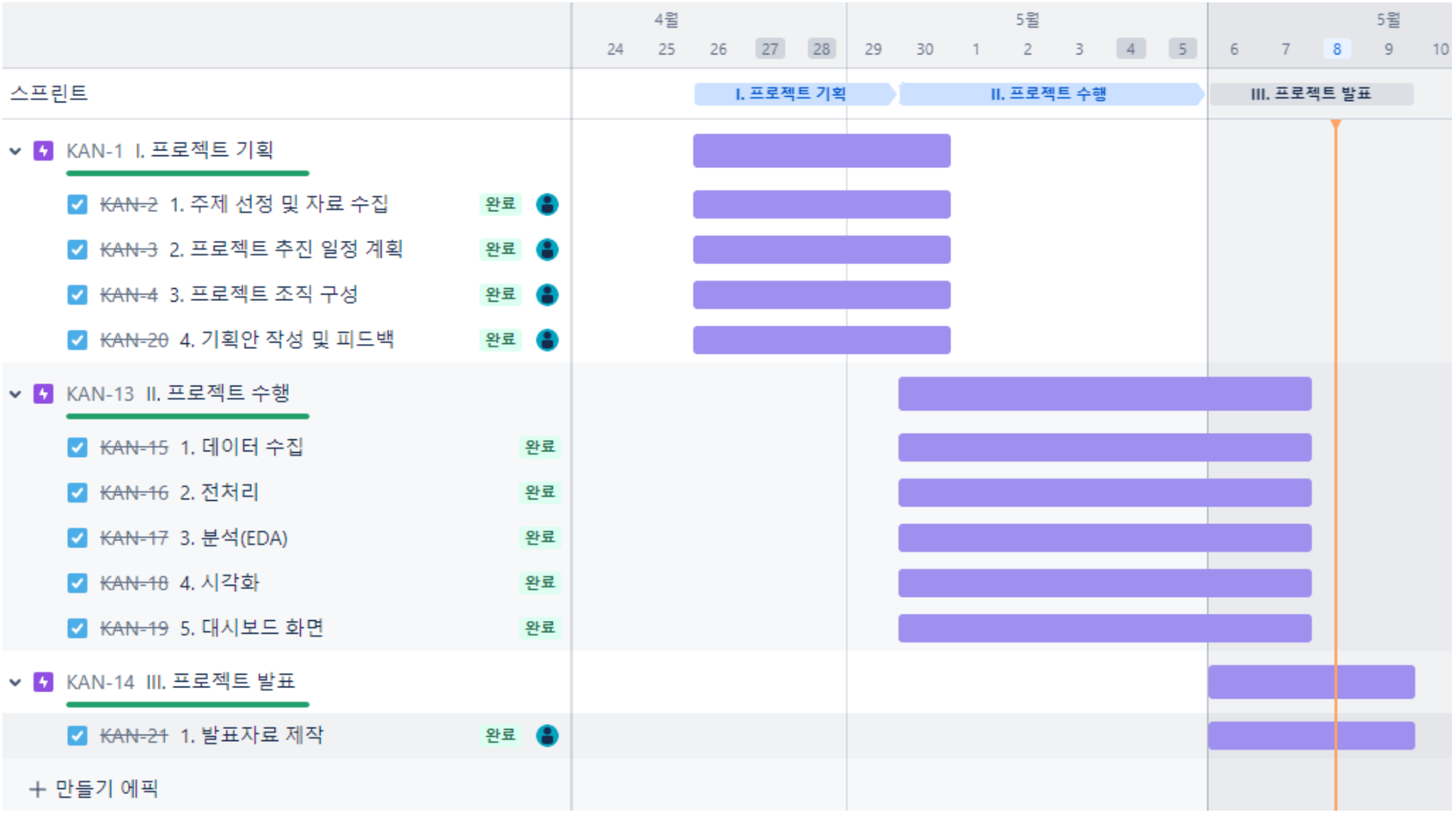
프로젝트 기간과 과업의 성격의 한계 인식
프로젝트를 통한 인사이트와 향후 보완점 정리

2. 프로젝트 설계

② 개발 일정

Drop D bit

JIRA 프로젝트



EDA 수행목표 (변수간 트렌드 분석)	- 비즈니스 목표를 위한 가설에 입각한 데이터 변수 설정 - 데이터 변수 간 연관성, 상관성, 관련성을 확인 - 데이터 시각화를 통해 목표변수와 설명변수의 관계를 확인
EDA 프로세스	- 히스토그램을 통한 분포 확인 - def calculate_statistics(df)를 통해 기초 통계값 확인 - 박스 플롯, 평균값과 중위값을 확인하여 이상치 여부 확인 - 목표변수, 설명변수 설정을 위한 변수간 상관관계 분석 시각화

Raw Data 탐색

구로구

법정동	내장형(RFID)	외장형(RFID)	인식표	등록품종수	동물소유자수	동물소유자당동물등록수	기준일
궁동	180	130	27	36	269	1.25	2022-04-30
항동	159	91	39	36	249	1.16	2022-04-30
개봉동	1189	627	185	75	1663	1.2	2022-04-30
고척동	740	310	95	60	959	1.19	2022-04-30
구로동	1457	795	314	75	2172	1.18	2022-04-30
오류동	678	311	102	55	922	1.18	2022-04-30
온수동	96	66	20	31	158	1.15	2022-04-30
천왕동	181	106	32	31	278	1.15	2022-04-30
가리봉동	187	79	21	39	194	1.48	2022-04-30
신도림동	424	202	77	55	606	1.16	2022-04-30
궁동	211	174	23	38	322	1.27	2023-04-27
항동	190	133	39	41	307	1.18	2023-04-27
개봉동	1374	778	180	75	1919	1.22	2023-04-27
고척동	859	421	94	65	1127	1.22	2023-04-27
구로동	1608	1072	311	79	2494	1.2	2023-04-27
오류동	746	398	104	57	1040	1.2	2023-04-27
온수동	117	83	20	35	190	1.16	2023-04-27
천왕동	210	154	32	33	339	1.17	2023-04-27
가리봉동	222	98	20	41	224	1.52	2023-04-27
신도림동	477	273	77	59	695	1.19	2023-04-27

도봉구

행정동	등록수
쌍문동	3,717건
방학동	3,481건
도봉동	2,081건
창 동	5,397건

서대문구

연도	누적등록수(마리)	기준일
2019	14090	2021-12-31
2020	15232	2021-12-31
2021	16944	2021-12-31

양천구

법정동	동물 등록	데이터기준일자
목동	8072	2022-08-16
신월동	7375	2022-08-16
신정동	9116	2022-08-16

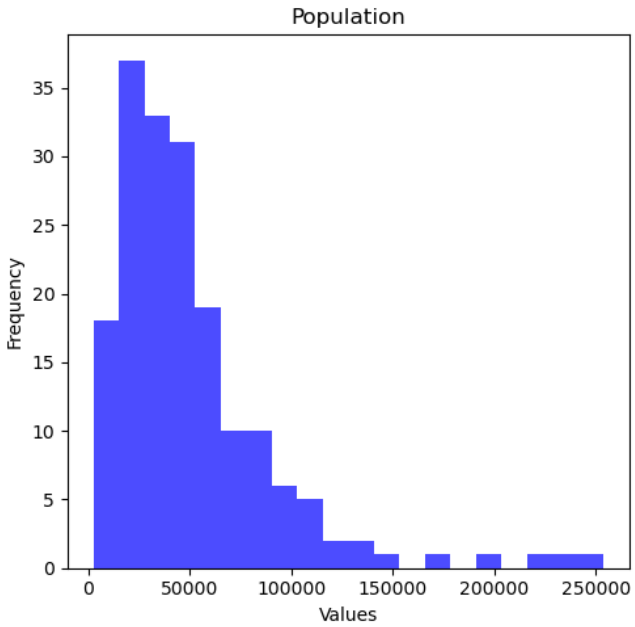
성동구

법정동	등록수	데이터기준일자
도선동	291	2023-08-23
마장동	1144	2023-08-23
사근동	352	2023-08-23
송정동	735	2023-08-23
옥수동	1739	2023-08-23
용답동	831	2023-08-23
응봉동	983	2023-08-23
행당동	2498	2023-08-23
홍익동	224	2023-08-23
금호동1가	1124	2023-08-23
금호동2가	793	2023-08-23
금호동3가	844	2023-08-23
금호동4가	929	2023-08-23
성수동1가	2285	2023-08-23
성수동2가	1791	2023-08-23
상왕십리동	256	2023-08-23
하왕십리동	1852	2023-08-23

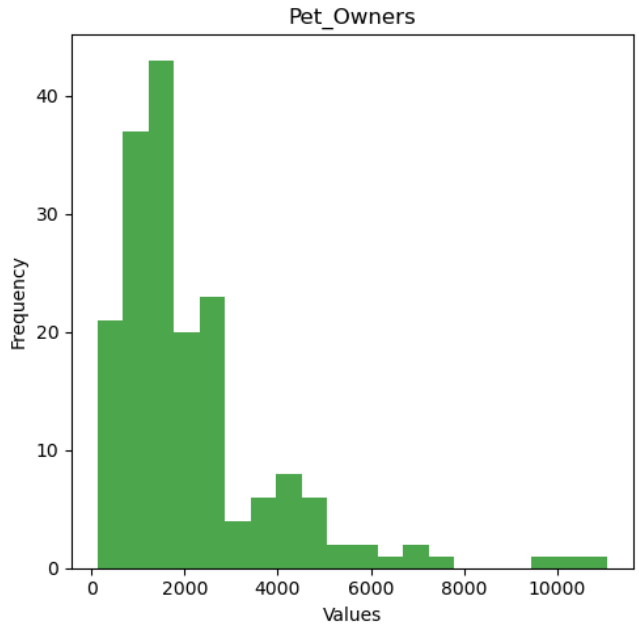
사용 데이터

결측 데이터

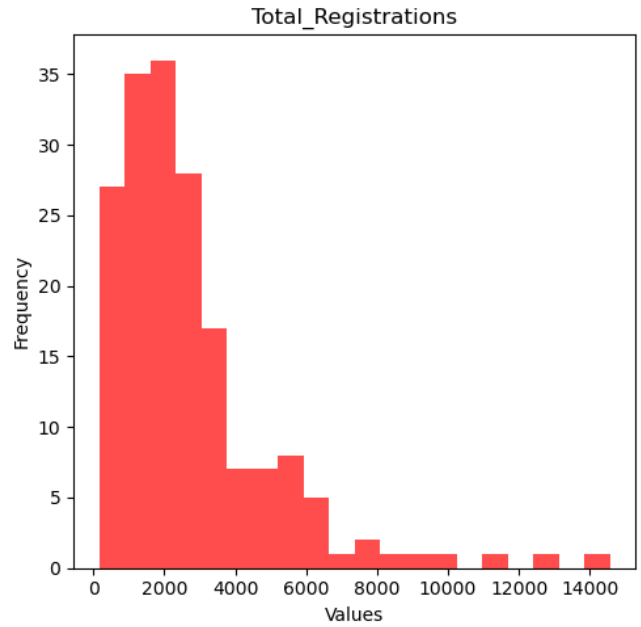
히스토그램



행정동별 인구수(Population)



반려동물 소유자수(Pet Owners)



반려동물 등록수(Total Registrations)

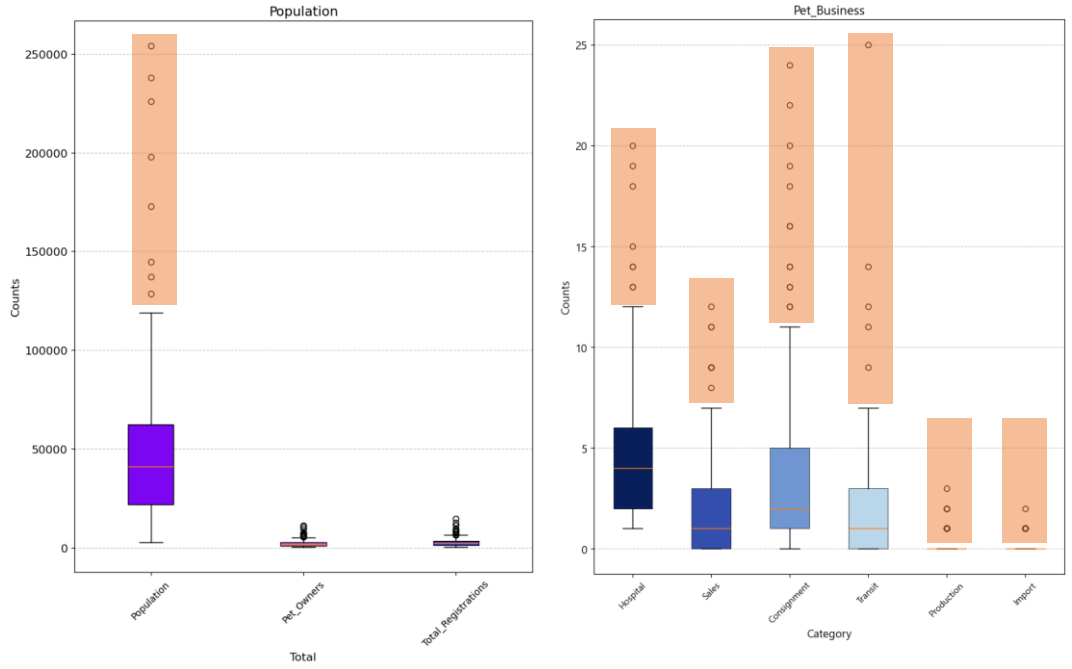
함수를 통한 대표값 확인

이상치 발견

- 대표값 중 Mean값과 Median 값의 차이를 발견, Boxplot을 통해 이상치를 확인

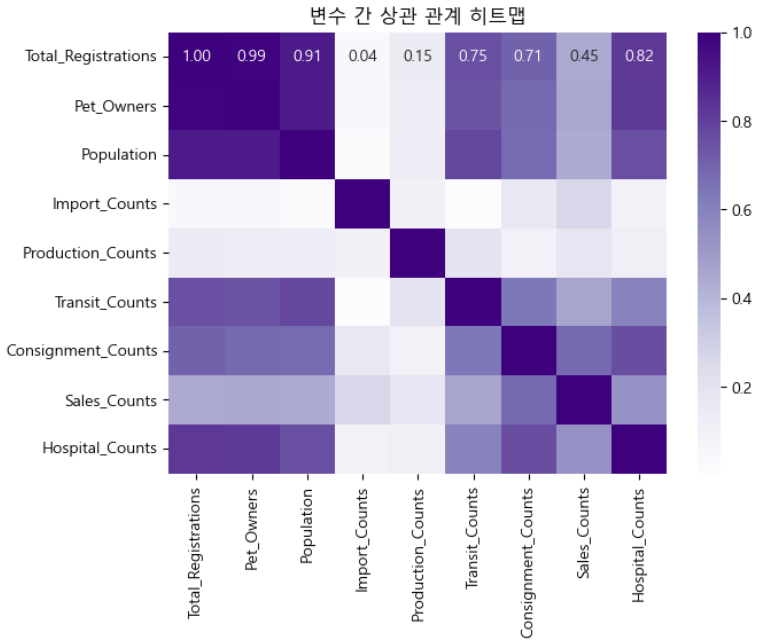
	Mean	Median
Sales_Counts	2.022346	1.0
Consignment_Counts	4.044693	2.0
Transit_Counts	2.201117	1.0
Production_Counts	0.156425	0.0
Import_Counts	0.089385	0.0
Population	50067.798883	41180.0
Pet_Owners	2159.206704	1672.0
Total_Registrations	2731.296089	2152.0

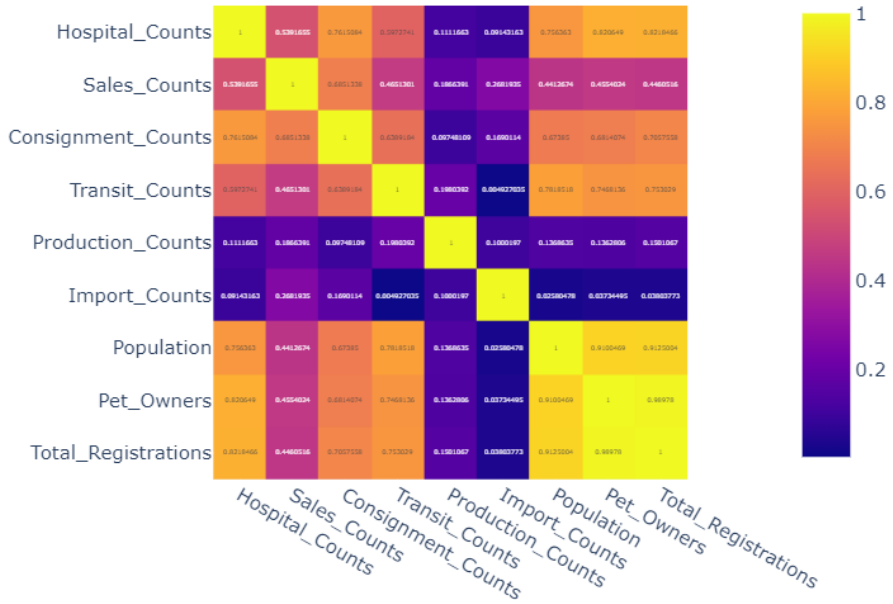
Basic Statistics



Box Plot

목표변수, 설명변수 설정을 위한 변수간 상관관계 분석 시각화





목표변수

반려동물 등록수(Total Registrations)

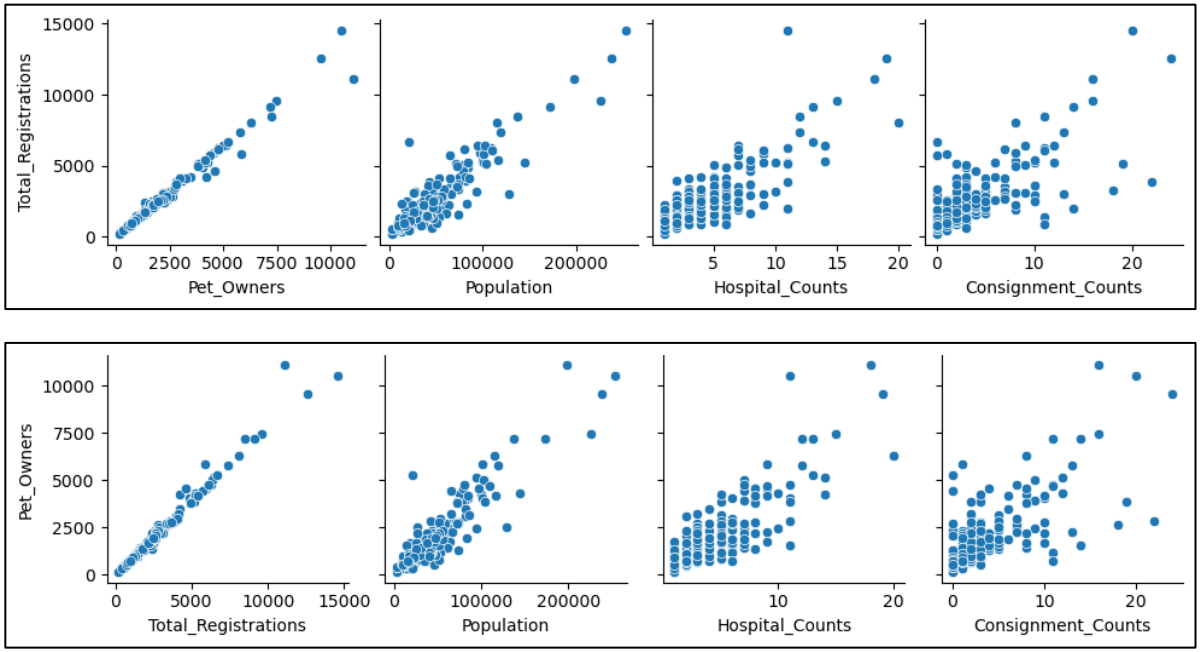
반려동물 소유자수(Pet Owners)

설명변수

인구수 (Population)

Hospital_Counts

Consignment_Counts



Drop D bit

0. OpenAPI

```python
import requests
import pandas as pd
import geopandas as gpd
from bs4 import BeautifulSoup

# 정보 호출
def call_api(start_idx, end_idx):
    api_key = "**********%2FyeabMAB6tEf%2BQpvpwd55bvlexDRqFtoIFH58NbkEY8PTpBHsy%**********"
    base_url = "http://211.237.50.150:7080/openapi/"
    api_url = "Grid_20210806000000000612_1"
    url = (
        f"{base_url}{api_key}/xml/{api_url}/{start_idx}/{end_idx}?"
    )
    response = requests.get(url)
    xml_data = response.text
    soup = BeautifulSoup(xml_data, 'xml') # XMl, Json 형태
    return soup
```

1. 결측 데이터 확인

	A	B	C	D	E
1	읍면동	시군구	RFID내장형	RFID외장형	인식표
2	길동	97	836	1062	121
3	강일동	18	406	387	21
4	고덕동	39	626	622	51
5	둔촌동	37	611	865	48
6	명일동	30	669	794	71
7	상일동	24	440	466	30
8	성내동	141	1408	1401	162
9	암사동	71	1217	1717	149
10	천호동	135	1923	1992	186

```python
# 데이터 값 더해서 새로운 열 생성
df_gd = pd.DataFrame(gangdong)
df_gd['동물 등록수'] = np.sum(df_gd[['RFID내장형','RFID외장형','인식표']], axis=1)

# 동물소유자수를 구하기 위해 동물 등록수를 '동물소유자당' 값으로 나눈 후 1의 자리까지 반올림, 정수로 만듬
df_gd['동물 소유자수'] = np.round(df_gd['동물 등록수'] / df_gd['동물소유자당'], decimals=0).astype(int)
#print(df_gd)
```

2. 처리 할 수 없는 데이터

	A	B	C	D	E
1	시군구	법정동	인식표	반려동물총계	동물소유자수
2	서울특별시	본동	67	585	462
3	서울특별시	대방동	224	1979	1579
4	서울특별시	동작동	23	342	283
5	서울특별시	사당동	445	5126	3871
6	서울특별시	상도동	593	6119	4738
7	서울특별시	흑석동	96	1702	1340
8	서울특별시	상도1동	39	495	382
9	서울특별시	노량진동	144	1509	1158
10	서울특별시	신대방동	229	2438	1888

동작구

	A	B	C
1	연도	누적등록수	기준일
2	2019	14090	2021-12-31
3	2020	15232	2021-12-31
4	2021	16944	2021-12-31

서대문구

3. Columns 통일

	A	B	C	D	E
1	시군구	법정동	인식표	반려동물총계	동물소유자수
2	서울특별시	본동	67	585	462
3	서울특별시	대방동	224	1979	1579
4	서울특별시	동작동	23	342	283
5	서울특별시	사당동	445	5126	3871
6	서울특별시	상도동	593	6119	4738
7	서울특별시	흑석동	96	1702	1340
8	서울특별시	상도1동	39	495	382
9	서울특별시	노량진동	144	1509	1158
10	서울특별시	신대방동	229	2438	1888

```python
def Change_Col_Names(*Variable_Name):
    for k in Variable_Name:
        k.rename(columns={
                        'Count': 'Counts',
                        'count': 'Counts',
                        '시도 명칭':'City',
                                        '시군구 명칭':'Gu',
                                        '법정읍면동명칭' : 'Dong',
                                        '위도' : 'Y',
                                        '경도' : 'X'}, inplace=True)
        k.sort_values(by=['Gu','Dong'], inplace=True)
        k.reset_index(drop = True, inplace=True)

Change_Col_Names(Pet_병원, Pet_판매, Pet_위탁, Pet_운송, Pet_생산, Pet_수입)
```

✓ 0.0s

3. 내용

② 데이터 준비

4. 데이터 수집 및 행정동으로 통일

행정동

법정동

```
"""
행정동 : ['남영동', '보광동', '서빙고동', '용문동', '용산동', '원효로동', '이촌동', '이태원동', '청파동','한강로동', '한남동', '효창동', '후암동']

법정동 : ['남영동''보광동' '서빙고동'  '용문동'   '용산동'    '원효로동'   '이촌동'  '이태원동'  '청파동'  '한강동'  '한남동'    '효창동'    '후암동']
         '갈월동'          '동빙고동' '도원동'             '문배동'                          '서계동'
         '동자동'          '주성동'                      '산천동'
                                                        '청암동'
                                                        '신창동'
                                                        '신계동'
"""
test_pet.loc[test_pet['Dong'].str.contains('갈월|동자'), 'Dong'] = '남영동'
test_pet = test_pet.groupby('Dong').agg({'Pet_Owners': 'sum', 'Total_Registrations': 'sum'}).reset_index()

test_pet.loc[test_pet['Dong'].str.contains('동빙고동|주성동'), 'Dong'] = '서빙고동'
test_pet = test_pet.groupby('Dong').agg({'Pet_Owners': 'sum', 'Total_Registrations': 'sum'}).reset_index()

test_pet.loc[test_pet['Dong'].str.contains('도원동'), 'Dong'] = '용문동'
test_pet = test_pet.groupby('Dong').agg({'Pet_Owners': 'sum', 'Total_Registrations': 'sum'}).reset_index()

test_pet.loc[test_pet['Dong'].str.contains('문배동|산천동|청암동|신창동|신계동'), 'Dong'] = '원효로동'
test_pet = test_pet.groupby('Dong').agg({'Pet_Owners': 'sum', 'Total_Registrations': 'sum'}).reset_index()

test_pet.loc[test_pet['Dong'].str.contains('서계동'), 'Dong'] = '청파동'
test_pet = test_pet.groupby('Dong').agg({'Pet_Owners': 'sum', 'Total_Registrations': 'sum'}).reset_index()
```

5. Counts

```
# '구'와 '동'을 기준으로 그룹화하여 카운트
location_counts1 = Hospital_data1.groupby(['Current_License_info','구', '동','Category']).size().reset_index(name='count')

# 'Location'과 'Category', 그리고 'count' 열만 선택하여 결과를 출력
result1 = location_counts1[['Current_License_info','구','동', 'Category','count']]
result1
```

	Current_License_info	구	동	Category	count
0	1975	은평구	응암동	동물병원	1
1	1983	구로구	오류동	동물병원	1
2	1988	관악구	봉천동	동물병원	1
3	1988	마포구	노고산동	동물병원	1
4	1988	영등포구	신길동	동물병원	1
...	...	...	...	...	...
755	2024	송파구	가락동	동물병원	2
756	2024	영등포구	신길동	동물병원	1
757	2024	용산구	한강로2가	동물병원	1
758	2024	중랑구	망우동	동물병원	1
759	2024	중랑구	상봉동	동물병원	1

760 rows × 5 columns

6. Feature

```python
# 각 특성의 필요한 Feature 구성
Pet_병원_col = Pet_병원.rename(columns={'Category':'Hospital','Counts':'Hospital_Counts'})
Pet_판매_col = Pet_판매.rename(columns={'Category':'Sales','Counts':'Sales_Counts'})
Pet_위탁_col  = Pet_위탁.rename(columns={'Category':'Consignment','Counts':'Consignment_Counts'})
Pet_운송_col  = Pet_운송.rename(columns={'Category':'Transit','Counts':'Transit_Counts'})
Pet_생산_col  = Pet_생산.rename(columns={'Category':'Production','Counts':'Production_Counts'})
Pet_수입_col  = Pet_수입.rename(columns={'Category':'Import','Counts':'Import_Counts'})
```

✓ 0.0s

7. 전처리 완료 후 통합

```python
# left 조인
merge_result = Seoul_Pet_Hospital_Attach.copy()

# 목록 만들기
Combine_Pet_business = [Seoul_Pet_Sales_Attach, Seoul_Pet_Consignment_Attach,
                        Seoul_Pet_Transit_Attach, Seoul_Pet_Production_Attach, Seoul_Pet_Import_Attach]

# 초기화
result = merge_result.copy()

# 반복문 안에서 left 조인 수행
for df in Combine_Pet_business:
    result = pd.merge(result, df, on=['Gu', 'Dong'], how='left')
    result.fillna(0, inplace=True)

# 결과 확인
Cols_int = ['Hospital_Counts', 'Sales_Counts', 'Consignment_Counts', 'Transit_Counts', 'Production_Counts', 'Import_Counts']
result[Cols_int] = result[Cols_int].astype('int64')

Gu_Dong_Pet_Business = result.copy()
Gu_Dong_Pet_Business.info()

# 각 파트별 통합본 병합.
# Gu와 Dong을 기준으로 두 데이터프레임 병합
Seoul_Gu_Dong_Pet_Business_Population = pd.merge(Gu_Dong_Pet_Business, Seoul_Population_merged, on=['Gu', 'Dong'], how='left')
```

각 업종별로 전처리가 완료된 데이터 병합하였습니다.

결측치 데이터

일반적인 데이터 형태

	시군구	법정동	내장형	외장형	인식표	반려동물총계	등록품종수	동물소유자수
0	서울특별시 동작구	본동	278	240	67	585	55	462
1	서울특별시 동작구	대방동	825	930	224	1979	73	1579
2	서울특별시 동작구	동작동	171	148	23	342	42	283

→ 법정동, 반려동물 총계(반려동물 총 등록 수), 동물 소유자 수 컬럼 존재

결측치 데이터

- 결측치 데이터 형태

	법정동	RFID 내장형	RFID 외장형	RFID 인식표	합계	데이터기준일자
0	중동	335.0	253.0	46.0	634.0	45337.0
1	공덕동	871.0	555.0	79.0	1505.0	45337.0
2	구수동	33.0	25.0	4.0	62.0	45337.0

→ 반려동물 소유자 수 추정 필요
→ 마포구, 도봉구, 양천구, 성동구 데이터에서 결측치 발견

결측치 데이터

- 결측치 데이터 형태

	연도	누적등록수(마리)	기준일
0	2019	14090	2021-12-31
1	2020	15232	2021-12-31
2	2021	16944	2021-12-31

→ 반려동물 소유자 수, 반려동물 총 등록 수 두개의 데이터 추정 필요
→ 서대문구 데이터에서 결측치 발견

모델링1

1) 변수 설정

- 목표: 반려동물 소유자 수 예측 필요
- 독립변수 : Population, Total_Registration
- 종속변수 : Pet_Owners

2) 모델 리스트

- Linear Regression
- Robust Regression
- Random Forest Regressor
- XGB Regressor
- Gradient Boosting

모델 평가 지표

```python
import numpy as np
import pandas as pd

# 평가 지표 계산 함수
def score(test_y, predict):
    from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

    mse = mean_squared_error(test_y, predict)
    mae = mean_absolute_error(test_y, predict)
    r_squared = r2_score(test_y, predict)
    rmse = mean_squared_error(test_y, predict, squared=False)

    print("Mean Squared Error (MSE):", mse)
    print("Mean Absolute Error (MAE):", mae)
    print("R-squared:", r_squared)
    print("RMSE:", rmse)
    from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

→ 5개의 모델 모두 r-square이 0.98이상으로 모델이 종속변수의 변동성을 완벽하게 설명함

Model 1

```python
# 그래디언트 부스팅 회귀 모델
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# 독립 변수와 종속 변수 설정
x = model1_train_x
y = model1_train_y

gb_reg = GradientBoostingRegressor(n_estimators=1000, random_state=800, max_depth=5)

# 모델 학습
model_gb = gb_reg.fit(x, y)

# 테스트 데이터에 대한 예측
y_pred_gb = gb_reg.predict(model1_test_x)

# 모델 평가 (RMSE 계산)
#rmse_xgb = mean_squared_error(y_test, y_pred_xgb, squared=False)

# 모델 평가
predict_gb = model_gb.predict(model1_test_x)

score(model1_test_y, predict_gb)
```

Gradient Boosting 성능 평가

```
Mean Squared Error (MSE): 10097.390618919175
Mean Absolute Error (MAE): 66.7724676596378
R-squared: 0.9919549642622238
RMSE: 100.4857732165065
```

➔ Gradient Boosting 모델이 r-square 0.99로 종속 변수의 변동성을 제일 잘 설명함

Model 1

```
raw_sd = pd.read_csv('seongdong_model1.csv')
#print(raw_sd)

sd = raw_sd[['Population', 'Total_Registrations']]

# Gradient Boosting 모델로 예측
predict_sd = model_gb.predict(sd)

#print(predict_sd)

# Pet_Owners 컬럼 추가 (반올림으로 정수 만듦)

raw_sd['Pet_Owners'] = predict_sd.round().astype(int)

print(raw_sd)

# 데이터 저장

#raw_sd.to_csv('seongdong_m1_result.csv', index=False)
```

소유자 수 추청

성동구, 서대문구, 마포구, 도봉구
➔Gradient Boosting 모델

	Dong	Population	Total_Registrations	Pet_Owners
0	금호동	51056	3690	2770
1	마장동	22170	1144	881

Model 1

```
raw_yc = pd.read_csv('yangcheon_model1.csv')
#print(raw_yc)

yc = raw_yc[['Population', 'Total_Registrations']]

# Robust 모델로 예측
predict_yc = result_robust.predict(sm.add_constant(yc))

# Pet_Owners 컬럼 추가 (반올림으로 정수 만듦)

raw_yc['Pet_Owners'] = predict_yc.round().astype(int)

print(raw_yc)

# 데이터 저장

#raw_yc.to_csv('yangcheon_m1_result.csv', index=False)
```

소유자 수 추청

양천구
➔ Robust 모델
➔ Gradient Boosting의 재현율 ↓

모델링2

1) 변수 설정

- 목표: 동 별 반려동물 총 등록 수 예측 필요
- 독립변수 : Population, Hospital_Count, Consignment_Count
- 종속변수 : Total_Registration

2) 모델 리스트

- Linear Regression
- Robust Regression
- Random Forest Regressor
- Ridge
- ElasticNet
- Lasso

Model 2

```python
from sklearn.linear_model import ElasticNet

# 엘라스틱넷 회귀 모델 생성
alpha = 8.0 # L1 및 L2 규제 강도 (하이퍼파라미터)
l1_ratio = 0.1  # L1 규제의 비율 (하이퍼파라미터)
model_el = ElasticNet(alpha=alpha, l1_ratio=l1_ratio)

x = model2_train_x
y = model2_train_y

# 모델 학습
model_el.fit(x,y)

# 테스트 데이터로 예측
predict_el = model_el.predict(model2_test_x)

# 미세조정
predict_el = np.clip(predict_el, 0, 9894.280840991367)

# 평가지표 계산
score(model2_test_y, predict_el)
```

ElasticNet 모델 성능 평가

```
Mean Squared Error (MSE): 434424.47177639836
Mean Absolute Error (MAE): 531.6760166457444
R-squared: 0.9130972331865643
RMSE: 659.1088466834582
```

➔ ElasticNet 모델이 r-square 0.91로 종속 변수의 변동성을 제일 잘 설명함

Drop D bit

Model 2

```
raw_sdm = pd.read_csv('seodaemun_model2.csv')
#print(raw_sdm)

sdm = raw_sdm[['Population',  'Hospital_Counts',  'Consignment_Counts']]

# ElasticNet 모델로 예측
predict_sdm = model_el.predict(sdm)

# 미세조정
predict_sdm = np.clip(predict_sdm, 0, 9894.280840991367)

#print(predict_sdm)

# Total_Registration 컬럼 추가 (반올림으로 정수 만듬)

raw_sdm['Total_Registrations'] = predict_sdm.round().astype(int)

print(raw_sdm)

# 데이터 저장

#raw_sdm.to_csv('seodaemun_m2_result.csv', index=False)
```

동 별 반려동물 총 등록 수 추정

서대문구
➔ ElasticNet 모델

	Dong	Population	Hospital_Counts	Consignment_Counts	Total_Registrations
0	남가좌동	47253	4.0	1.0	2471
1	북가좌동	48560	5.0	1.0	2609

Model 2 성능 개선 - PCA 분석

1. 독립변수 정규화 작업

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 독립변수와 종속변수 나누기
x = data[['Hospital_Counts', 'Transit_Counts', 'Population']].values
y = data['Total_Registrations'].values
y = pd.DataFrame(y)

# 독립변수 정규화 작업
x = StandardScaler().fit_transform(x)

# 컬럼 명 지정
features = ['Hospital_Counts', 'Transit_Counts', 'Population']
pd.DataFrame(x, columns=features).head()
```

Model 2 성능 개선 - PCA 분석

2. 주성분 3개 선택

```
# 주성분을 3개로 선택
pca = PCA(n_components=3)
printcipalComponents = pca.fit_transform(x)

# 주성분으로 이루어진 데이터 프레임 구성
principalDf = pd.DataFrame(data=printcipalComponents,
                           columns = ['principal component1','principal component2',
                                      'principal component3'])
principalDf
```

Model 2 성능 개선 - PCA 분석

3. 회귀 모델 시도(Ridge)

```
from sklearn.linear_model import Ridge

alpha = 0.9  # 규제 강도 (하이퍼파라미터)
model_ridge = Ridge(alpha=alpha)
x = model2_train_x_v4
y = model2_train_y_v4
# 모델 학습
model_ridge.fit(x,y)

# 테스트 데이터로 예측
predict_ridge = model_ridge.predict(model2_test_x_v4)

# 평가지표 계산
score(model2_test_y_v4, predict_ridge)
```

```
Mean Squared Error (MSE): 311842.1324716351
Mean Absolute Error (MAE): 435.5419943120359
R-squared: 0.9532510804547811
RMSE: 558.4282697640182
```

➔ Model 2 6개의 모델 리스트 중 Ridge 모델이 R-squared 0.95로 성능 개선

PCA 분석 피드백

➔ 독립변수 3개, 주성분 3개
➔ 모델의 복잡성과 과적합 문제

해결방안
- 주성분의 개수↓ - 복잡성 문제 해결 시도

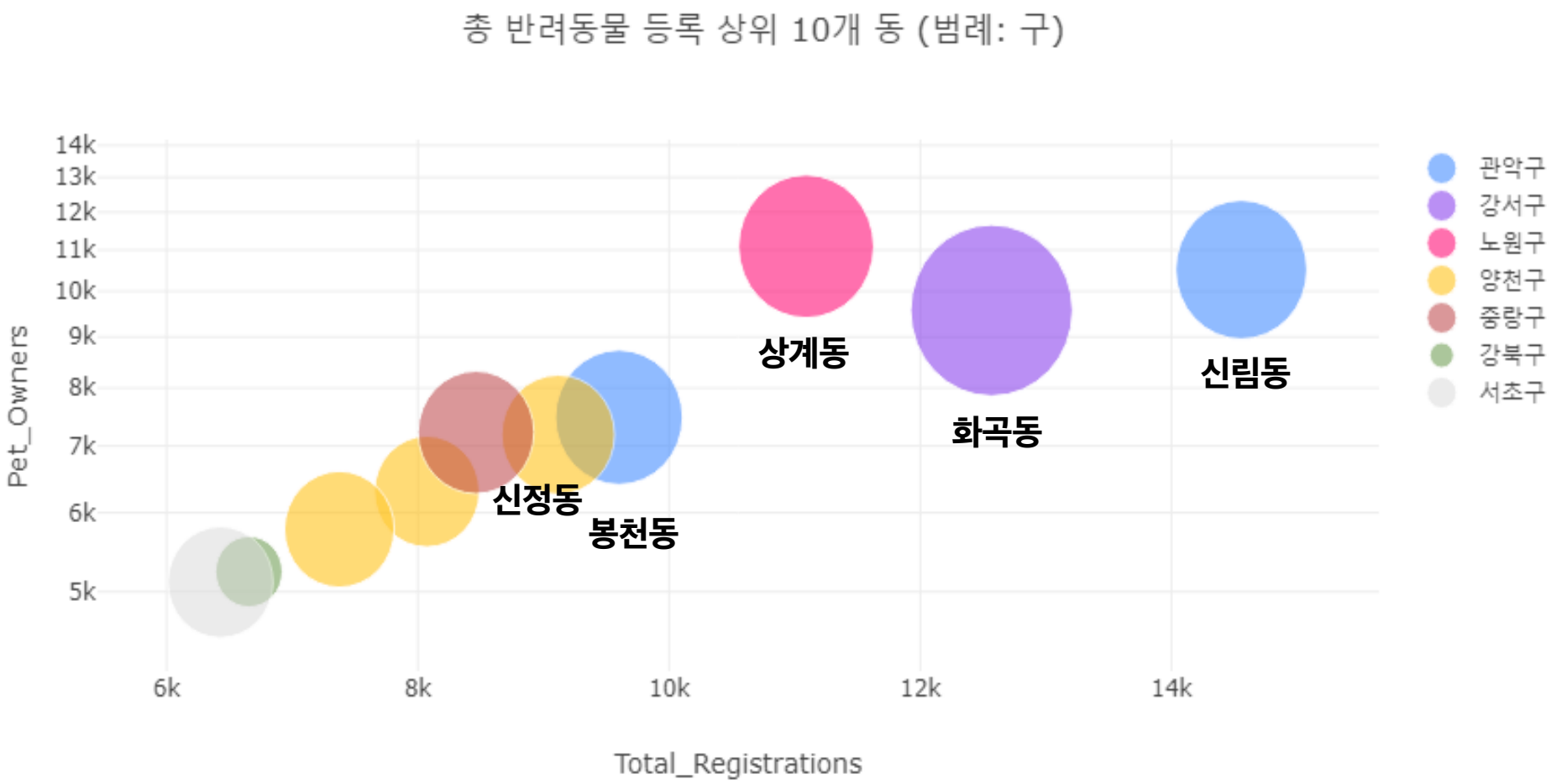
```
# 주성분을 2개로 선택
pca = PCA(n_components=2)
printcipalComponents = pca.fit_transform(x)

# 주성분으로 이루어진 데이터 프레임 구성
principalDf = pd.DataFrame(data=printcipalComponents,
                           columns = ['principal component1','principal component2'])
principalDf
```

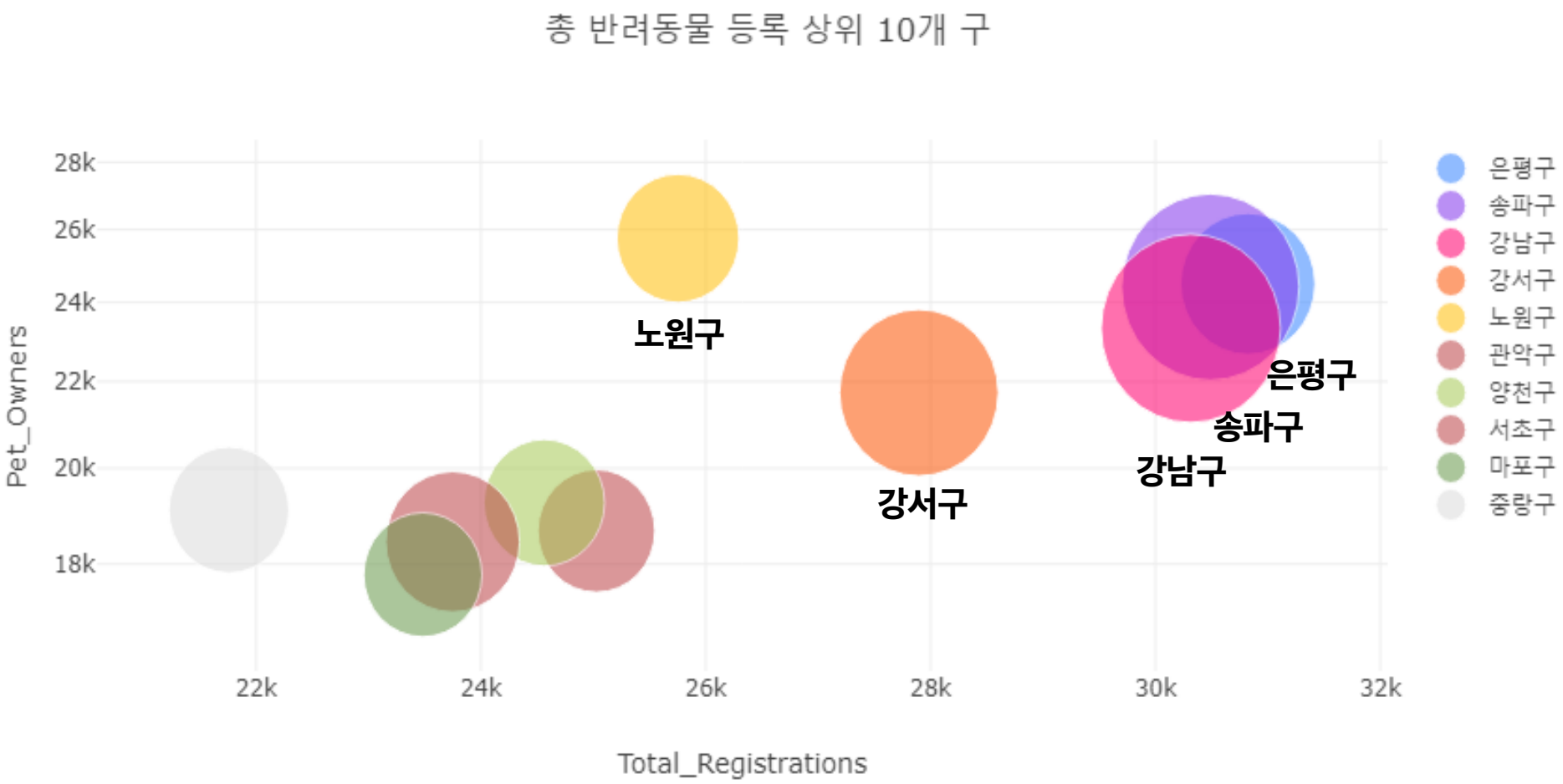
PCA 분석 피드백

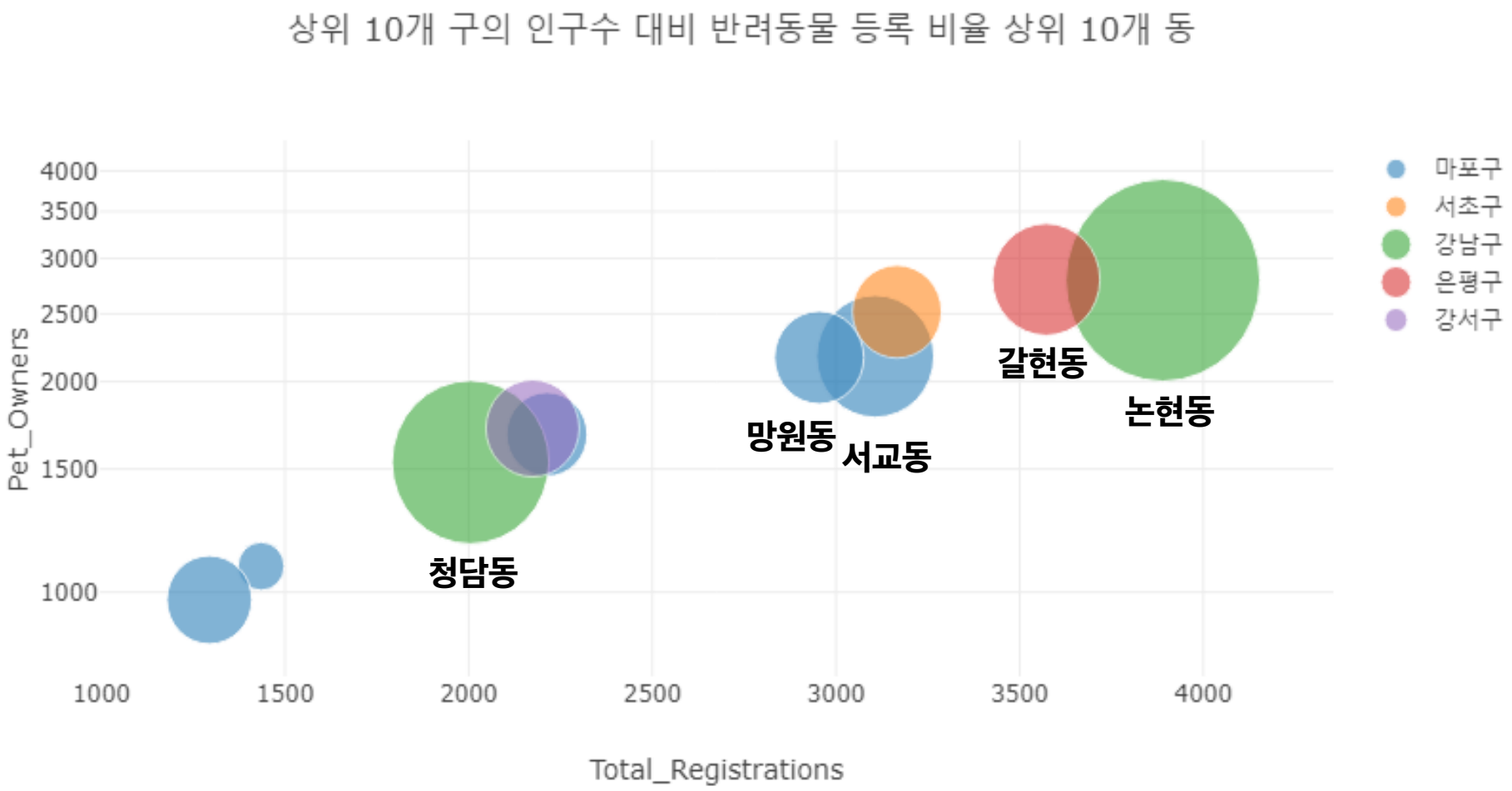
```python
from sklearn.linear_model import Ridge

alpha = 0.751216# 규제 강도 (하이퍼파라미터)
model_ridge = Ridge(alpha=alpha)
x = model2_train_x_v4
y = model2_train_y_v4
# 모델 학습
model_ridge.fit(x,y)

# 테스트 데이터로 예측
predict_ridge = model_ridge.predict(model2_test_x_v4)

# 평가지표 계산
score(model2_test_y_v4, predict_ridge)
# 232  R-squared: 0.939936572236506
```

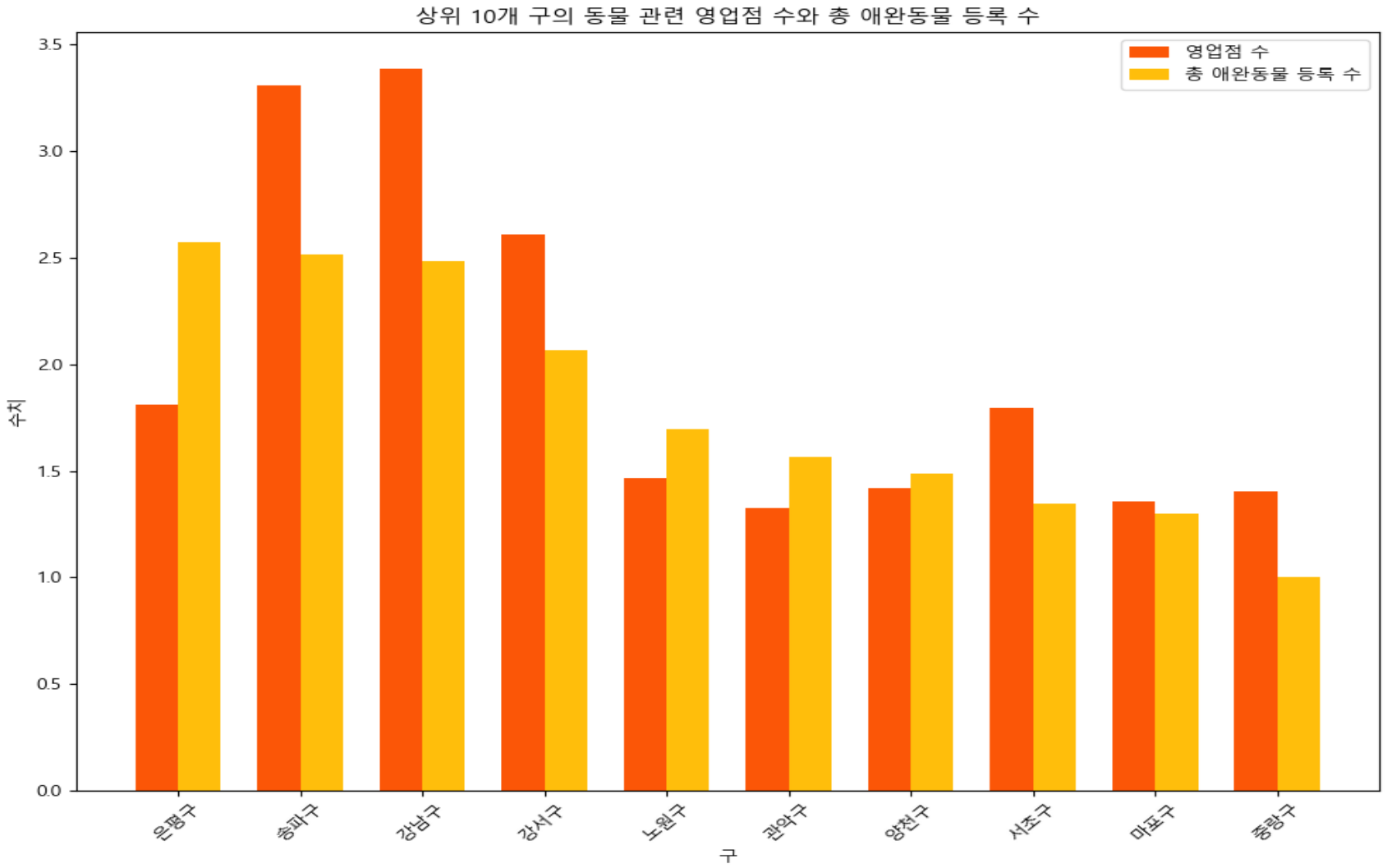
```
Mean Squared Error (MSE): 494184.1923285271
Mean Absolute Error (MAE): 551.0577162141601
R-squared: 0.9400213242499704
RMSE: 702.9823556310123
```

➔ 주성분이 2개일 때,
Ridge 모델이 R-squared 0.94로 나타남

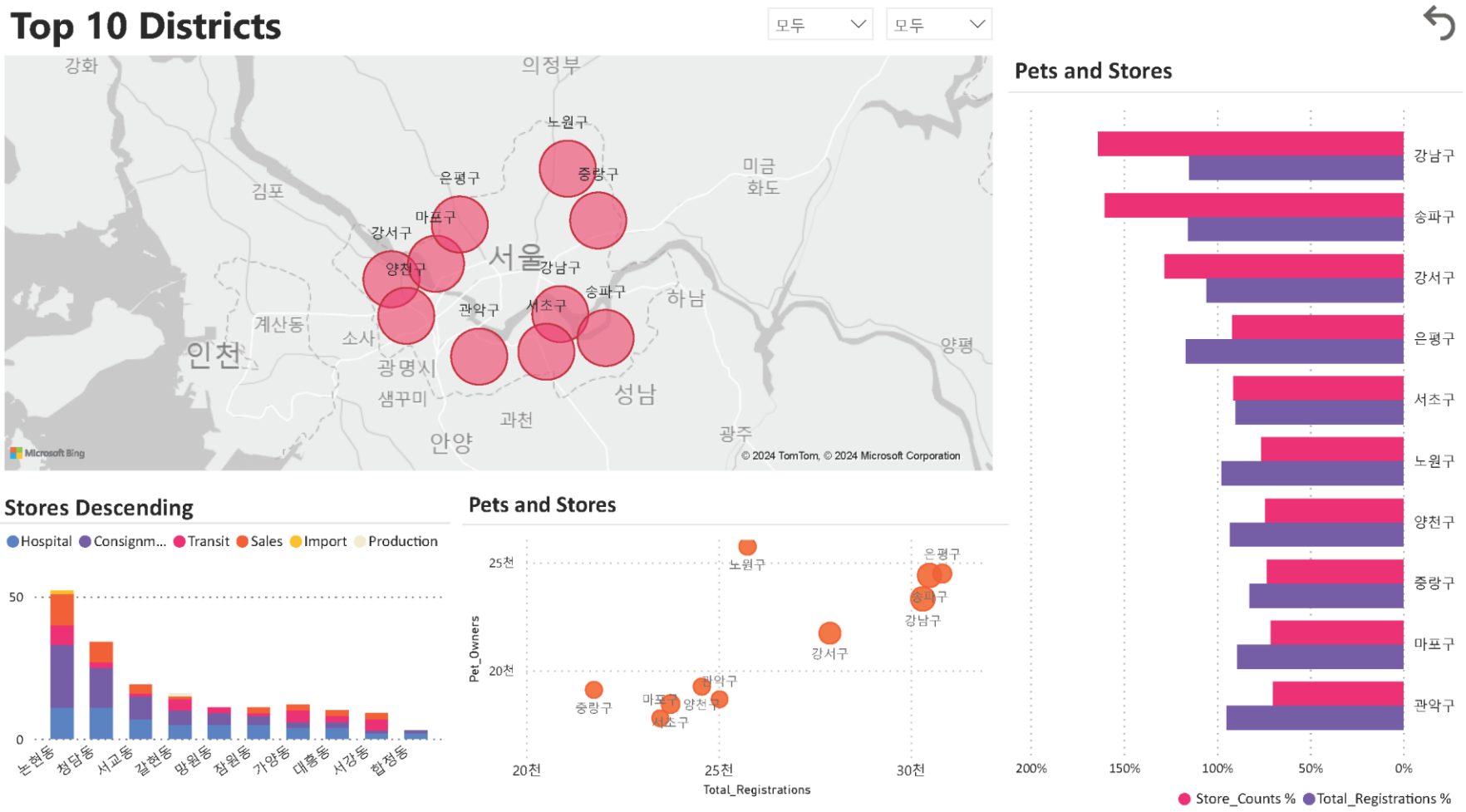

총 반려동물 등록 상위 10개 동 (범례: 구)
Pet_Owners
14k
13k
12k
11k
10k
9k
8k
7k
6k
5k
6k
8k
10k
12k
14k
Total_Registrations
상계동
화곡동
신림동
신정동
봉천동
관악구
강서구
노원구
양천구
중랑구
강북구
서초구


총 반려동물 등록 상위 10개 구
Pet_Owners
28k
26k
24k
22k
20k
18k
22k
24k
26k
28k
30k
32k
Total_Registrations
노원구
강서구
은평구
송파구
강남구
은평구
송파구
강남구
관악구
양천구
서초구
마포구
중랑구


상위 10개 구의 인구수 대비 반려동물 등록 비율 상위 10개 동
Pet_Owners
4000
3500
3000
2500
2000
1500
1000
1000
1500
2000
2500
3000
3500
4000
Total_Registrations
마포구
서초구
강남구
은평구
강서구
청담동
망원동
서교동
갈현동
논현동


상위 10개 구의 동물 관련 영업점 수와 총 애완동물 등록 수
영업점 수
총 애완동물 등록 수
수치
구
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
은평구
송파구
강남구
강서구
노원구
관악구
양천구
서초구
마포구
중랑구

Power BI


Top 10 Districts
모두
모두
강화
의정부
노원구
은평구
중랑구
김포
미금
화도
마포구
강서구
양천구
서울
강남구
송파구
관악구
서초구
하남
계산동
소사
인천
광명시
양평
샘꾸미
성남
과천
안양
광주
Microsoft Bing
© 2024 TomTom, © 2024 Microsoft Corporation
Pets and Stores
강남구
송파구
강서구
은평구
서초구
노원구
양천구
중랑구
마포구
관악구
200%
150%
100%
50%
0%
Store_Counts %
Total_Registrations %
Stores Descending
Hospital
Consignm...
Transit
Sales
Import
Production
50
0
논현동
청담동
서교동
갈현동
망원동
잠원동
가양동
대흥동
서강동
합정동
Pets and Stores
25천
20천
Pet_Owners
노원구
은평구
송파구
강남구
강서구
관악구
중랑구
마포구
양천구
서초구
20천
25천
30천
Total_Registrations

4. 개발 후기 및 느낀점

Drop D bit

① 자체 피드백

데이터의 한계

- **공공데이터**를 수집해 통합하는 과정에서 **데이터 형식의 불일치**
- **흩어진 데이터 파일**을 수집하고 통합하는 데에 상당한 시간이 소요됨
- 모든 동의 데이터를 포함해 분석하기 위해서 **결측치를 예측**하는 과정이 필요
- 데이터셋의 **모수**가 크지 않음 (약 150행)
- 요인이 되는 **피쳐 개수의 한계 (분석에 사용한 피쳐 약 5개)**

ML모델의 한계

- 상관분석, PCA 분석 등 작업을 통해 데이터셋의 피쳐를 조정하며
  최초 **0.64에서 0.95**로 **R squared** 값을 높여, 모델의 **설명력**이 높은 것으로 확인되었으나
  MAE, RMSE 등의 값이 세 자릿수 이상으로 오차에 대한 성능은 여전히 아쉬운 것으로 판단

경향성 분석의 한계

- **상권**이라는 분석 도메인의 특성상 **인구수나 관련점포수**만으로 판단하기 어려움
- 인구 피쳐에는 **반려인구의 소득 수준, 거주 형태, 가구원수** 등 다양한 인자가 복합적으로 작용하고
- 상권 분포 분석에는 **임대료, 시장성** 등 다양한 요건이 함께 고려되어야 함

4. 개발 후기 및 느낀점

② 향후 프로젝트 보완 계획 / 프로젝트 진행 소감

프로젝트 보완계획

- 반려인구수의 분포를 설명할 수 있는 **다양한 인자**에 관한 데이터를 추가 수집
- 피쳐간 **가중치**를 두고 학습
- 누적 데이터 수집 등을 통해 데이터셋의 **모수를 확대**

프로젝트 진행소감

- **진도상황을 넘어선 몇몇 파트**에서 파트별 협업으로 진행한 점 아쉬움
- 목표 달성을 위해 **학습하고 성장**
- 화상회의 등 **커뮤니케이션 빈도**를 높여 **비대면 협업 능률 향상**
- 조화로운 업무 분장과 **팀워크**

4. 개발 후기 및 느낀점

② 향후 프로젝트 보완 계획 / 프로젝트 진행 소감

참고논문

- 임성현, 2021, 결측치 대치를 활용한 신용데이터 분석방법에 관한 연구
- 박형빈 외, 2022, 딥러닝 데이터 분석을 통한 최적의 상권 입지 추천 기술 개발
- 김동준 외, 2018, 서울시 홍대상권 내 업종변화 필지의 공간적 특성 분석_젠트리피케이션
- 김영규 외, 2023, 도시재생사업에 따른 업종별 상업생존율 변화 비교 서울특별시 신촌 상권을 중심으로
- 강현모 외, 2019, 시계열 군집분석과 로지스틱 회귀분석을 이용한 골목상권 성장요인연구