

# 회귀분석 실습

C조





1. 주제 설정

2. 자료 수집, Data 제작 과정

3. 분석 과정과 코드 설명

4. 결과값

5. 한계와 의의



# 목차

# 주제 설정(1)

- 조원의 관심사인 마케팅 분야에서 적절한 주제를 찾기 위해 논의
  - A. 특정 상품이 출시 됐을 때의 고객의 구매 여부 예측
    - > 연속형 값이 아님
  - B. 상품의 적정 용량 / 가격 / 생산량 예측
    - > 단순히 값을 예측하는 것은 가능하나  
그 값이 최고 효율을 가지는 지는 판단할 수 X

# 주제 설정(2)

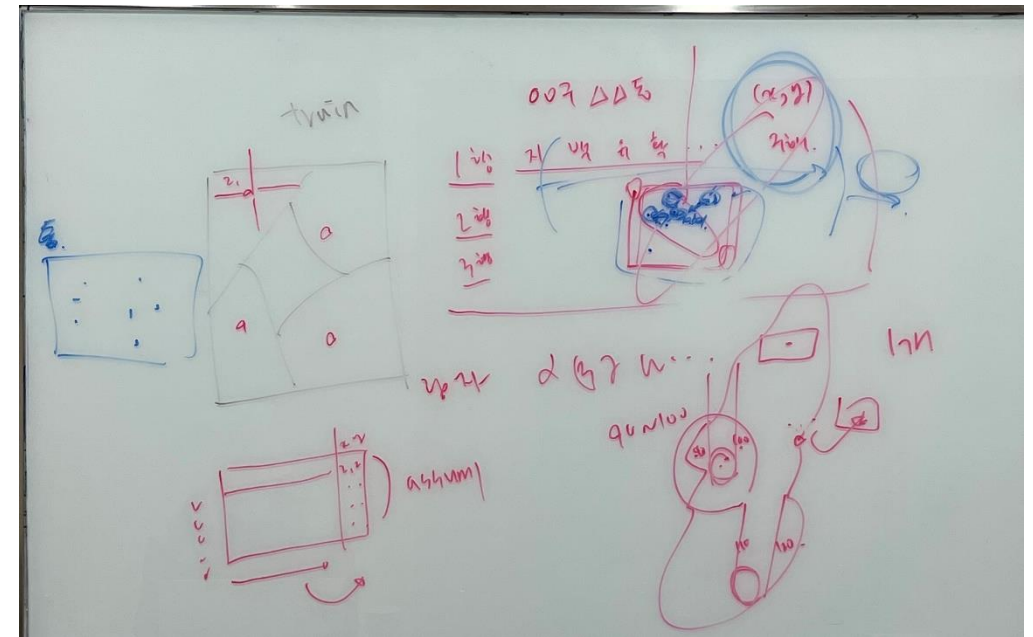
- 선별진료소 장소 예측

1. 축적을 통일한 동일한 범위의 정사각형 구역들을 설정

2. 그 안에서 선별진료소와 feature의 좌표 표시

3. 해당 좌표들을 이용해 특정 한 구역의 선별진료소 위치를 예측

-> 그 구역의 실제 선별진료소 위치와 동일한지 비교



## 주제 설정(2)-2

- 선별진료소의 위치와 관련이 있을 것으로 보이는 feature 분석
  - A. 교통수단(: 버스 정류장, 지하철 역 등)
  - B. 고층 건물(: 회사, 백화점 등)
  - C. 넓은 공간(: 공원, 공터 등)

-> 선별진료소는 건물 내 설치가 불가하여 조건이 까다로움  
제약조건이 상대적으로 적고 매장 수가 많아 data를 수집하기 유리한  
스타벅스의 위치를 예측하는 것으로 변경

# 주제 설정(3)

- Feature 설정

: 스타벅스 입점 원칙에 가장 중요한 요인은 무엇일까?

→ 유동인구

- 유동인구에 영향을 주는 feature 탐색

1차) 프랜차이즈, 쇼핑몰, 공공시설, 역, 회사, 음식점

2차) 이디야 커피, 다이소, 올리브영

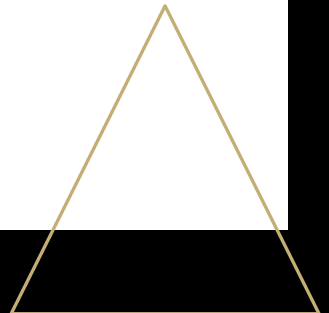
# 주제 설정(3)-2

- 1차 feature들의 문제점
  - 쇼핑몰: 네이버 지도 검색 기준으로 '쇼핑몰' 검색
    - 중소형 슈퍼마켓 같은 애매한 결과 多
  - 공공시설: 공원과 병원 등은 규모와 용도가 제각각이라 설득력 떨어짐
  - 음식점, 회사 건물: 수가 너무 많고 기준이 모호하여 설득력 떨어짐
- feature로 채택하는 기준 설정에 난관 + 유동인구와의 관련성 모호의 이유로 기각.
- 구체화 필요



# 주제 설정(3)-3

- 2차 feature 설정 이유
  - 이디야 커피: 입점 조건이 스타벅스 근처
  - 다이소: 유동인구가 많은 프랜차이즈
  - 올리브영: 유동인구가 많은 프랜차이즈
- 3개 모두 지하철 역 근처에 많고 유동인구가 많음





# 자료 수집

- 자료 수집 과정
  1. 네이버 지도에 지하철 역 검색
  2. 축척은 100m로 설정
  3. 스타벅스와 feature들이 모두 존재하는지 확인
    - 기준: 검색 했을 때 지도가 움직이지 않음
    - : 여러 개일 경우 역과 가장 가까운 시설로 설정
  4. 시설이 존재하면 맥북 내 기능(모니터 내에서 마우스 포인터의 위치의 좌표 표시) 을 이  
용해 각 좌표 입력
    - 좌표는 위도/경도 역할을 함



신속항원검사 선별진료소 음식점 카페 네이버주문 캠핑장 ...

지도 홈

길찾기

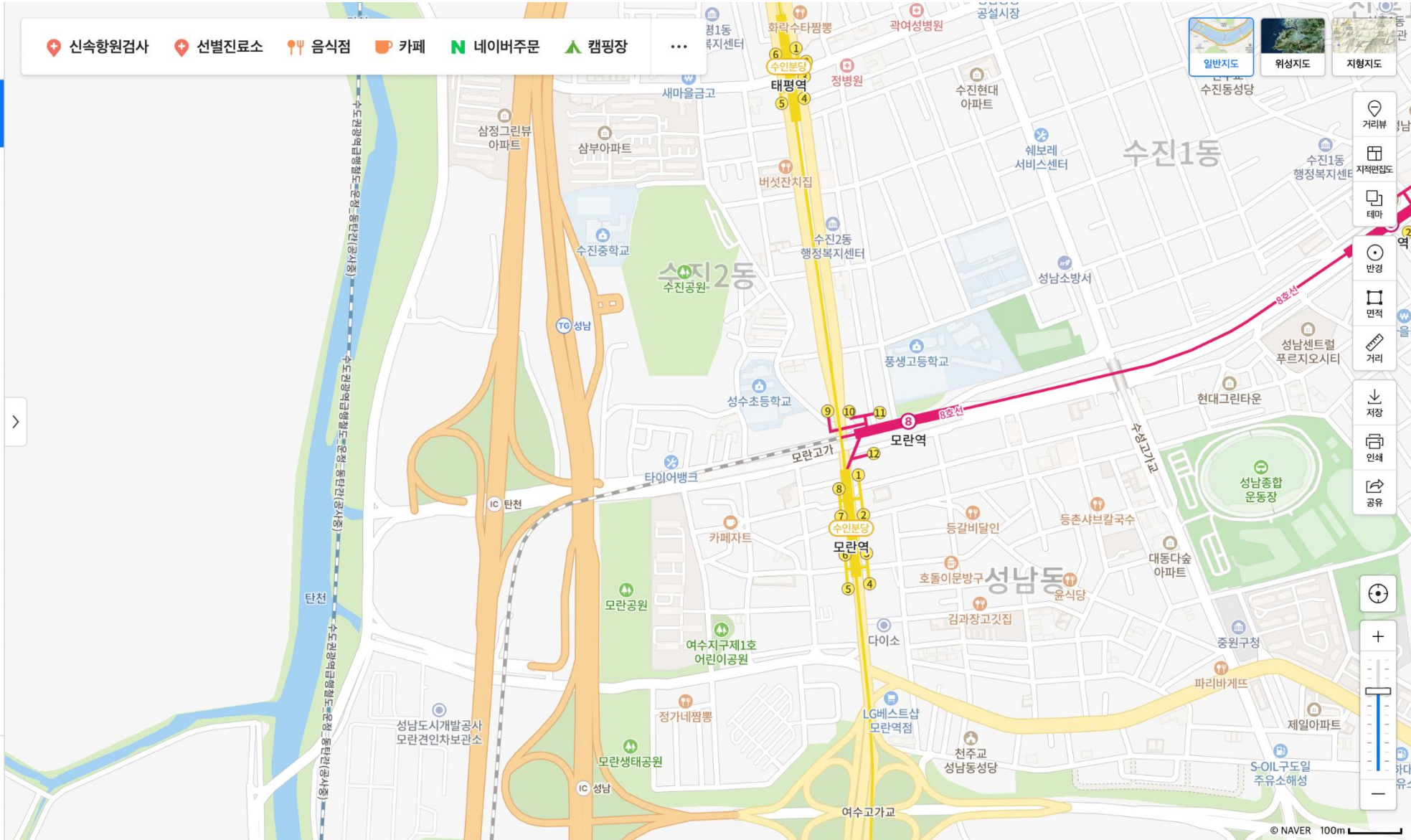
버스

지하철

기차

즐거찾기

더보기



© NAVER 100m



지도 홈

길찾기

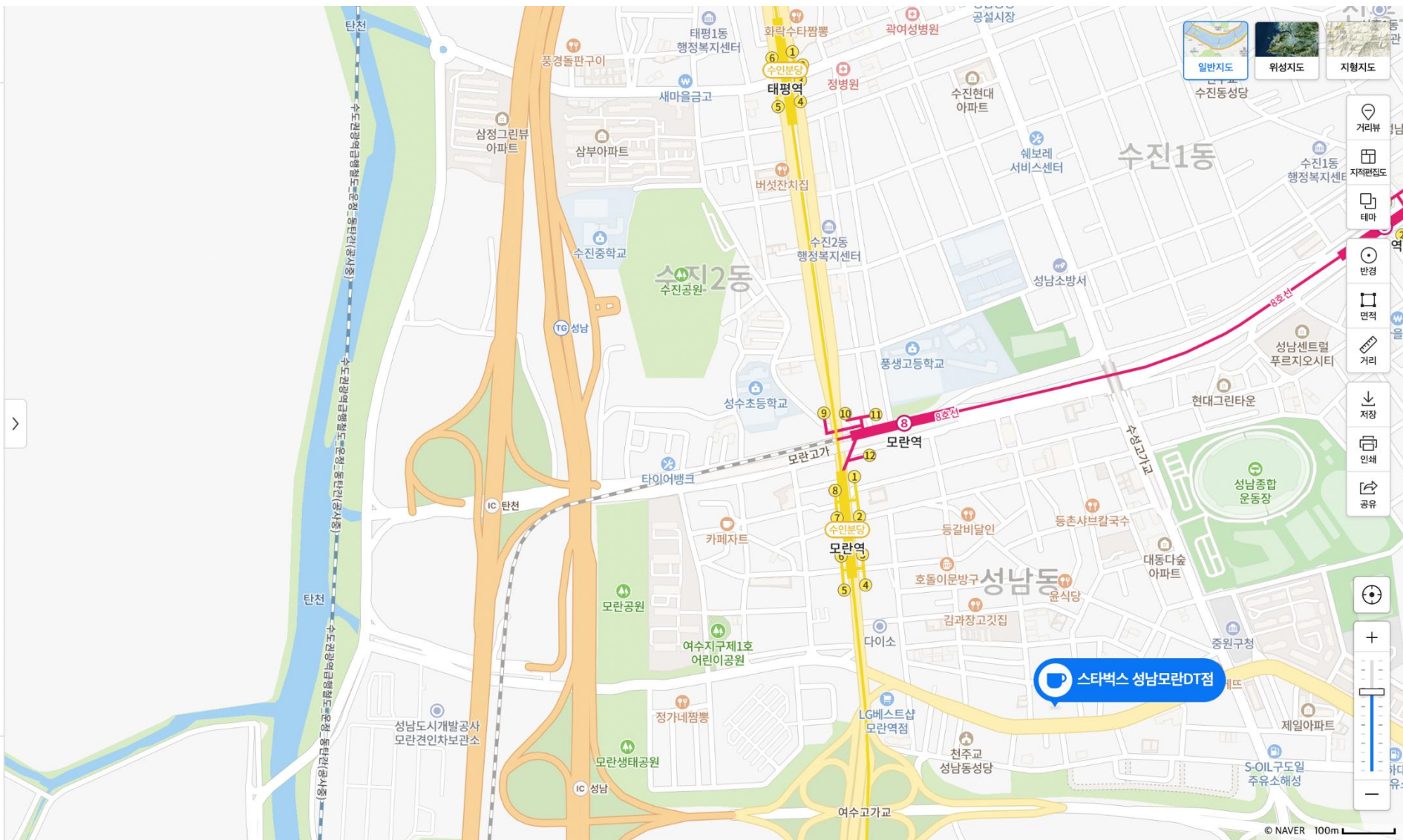
버스

지하철

기차

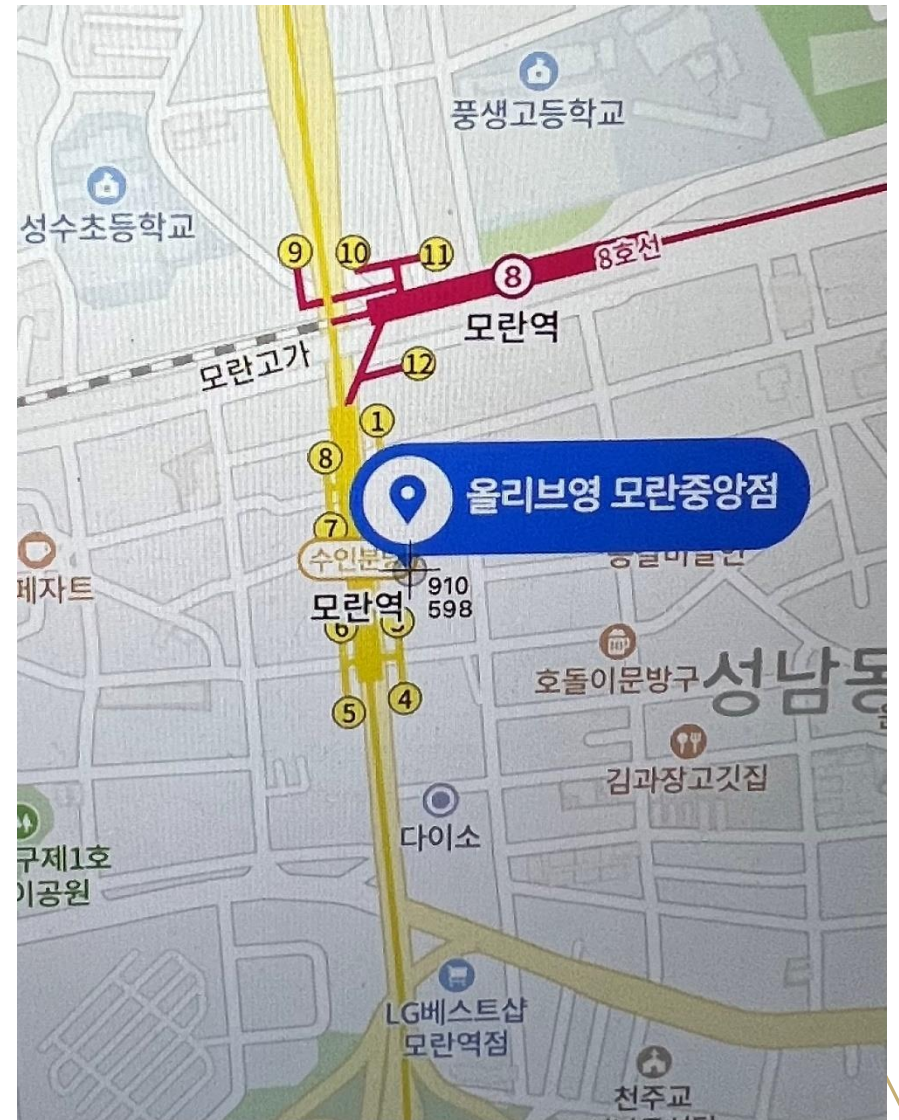
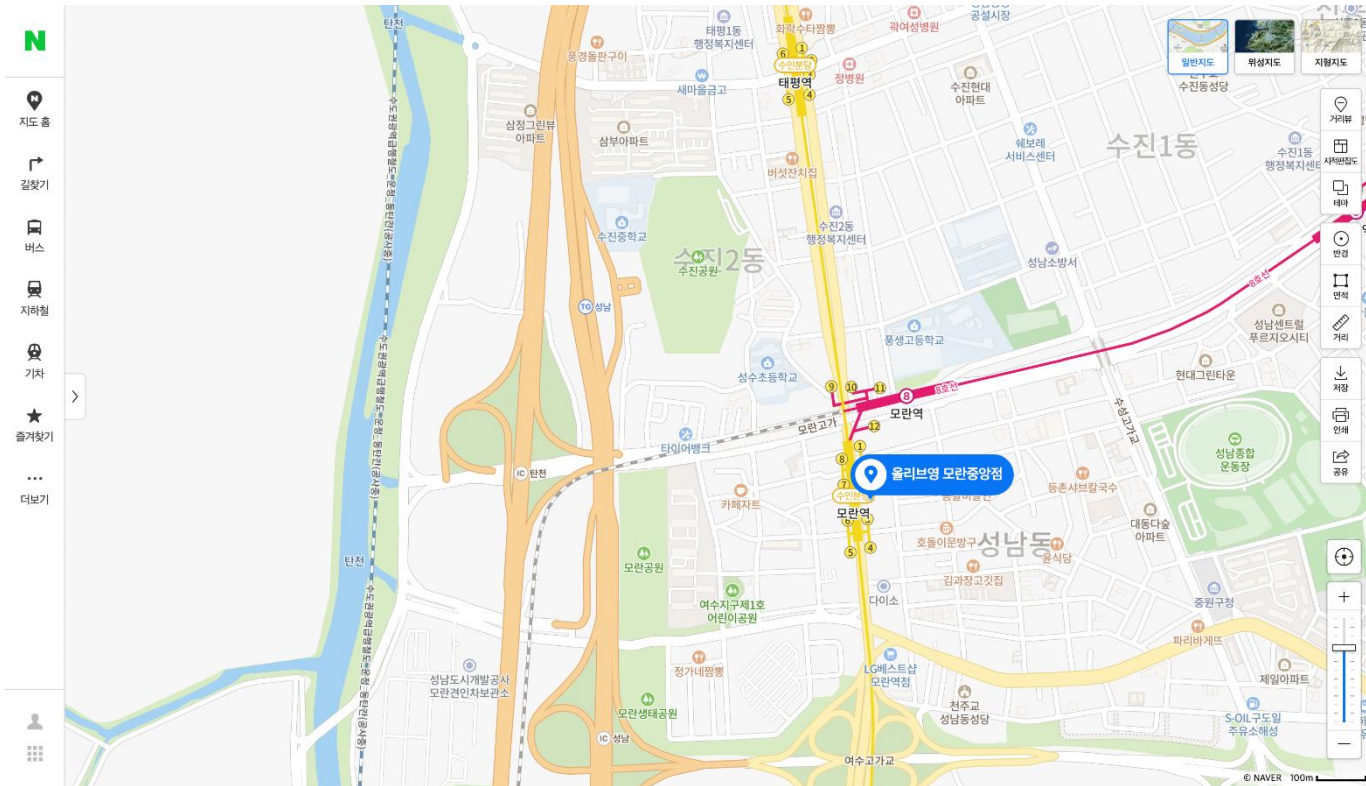
즐거찾기

더보기



© NAVER 100m





name	s_longitude	s_latitude	e_longitude	e_latitude	d_longitude	d_latitude	o_longitude	o_latitude	subway
양재역	1081	624	975	556	1080	496	1032	489	신분당
정자역	865	544	814	573	883	619	892	618	신분당
미금역	929	595	999	509	1029	519	916	517	신분당
수지구청역	965	518	954	531	973	433	964	461	신분당
성북역	940	530	899	514	969	420	992	517	신분당
상현역	946	430	985	526	892	582	962	485	신분당
강남구청	942	503	912	517	941	472	920	436	분당선
선정릉	1000	456	911	517	1325	797	976	465	분당선
선릉	910	523	892	416	1052	469	845	548	분당선
한티역	1065	455	1099	387	1197	366	930	409	분당선
모란역	1090	750	943	551	923	691	909	598	분당선
야탑역	994	520	854	540	996	467	1003	441	분당선
서현역	941	519	1002	571	861	545	977	449	분당선
수내역	930	527	864	541	1033	603	887	567	분당선
죽전역	802	668	950	440	1000	388	1210	718	분당선
신갈역	929	799	953	321	970	843	953	830	분당선
구성역	975	548	835	215	986	561	947	509	분당선
수원시청역	956	643	985	437	1062	286	964	595	분당선
중랑역	867	562	875	545	927	530	932	532	경춘선
망우역	987	612	888	599	1262	533	978	619	경춘선
갈매역	999	502	1048	556	962	632	1031	665	경춘선
남춘천역	809	282	964	470	822	371	721	297	경춘선
노원역	957	596	892	519	899	523	893	495	4호선
창동역	897	556	875	493	1091	378	894	522	4호선
쌍문역	956	503	947	441	956	433	956	500	4호선
수유역	977	479	1009	400	974	480	933	457	4호선
미아역	965	506	920	450	982	778	973	533	4호선
미아사거리역	970	574	931	473	974	713	935	509	4호선
길음역	961	557	640	505	673	510	959	559	4호선

- 이렇게 수집한 좌표 값들을 엑셀 파일에 입력해 data 생성

# 코드 설명



```
1 import pandas as pd
2 import numpy as np
3
4 df1 = pd.read_csv("train1.csv")
5 df2 = pd.read_csv("train2.csv")
6 df1 = df1.iloc[:100, 1:-2]
7 df2 = df2.iloc[:, 1:-2]
8 df = pd.concat([df1, df2], axis = 0)
```

- Dataframe 결합

→ 조사 과정에서 기입했던 역 이름,  
호선 등은 지우고 필요한 값만 남김

→ 결합은 concat 메소드 사용

→ axis를 0으로 설정하여 데이터가  
위아래로 합쳐지도록 유도



```
1 X_1, X_2 = df[["e_longitude", "d_longitude", "o_longitude"]], df[["e_latitude", "d_latitude", "o_latitude"]]
2 y_1, y_2 = df["s_longitude"], df["s_latitude"]
3
4
5 test1 = pd.DataFrame({"e_longitude" : [922], "d_longitude" : [911], "o_longitude" : [1002]})
6 test2 = pd.DataFrame({"e_latitude" : [463], "d_latitude" : [460], "o_latitude" : [470]})
7
8 S = [[1040, 602], [1032, 420], [1120, 594]]
```

- 좌표 값 할당

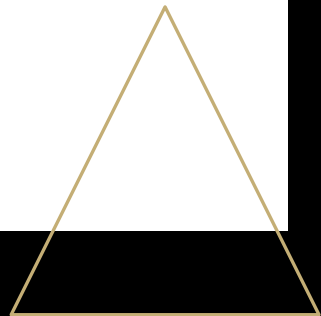
X\_1: feature들의 좌표 x값 | X\_2: feature들의 좌표 y값

y\_1: 스타벅스의 좌표 x값 | y\_2: 스타벅스의 좌표 y값

test1: 건대입구역 feature들의 좌표 x값

test2: 건대입구역 feature들의 좌표 y값

S: 건대입구역 근처 스타벅스 3개의 좌표



```
1  from sklearn.linear_model import LinearRegression
2  from sklearn import svm
3
4  from xgboost import XGBRegressor
5  from lightgbm import LGBMRegressor
6
7  from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
8
9  Model = [4), svm.SVR(), XGBRegressor(), LGBMRegressor(), RandomForestRegressor(), GradientBoostingRegressor()]
10 temp1, temp2 = [], []
11 for i in range(len(Model)):
12     for j in range(len(Model)):
13         if i == j:
14             continue
15         else:
16             model1, model2 = Model[i], Model[j]
17             model1.fit(X_1, y_1)
18             model2.fit(X_2, y_2)
19             temp1.append(model1.predict(test1)[0])
20             temp2.append(model2.predict(test2)[0])
```

- 모델 불러옴
- temp1,2 → 위도, 경도 각각의 예측값 넣는 list
- If문  
→ i=j 일 때는 continue  
→ 아닐 경우에는 i번째 모델과 j번째 모델을 model1과 model2에 각각 적용



- 같은 모델로 돌렸을 때의 예측값 구하는 코드

```
1 model1 = LinearRegression()
2 model2 = LinearRegression()
3 model1.fit(X_1, y_1)
4 model2.fit(X_2, y_2)
5 a, b = model1.predict(test1), model2.predict(test2)
6 temp1.append(a[0]), temp2.append(b[0])
```

```
1 model1 = svm.SVR()
2 model2 = svm.SVR()
3 model1.fit(X_1, y_1)
4 model2.fit(X_2, y_2)
5 a, b = model1.predict(test1), model2.predict(test2)
6 temp1.append(a[0]), temp2.append(b[0])
```

```
1 model1 = XGBRegressor()
2 model2 = XGBRegressor()
3 model1.fit(X_1, y_1)
4 model2.fit(X_2, y_2)
5 a, b = model1.predict(test1), model2.predict(test2)
6 temp1.append(a[0]), temp2.append(b[0])
```

```
1 model1 = LGBMRegressor()
2 model2 = LGBMRegressor()
3 model1.fit(X_1, y_1)
4 model2.fit(X_2, y_2)
5 a, b = model1.predict(test1), model2.predict(test2)
6 temp1.append(a[0]), temp2.append(b[0])
```

```
1 model1 = RandomForestRegressor()
2 model2 = RandomForestRegressor()
3 model1.fit(X_1, y_1)
4 model2.fit(X_2, y_2)
5 a, b = model1.predict(test1), model2.predict(test2)
6 temp1.append(a[0]), temp2.append(b[0])
```

```
1 model1 = GradientBoostingRegressor()
2 model2 = GradientBoostingRegressor()
3 model1.fit(X_1, y_1)
4 model2.fit(X_2, y_2)
5 a, b = model1.predict(test1), model2.predict(test2)
6 temp1.append(a[0]), temp2.append(b[0])
```

```

1  res = pd.DataFrame({"x" : temp1, "y" : temp2})
2  res["D"] = 100000000
3  res["idx"] = -1
4  for i in range(len(res)):
5      for s in S:
6          temp = ((s[0] - res["x"][i])**2 + (s[1] - res["y"][i])**2)**0.5
7          if res["D"][i] >= temp:
8              res["D"][i] = temp
9              res["idx"][i] = S.index(s)

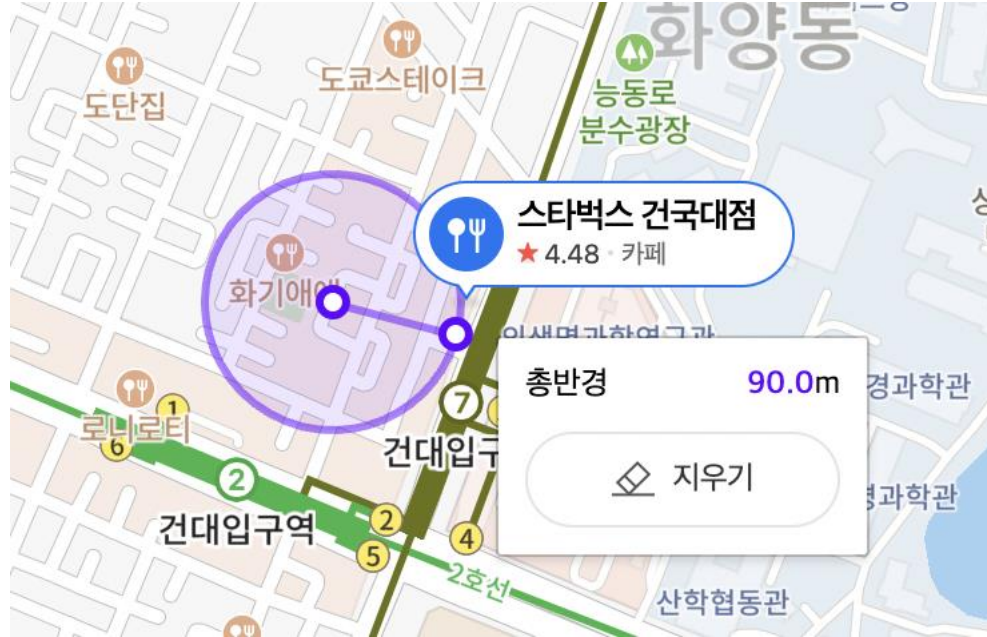
```

- res(result): 개별 모델들이 예측한 값들을 하나의 Dataframe으로 묶음
- res["D"]와 res["idx"]를 나올 수 없는 임의의 값으로 설정
- D : distance
- for s in S: S에 있는 좌표 set을 불러와 s에 설정하는 반복문(3번)
- $\text{temp} = \{ (\text{스타벅스 } x\text{값} - \text{result } x\text{값})^2 + (\text{스타벅스 } y\text{값} - \text{result } y\text{값})^2 \}^{(1/2)}$   
→ 거리공식
- 조건문을 통해 temp에 최솟값이 남도록 함
- res["idx"][i] = S.index(s) : 최소 거리인 스타벅스 지점의 index

# 결과(1차)

1 `res.sort_values("D")`

	x	y	D	idx
19	988.797017	424.375729	43	1
17	988.797017	407.182678	45	1
22	988.460000	407.182678	45	1
18	988.797017	446.660000	50	1
33	988.797017	386.712531	54	1
34	974.250000	419.390000	57	1
14	973.905823	427.780958	58	1
32	973.905823	407.182678	59	1
13	973.905823	439.830000	61	1
23	977.260000	386.712531	64	1




- 예측 x좌표 : 988
- 예측 y좌표 : 424
- x좌표 모델 : LightGBM
- y좌표 모델 : RandomForest
- 오차: 90m

# 그러나...

```
1 X = df.drop(["s_longitude", "s_latitude"], axis = 1)
2 y_1 = df["s_longitude"]
3 y_2 = df["s_latitude"]
4
5 test = pd.DataFrame({"e_longitude" : [922], "e_latitude" : [463], "d_longitude" : [911], "d_latitude" : [460], "o_longitude" : [1002], "o_latitude" : [470]})
6 S = [[1040, 602], [1032, 420], [1120, 594]]
```

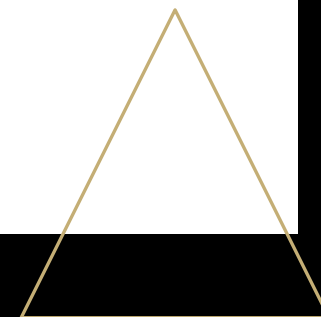
- 새로운 방식 적용  
→ X좌표와 Y좌표 간에 어떠한 관계가 있지 않을까?

feature의 위도, 경도를 나눠 설정하지 않고 한꺼번에 X에 설정함.  
그 외는 동일.



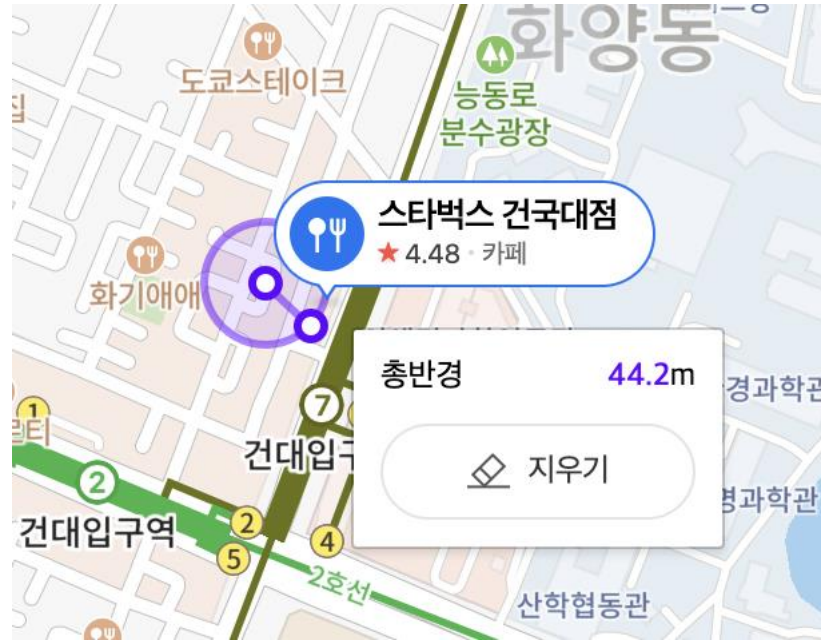
```
1 Model = [LinearRegression(), svm.SVR(), XGBRegressor(), LGBMRegressor(), RandomForestRegressor(), GradientBoostingRegressor()]
2 temp1, temp2 = [], []
3 for i in range(len(Model)):
4     for j in range(len(Model)):
5         if i == j:
6             continue
7         else:
8             model1, model2 = Model[i], Model[j]
9             model1.fit(X, y_1)
10            model2.fit(X, y_2)
11            temp1.append(model1.predict(test)[0])
12            temp2.append(model2.predict(test)[0])
```

- Model에 fit할 때도 X\_1, X\_2 대신에 X 사용



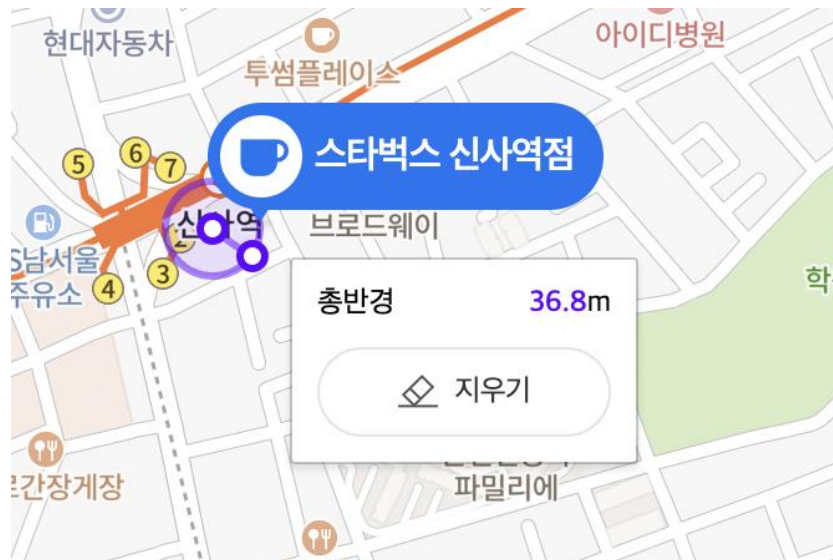
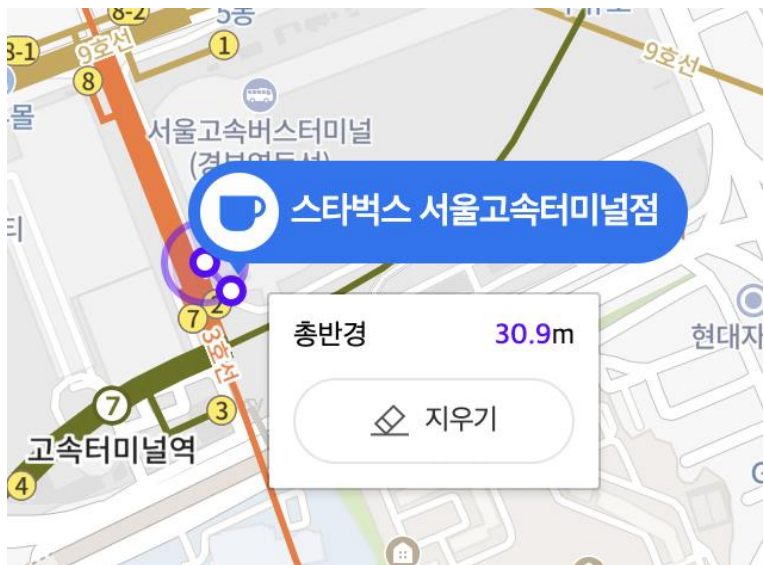
# 결과(2차)

	x	y	D	idx
17	1021.811683	413.394684	12	1
19	1021.811683	428.328064	13	1
22	987.100000	413.394684	45	1
14	986.629150	428.328064	46	1
23	987.250000	403.734848	47	1
12	986.629150	403.734848	48	1
18	1021.811683	473.130000	54	1
13	986.629150	452.960000	56	1
24	974.540000	428.328064	58	1
4	967.107005	428.328064	65	1

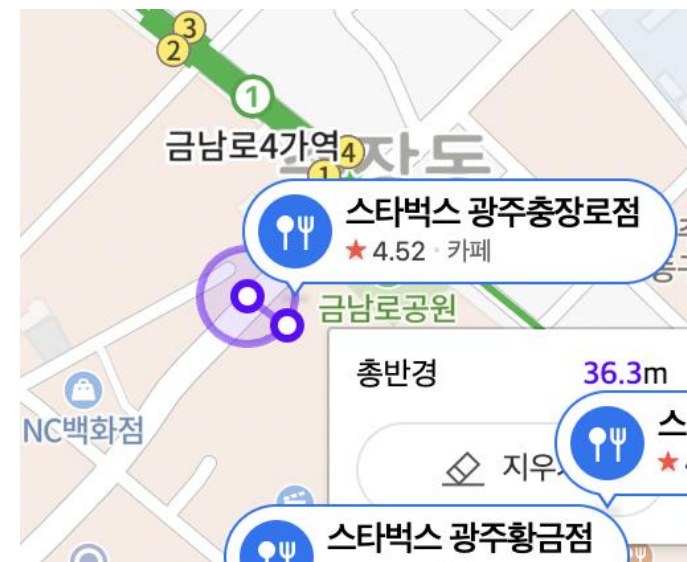
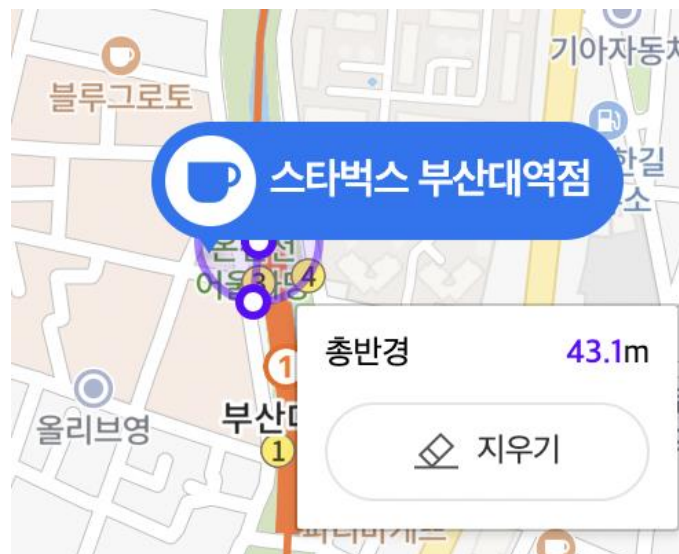
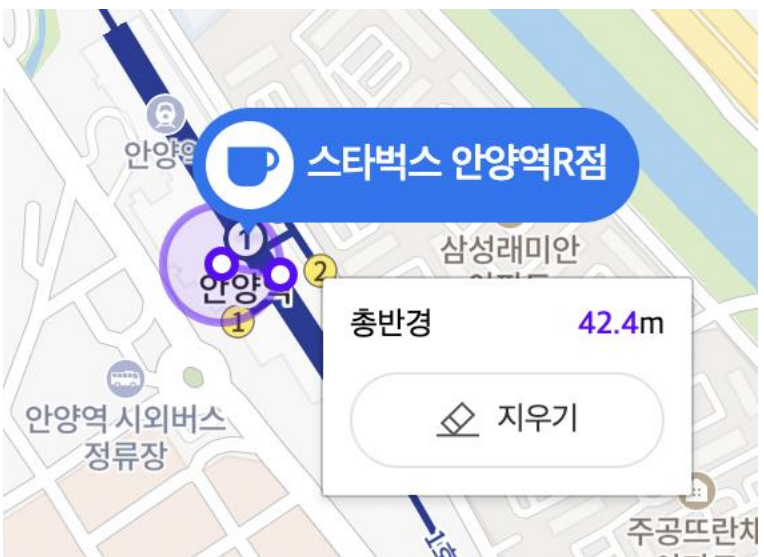


- 예측 x좌표 : 1021
- 예측 y좌표 : 413
- x좌표 모델 : LightGBM
- y좌표 모델 : SVM
- 오차: 44.2m





- 다른 지역에서도 높은 정확도를 보임



# 한계와 의의


## <한계>

1. 역 근처 한정
2. Feature 선택에 아쉬움
3. 결과의 성능을 평가할 상대지표의 부재

## <의의>

1. 일반적인 숫자를 다루는 것이 아니라 좌표를 다루었다는 점에서 창의적이었음
2. 꽤 높은 정확도





감사합니다

