빅데이터와 인공지능을 활용한 상권분석 서비스

2020년 11월 27일

팀 AI STAR

허정훈

강다미

권기호

이가영

개요

빅데이터와 인공지능을 이용하여 소매및 도매점에 한하여 무인점포를 위한 상권분석 서비스 웹페이지를 구축한다.

설명

카카오 API를 이용하여 지도를 띄우고 메인페이지를 표출한다. 사용자가 구를 검색해 동을 선택하면 인구수를 보여주고 유동인구, 경쟁업체수, 총인구, 구매력, 매출액대비 영업이익 등 5개의 변수로 생성된 방사형그래프를 지도의 오른쪽에 띄움으로써 상권분석의 결과를 보여준다. 상권분석의 척도는 좋음, 양호, 고려, 나쁨으로 구분해 사용자가 시각적으로 판단할 수 있게 해 준다.

데이터 출처

I. 상가정보 - 공공데이터 포털

부산에 있는 행정동별 상가정보를 추출하여 편의점, 대형마트, 슈퍼마켓으로 구분하여 위도 경도와 각 카테고리별 점포 수와 경쟁업체 수를 얻었다.

II. 읍면동별 세대 및 인구 - 통계청(KOSIS)

2019년도 부산시의 총 인구 수를 최종데이터에 추가했다.

III. 구군별 총 카드이용금액 - 부산시 빅데이터포털

3개월간 총 카드이용금액을 이용하여 선형회귀에 의한 예측구매력을 최종데이터에 추가했다.

IV. 부산시 읍면동별 매출액 - 통계청(KOSIS)

읍면동별 매출과 영업이익 데이터를 이용해 매출대비 영업이익 데이터를 얻었다.

V. 읍면동별 요일별 유동인구 - 부산시 빅데이터 포털

요일별 유동인구로 2019, 2020년도 총 유동인구 데이터를 얻어 사용했다.

최종데이터

데이터 출처에서 얻은 데이터를 가지고 읍면동별 총유동인구와 총인구, 편의점, 대형마트, 슈퍼마켓의 각 점포 수와 경쟁업체수, 영업이익대비 매출액, 구매력을 열로 가지는 데이터를 생성했다.

그리고 선형회귀를 이용하여 예측 유동인구와 예측 구매력을 얻었다.

선형 회귀

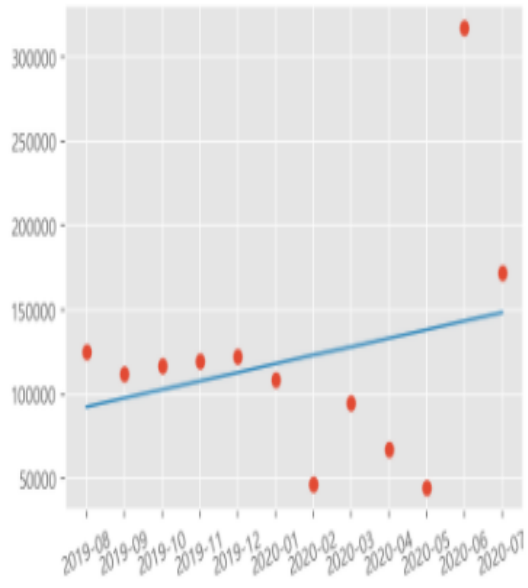
변수가 시간과 유동인구 2개뿐이었기에 LinearRegression을 통해 다음 달의 유동인구를 예측했다. 마찬가지로 구매력도 예측해볼 수 있었다.

LinearRegression 이외의 다른 model도 사용해보려 했지만, 예를 들면 RandomForest, 변수가 2개뿐이고 기간도 크지 않아 제대로 된 예측이 이루어지지 않았다.

따라서 LinearRegression이 우리의 예측에 가장 좋은 model이라고 판단했다.

```
predict_people('중앙동','총 유동인구')
```

153358명



RandomForest Regressor 사용

- 미래 시간에 대한 예측이 이루어지지 않아서 사용 불가

```
from sklearn.ensemble import RandomForestRegressor

predict_list=data[data['읍면동']!='장전3동']['읍면동']
predict_people_all2=[]
for i in predict_list:
    data['년월값']=0
    data['년월값']=pd.to_datetime(data['년월'])
    data['년월값']=data['년월값'].map(dt.datetime.toor

    X=pd.DataFrame(data[data['읍면동']==i]['년월값'])
    y=pd.DataFrame(data[data['읍면동']==i]['총 유동인

    X['년월']=X['년월값'].map(dt.date.fromordinal)
    X['년월']=pd.to_datetime(X['년월'])

    data_list=X['년월'].to_list()
    data_list
    df=pd.DataFrame({'year_month': data_list})

    X=X.reset_index()
    X['년월']= df['year_month'].dt.strftime('%Y-%m')

    line_fitter=RandomForestRegressor(max_depth=2, ran
    line_fitter.fit(X['년월값'].values.reshape(-1,1),y

    a=line_fitter.predict([[X['년월값'][11]+30]])
    predict_people_all2.append(int(np.round([[a]][-1]
```

```
line_fitter.predict([[X['년월값'][11]+30]])
```

```
array([101241.17316667])
```

```
line_fitter.predict([[X['년월값'][11]]])
```

```
array([101241.17316667])
```

방사형 그래프

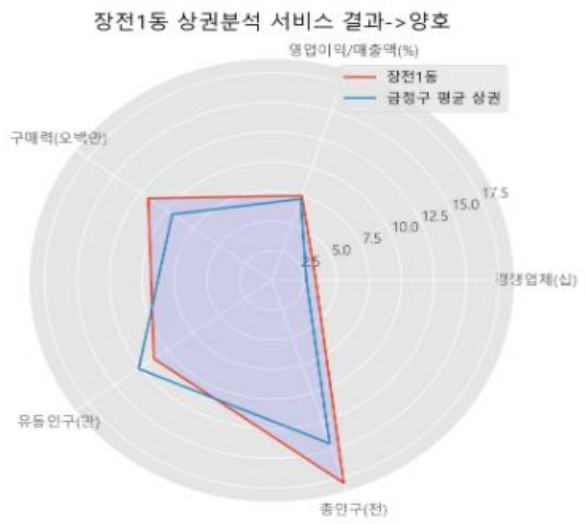
원하는 상권과 그 상권이 속한 구의 평균 상권의 차이를 구해서 각 항목의 합을 구했다.

단위가 차이 나기 때문에 개별 컬럼별로 Z-normalization을 해주고 가중치를 조절했다.

소상공인 상권분석 서비스와 비교하여 가장 비슷했던 [1.8, 1.0, 1.4, 1.0, 1.8]을 가중치로 쓰기로 결정했다. (경쟁업체, 영업이익/매출액, 구매력, 유동인구, 총인구)

하위 25% : 나쁨, 하위 25%-평균 : 고려, 평균-상위 25% : 양호, 상위 25% : 좋음이라고 평가했다.

장전1동



정답률은 42.2입니다. [1.8, 1.0, 1.2, 1.8, 1.2]
정답률은 42.7입니다. [1.8, 1.0, 1.2, 1.8, 1.4]
정답률은 42.2입니다. [1.8, 1.0, 1.2, 1.8, 1.6]
정답률은 43.2입니다. [1.8, 1.0, 1.2, 1.8, 1.8]
정답률은 42.7입니다. [1.8, 1.0, 1.2, 1.8, 2.0]
정답률은 43.7입니다. [1.8, 1.0, 1.2, 2.0, 1.0]
정답률은 42.7입니다. [1.8, 1.0, 1.2, 2.0, 1.2]
정답률은 43.2입니다. [1.8, 1.0, 1.2, 2.0, 1.4]
정답률은 42.2입니다. [1.8, 1.0, 1.2, 2.0, 1.6]
정답률은 43.2입니다. [1.8, 1.0, 1.2, 2.0, 1.8]
정답률은 43.2입니다. [1.8, 1.0, 1.2, 2.0, 2.0]
정답률은 43.2입니다. [1.8, 1.0, 1.4, 1.0, 1.0]
정답률은 43.2입니다. [1.8, 1.0, 1.4, 1.0, 1.2]
정답률은 42.7입니다. [1.8, 1.0, 1.4, 1.0, 1.4]
정답률은 43.7입니다. [1.8, 1.0, 1.4, 1.0, 1.6]
정답률은 46.2입니다. [1.8, 1.0, 1.4, 1.0, 1.8]
정답률은 44.2입니다. [1.8, 1.0, 1.4, 1.0, 2.0]
정답률은 41.7입니다. [1.8, 1.0, 1.4, 1.2, 1.0]
정답률은 43.2입니다. [1.8, 1.0, 1.4, 1.2, 1.2]
정답률은 43.2입니다. [1.8, 1.0, 1.4, 1.2, 1.4]
정답률은 42.2입니다. [1.8, 1.0, 1.4, 1.2, 1.6]
정답률은 44.2입니다. [1.8, 1.0, 1.4, 1.2, 1.8]
정답률은 45.2입니다. [1.8, 1.0, 1.4, 1.2, 2.0]
정답률은 42.2입니다. [1.8, 1.0, 1.4, 1.4, 1.0]
정답률은 41.7입니다. [1.8, 1.0, 1.4, 1.4, 1.2]
정답률은 41.7입니다. [1.8, 1.0, 1.4, 1.4, 1.4]
정답률은 41.7입니다. [1.8, 1.0, 1.4, 1.4, 1.6]

프로젝트 총괄 코드

Python을 사용해서 작업하기 위해 Jupyter Notebook에서 작업을 했다.

1. 데이터 원본, 만든 데이터를 로드해 사용

```
ori_data19=pd.read_csv("./data/2019동별 유동인구.csv", encoding='utf-8')
ori_data20=pd.read_csv("./data/2020동별 유동인구.csv", encoding='cp949')
ori_data_total=pd.read_csv('./data/2019년도 부산시 총인구(동별).csv', encoding='cp949')
ori_data_money=pd.read_csv("./data/부산_구군별 총 카드이용금액.csv", encoding='utf-8')
ori_data_rival=pd.read_csv("./data/부산_읍면동별_경쟁업체수.csv", encoding='cp949')
ori_data_benifit=pd.read_csv('./data/부산_읍면동별_매출대비이익.csv',encoding='cp949')
ori_data_do=pd.read_csv("./data/save/상가정보_부산_코드구분.csv", encoding='cp949')
```

2. LinearRegression

데이터 프레임 X에 년월값을 ordinal 값으로 주고 y 값에 예측하고 싶은 데이터를 넣어서 원하는 데이터를 얻었다.

```
predict_list=data[data['읍면동']!='장전3동']['읍면동'].unique().tolist()
predict_people_all=[]
for i in predict_list:
    data['년월값']=0
    data['년월값']=pd.to_datetime(data['년월'])
    data['년월값']=data['년월값'].map(dt.datetime.toordinal)

    X=pd.DataFrame(data[data['읍면동']==i]['년월값'])
    y=pd.DataFrame(data[data['읍면동']==i]['총 유동인구'])

    X['년월']=X['년월값'].map(dt.date.fromordinal)
    X['년월']=pd.to_datetime(X['년월'])

    data_list=X['년월'].to_list()
    data_list
    df=pd.DataFrame({'year_month': data_list})

    X=X.reset_index()
    X['년월']= df['year_month'].dt.strftime('%Y-%m')

    line_fitter=LinearRegression()
    line_fitter.fit(X['년월값'].values.reshape(-1,1),y)

    a=line_fitter.predict([[X['년월값'][11]+30]])
    predict_people_all.append(int(np.round([[a]][-1][-1])))
```

3. Z-정규화를 통해 단위를 통일

```
def z_score_normalize(lst):
    normalized = []
    for value in lst:
        normalized_num = (value - np.mean(lst)) / np.std(lst)
        normalized.append(normalized_num)
    return normalized
```

4. 등급 설정을 위한 가중치 조절 작업

- grade_store_list밑의 1.8, 1.4, 1.8 부분의 숫자들을 바꿔주는 방법을 사용한다.

```
    grade_store_list=[]
    for i in range(len(nor_want_s)):
        if i==0:
            grade_store_list.append(-(nor_want_s[i]-nor_ave_s[i])*1.1)
        elif i==1:
            grade_store_list.append((nor_want_s[i]-nor_ave_s[i])*1.2)
        elif i==4:
            grade_store_list.append((nor_want_s[i]-nor_ave_s[i])*1.3)
        else:
            grade_store_list.append((nor_want_s[i]-nor_ave_s[i]))
```

5. 함수 사용법

- 총괄 파일 코드의 맨 위의 Function 마크다운을 참고

Function

- def predict_people(city, want):
  - city:읍면동, want:총 유동인구
  - ex) predict_people('장전1동', '총 유동인구')
  - 읍면동의 총 유동인구를 예측하여 시각화 해주는 함수
- def predict_money_power(gu):
  - gu:구군
  - ex) predict_money_power('금정구')
  - 구군의 구매력을 예측하여 시각화 해주는 함수
- def z_score_normalize(lst):
  - lst:Z-정규화를 시키고 싶은 리스트
  - 리스트를 Z-정규화 시켜줌
- def analysis_store(dong):
  - dong: 읍면동
  - 방사형 그래프를 시각화 시켜줌

```
def predict_people(city, want):
    data['년월값']=0
    data['년월값']=pd.to_datetime(data['년월'])
    data['년월값']=data['년월값'].map(dt.datetime.toordinal)

    X=pd.DataFrame(data[data['읍면동']==city]['년월값'])
    y=pd.DataFrame(data[data['읍면동']==city][want])

    X['년월']=X['년월값'].map(dt.date.fromordinal)
    X['년월']=pd.to_datetime(X['년월'])

    data_list=X['년월'].to_list()
    data_list
    df=pd.DataFrame({'year_month': data_list})

    X=X.reset_index()
    X['년월']= df['year_month'].dt.strftime('%Y-%m')

    line_fitter=LinearRegression()
    line_fitter.fit(X['년월값'].values.reshape(-1,1),y)

    predict=line_fitter.predict([[X['년월값'][11]+30]])
    print(str(int(np.round([[predict]][-1][-1])))+'명')

    plt.plot(X['년월'], y, 'o')
    plt.plot(X['년월'],line_fitter.predict(X['년월값'].values.reshape(-1,1)))
    plt.xticks(rotation=20)
    plt.show()
# line_fitter.coef_, line_fitter.intercept_
```

개발환경

[서버 사양]

서버 : AWS t2.micro 인스턴스
- CPU 1, 메모리 1GiB, i386, x86_64
- ubuntu 18.04
- IP주소 : 52.78.155.149

Service Domain: http://52.78.155.149:5000

[개발 환경]

파이썬 : python3.8.5
파이썬 라이브러리 : flask, pymysql

데이터베이스 : db.t2.micro
- mysql 8.0.20
- 엔드 포인트 : database-1.c4y1s8zaxihb.ap-northeast-2.rds.amazonaws.com
- 계정 : admin, 비밀번호 : mypassword

[코드 저장소]
https://github.com/Kangdamii/yproject

[파일 구조]
yproject
- code
    - static
        - img : 상권분석 결과 출력을 위한 이미지
            - 고려.png
            - 나쁨.png
            - 양호.png
            - 좋음.png
        - json : json 파일이 저장돼 있음
            - geo.json : polygon좌표가 포함된 json파일
    - templates
        - main.html : 웹 메인페이지 html
- __init__.py : main flask 코드

[테이블 사용]

```
mysql> show tables;
+---------------------------+
| Tables_in_retail_info     |
+---------------------------+
| busan_dong_analysis       |
| busan_dong_analysis_graph |
| busan_dong_boundary       |
| busan_dong_population     |
| busan_store               |
| busan_store_select2       |
+---------------------------+
6 rows in set (0.02 sec)
```

busan_store : 구와 카테고리를 선택했을 때 마커 출력을 위한 상가 좌표, 주소, 이름을 출력하기 위해 사용

busan_dong_population : 부산의 구에 속한 행정동 정보와 동별 인구를 출력하기 위해 사용

busan_dong_analysis_graph : 상권 분석 결과를 방사형 그래프로 출력을 하기위해 사용

Test Case Demo Desc

그림1 [메인 페이지1]

사이트에 처음 접속했을 때 볼 수 있는 메인 페이지이다.

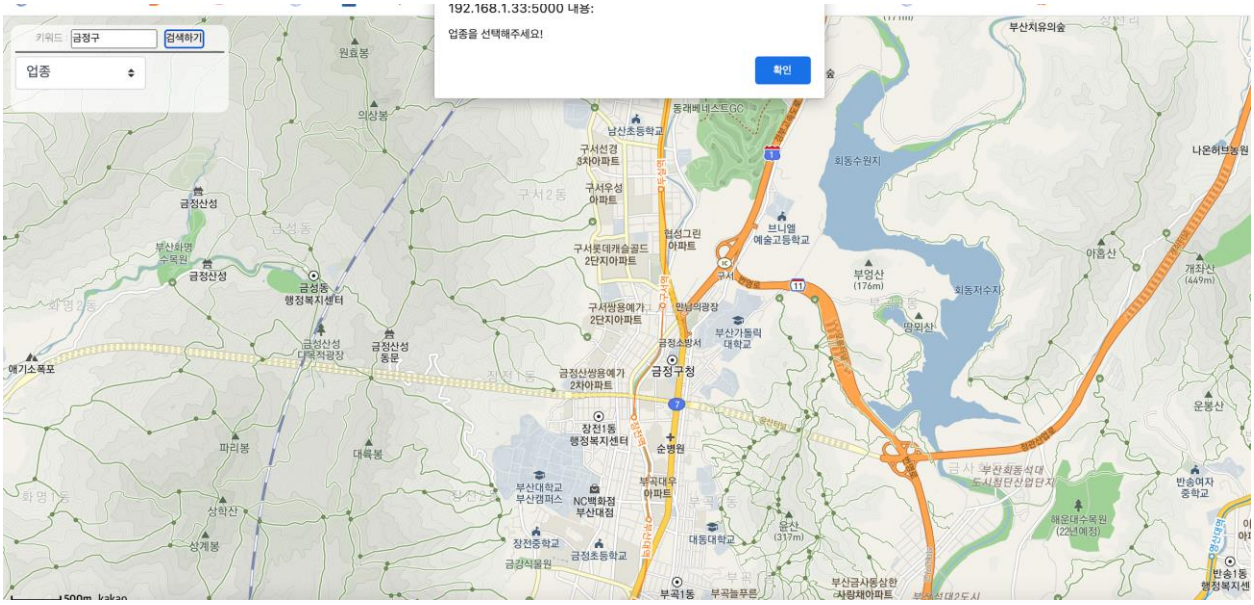


그림2 [메인 페이지2]

사이트에 처음 접속했을 때 볼 수 있는 메인 페이지이다. 구를 입력하고 업종을 선택하지 않았을 시 화면이다.

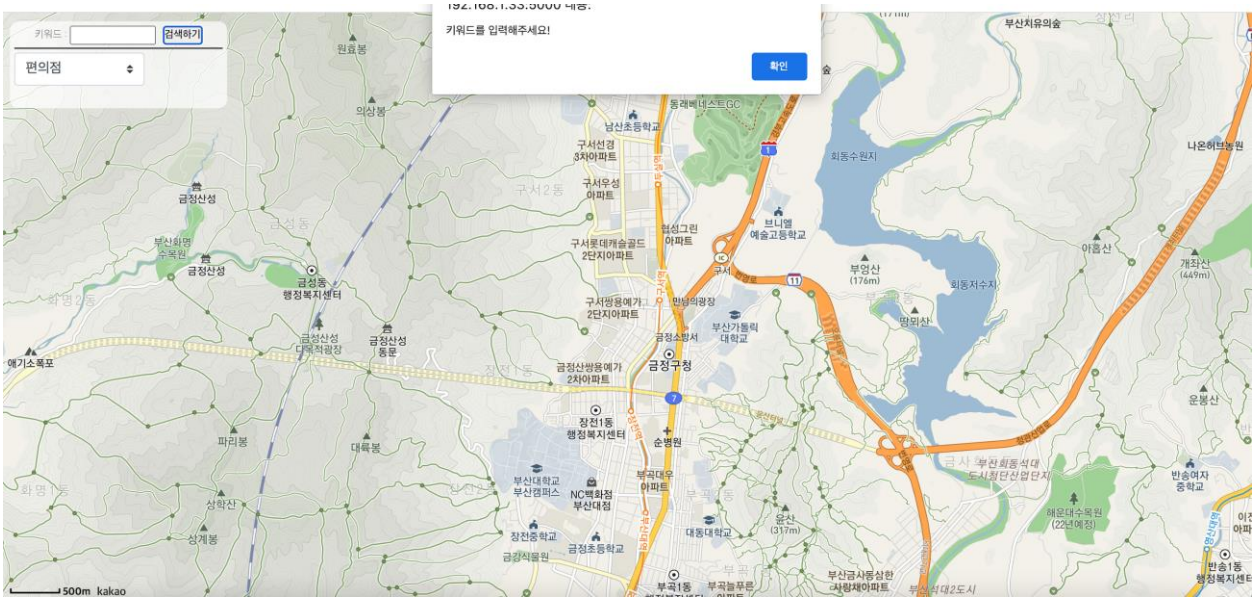


그림3 [메인 페이지3]

사이트에 처음 접속했을 때 볼 수 있는 메인 페이지이다. 구를 입력하지 않고 카테고리만 설정하고 클릭 했을 시 화면이다.

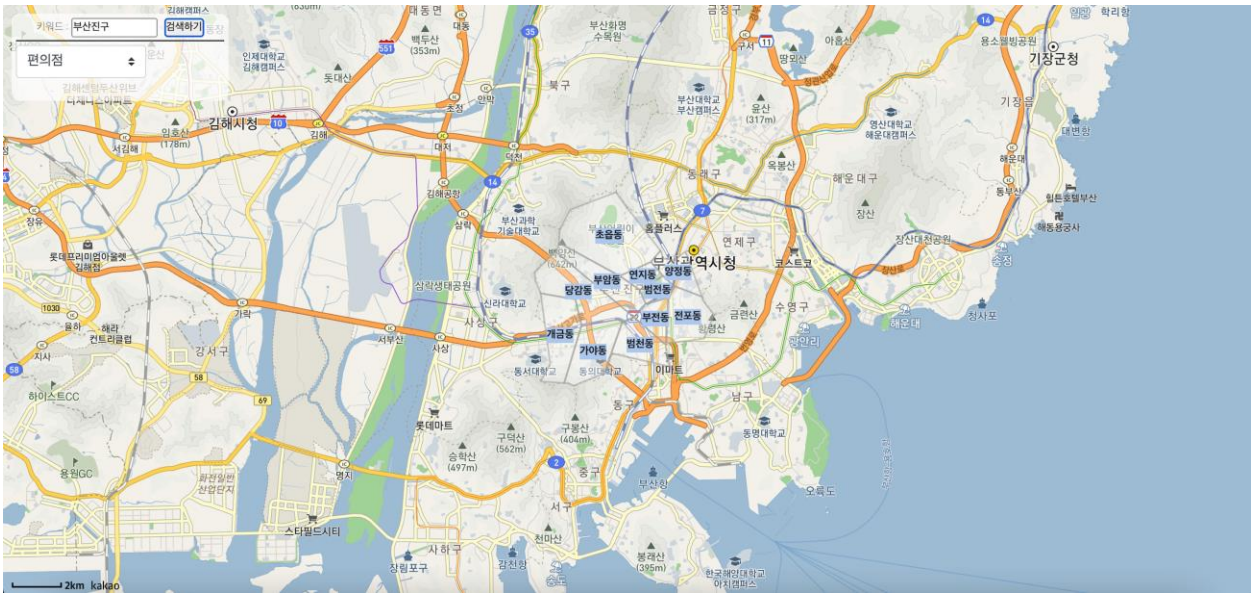


그림4 [동 경계표시]

구와 카테고리를 선택하고 검색을 하면 해당 구의 경계와 해당 동을 표시한다.

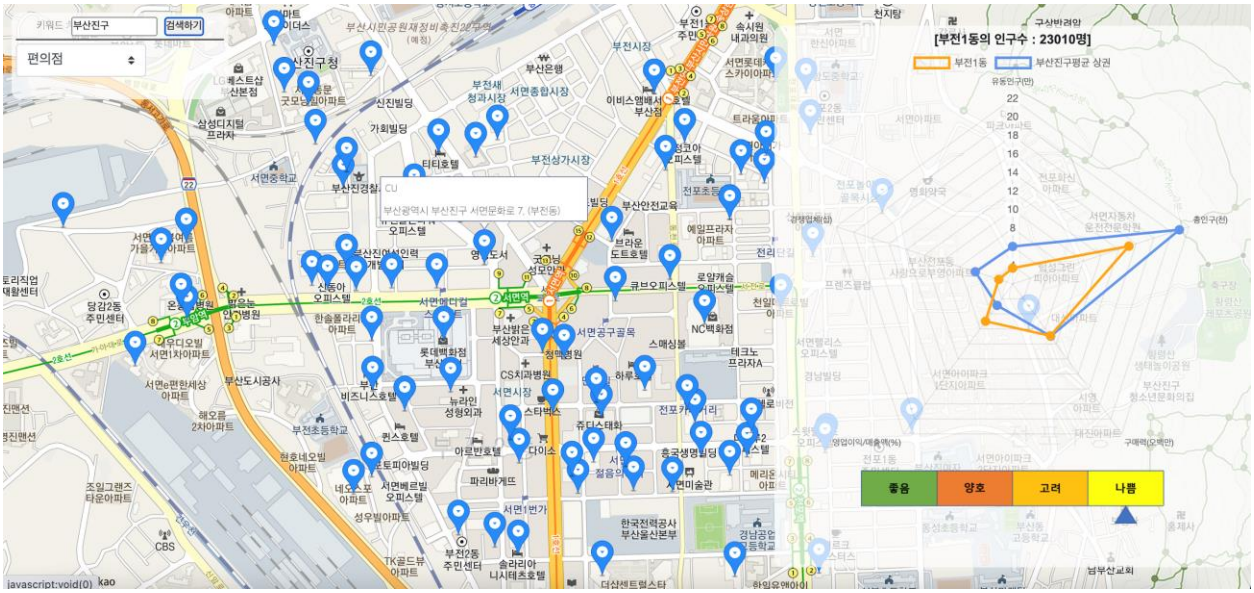


그림5 [동 상세표시]

동을 선택하여 지도를 확대 하면 상가를 나타내는 마커가 출력되고 해당 동의 상권분석 결과를 그래프로 보여준다. 상권분석 평가는 위의 4가지 형태 중 하나로 나타낸다.

그림6 [동 상세표시2]

동을 선택하여 지도를 확대 하면 상가를 나타내는 마커가 출력되고 해당 동의 상권분석 결과를 그래프로 보여준다. 상권분석 평가는 위의 4가지 형태 중 하나로 나타낸다. 해당 그림은 금정구 구서1동에서 편의점을 선택했을 시 출력 화면이다.