

PINEAT

서울시 공공자전거 이동 경로를 활용한
푸드트럭 입지선정

INDEX

- 푸드트럭 운영 지원 현황 문제점
- PINEAT의 목표(서비스 개요)
- PINEAT 모델
- 결론 및 한계점

TEAM



박민준
삼육대학교
컴퓨터공학부

박현우
세종대학교
전자정보통신공학과

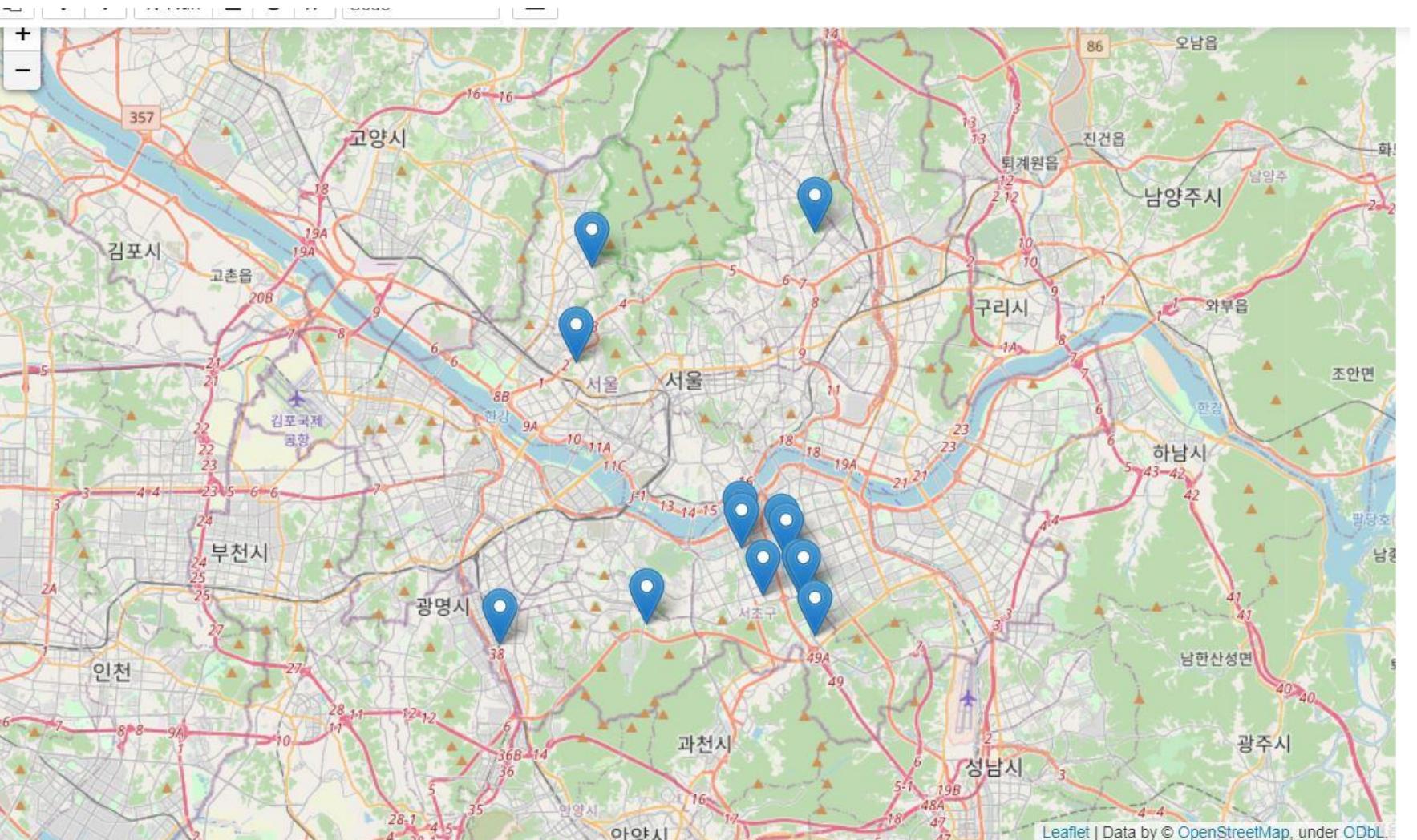
곽석우
세종대학교
기계공학과

박상현
세종대학교
호텔경영학과

양형석
서울시립대학교
전자전기컴퓨터공학부

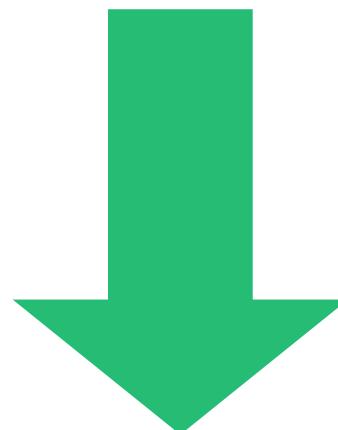
푸드트럭 운영 지원 현황과 문제점

푸드트럭 현황 및 입지조건 분석 목적



기존 푸드트럭 영업 허가지

총 13곳



527개의 서울시 푸드트럭 수에 비해
영업 가능 장소가 현저하게 부족하다

푸드트럭 현황 및 입지조건 분석 목적

숨어서 불법 영업..푸드트럭 기부 취지
'무색' - R

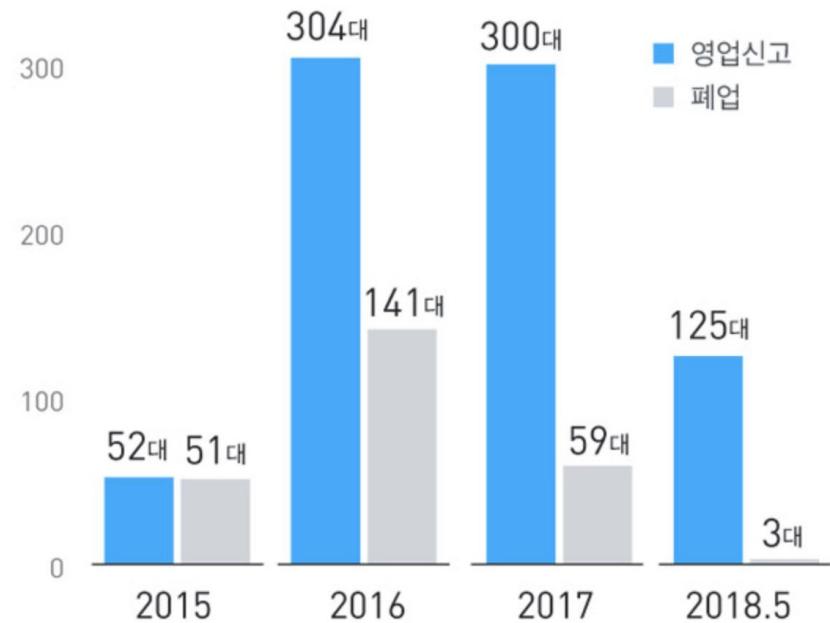
강서영 2020년 01월 30일 20시 30분 00초
글자 크기 + - 인쇄

뉴스홈 | 최신기사

'푸드트럭' 월 수익 176만원···장소·경험 따라 10배 차이

영업 허가지 부족으로
발생하고 있는 문제점

서울시 푸드트럭
영업신고 및 폐업 현황 (단위: 대)



자료 서울연구원 인포그래픽 강준희

KBS

서울시 푸드트럭 신규영업장소 발굴 협조 요청

서울시 청년·취약계층의 일자리 창출 등을
위한 푸드트럭 영업 활성화 대책으로

푸드트럭의 수익성 있는
신규 영업 공간 발굴 사업을 위해

푸드트럭 입지를 선정하는 프로젝트 기획

코로나19바이러스감염병 예방 및 확산방지에 적극 동참합시다!



서울특별시



수신 수신자참조

(경유)

제목 푸드트럭 신규영업장소(상시영업지) 발굴 협조요청

- I. 우리 시에서는 청년·취약계층의 일자리 창출 등을 위해 푸드트럭 영업 활성화 대책을 수립·추진 중에 있으며, 푸드트럭의 수익성 있는 신규 영업 공간 발굴을 위해 노력하고 있습니다.
2. 이에, 서울시내의 푸드트럭 영업 활성화를 위해 시 전 부서 및 자치구의 적극적인 참여와 협조를 요청드리며, 아래와 같이 상시 영업이 가능한 푸드트럭 신규 영업장소를 기한 내 발굴하여 제출해주시기 바랍니다.

- 아 래 -

○ 요청사항 : 수익성 있는 푸드트럭 상시영업장소 발굴

* 청사 앞, 구민회관, 지하철 역 인근, 도시공원, 전통시장 등 각 부서 및 자치구 기존 관리시설·신규시설 중 푸드트럭 영업가능 공간 적극 발굴 요청(해당 장소 재산관리부서와 인근상권과 중복 여부 등 협의·검토 후 제출)

* 푸드트럭 영업허용 장소 : 식품위생법 시행규칙 제42조제1항제14호 [별표 15의2], 서울특별시 음식판매자동차 영업장소 지정 및 관리 등에 관한 조례 제2조

○ 제출서식

담당부서 (재산관리부서)	운영장소	푸드트럭 운영(예정)대수	운영가능시간	비고

* 위치도 및 사진 : 불임파일에 작성하여 제출(필수)

○ 제출기한 : 8.21.(금)까지

3. 아울러, 제출한 신규영업장소에 대해서는 추후 현장실사단의 현장 검증 등을 거쳐 영업장소 지정 여부를 최종 결정할 것이며, 필요시 소상공인정책담당관의 통합공모를 통해 영업장을 선정할 계획임을 알려드립니다.

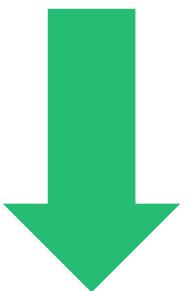
PinEat의 목표

푸드트럭 입지 선정을 위한 방법 모색

신규 입지 선정 배경

유동인구와 매출액의

유의미한 상관관계



푸드트럭 입지로 선정의 필요조건으로

유동인구가 많은 곳을 설정

국내 100대 상권의 유동인구와 매출액과의 상관관계 확인

한양대학교 사이버대학교 외식경영학 교수 김영갑

2012년 3월 18일 매일경제신문과 SK텔레콤 ICT 사업팀이 공동으로 국내 100대 상권을 조사한 자료가 있습니다. 이 자료에는 상권의 년매출액과 하루 유동인구가 포함되어 있더군요. 이 자료를 이용하면 유동인구와 매출액과의 상관관계를 쉽게 확인 할 수 있겠다는 생각을 했습니다.

실제로 국내 100대 상권의 매출액과 일일유동인구간의 상관관계를 구한 결과 상관계수는 0.01 수준에서 유의한 것으로 나타났습니다. 계수의 값은 +0.3996 이었습니다.

상관관계 계수는 -1부터 1까지의 값을 가질 수 있습니다. +0.4 정도라면 어느 정도 양의 상관관계가 있다고 할 수 있지만 생각만큼 매우 높다고 볼 수는 없습니다.

보통 상관계수가 0.7 이상이면 매우 강한 관계를 0.4~0.7 이면 상당한 관계로 볼 수 있으며, 0.2~0.4정도는 약한 관계로 봅니다. 따라서 매출액과 유동인구와의 관계는 약한 관계와 상당한 관계의 사이에 위치하는 것으로 볼 수 있습니다. 따라서 보통의 상관관계가 있다고 보시면 됩니다.

유동인구가 늘어나면 상권내 점포들의 매출액이 증가하는 것은 당연합니다. 만약 유동인구를 독립변수로 하고 매출액을 종속변수로 하는 회귀분석을 해 보면 좀 더 명쾌한 답이 나옵니다. 회귀분석 결과 유동인구의 매출액에 대한 **설명력은 약 15%(매출액의 총 변화량 중 15%가 유동인구에 의해 설명된다고 해석할 수 있다)**였으며, **유동인구가 100% 증가한다면 매출액은 약 40% 증가한다는 결과가 나옵니다.**

그럼에도 불구하고 매출액의 추정을 유동인구만으로 하기에는 다소 한계가 있다는 해석도 가능합니다. 특히 이 부분은 업종에 따라서 많은 차이가 있을 것 같습니다. 예를 들어 편의점과 같은 점포는 유동인구의 증가가 매출에게 더 큰 영향을 미칠 것입니다. 음식점 중에서 패스트푸드점의 경우도 그럴 것 같군요. 하지만 높은 수준의 서비스와 상품의 품질을 요하는 점포의 경우는 유동인구의 증가가 매출에 미치는 영향이 작아집니다.

코로나19로 변화한 서울 시민의 일상

지하철 대신 '파랑이'로 출퇴근… 이용률 67% 증가

교육 ▼

고강도 사회적 거리두기 이후… "출퇴근 따릉이' 이용 1.8배 늘었다"

서울시민, 코로나19로 대중교통보다 '따릉이' 더 이용

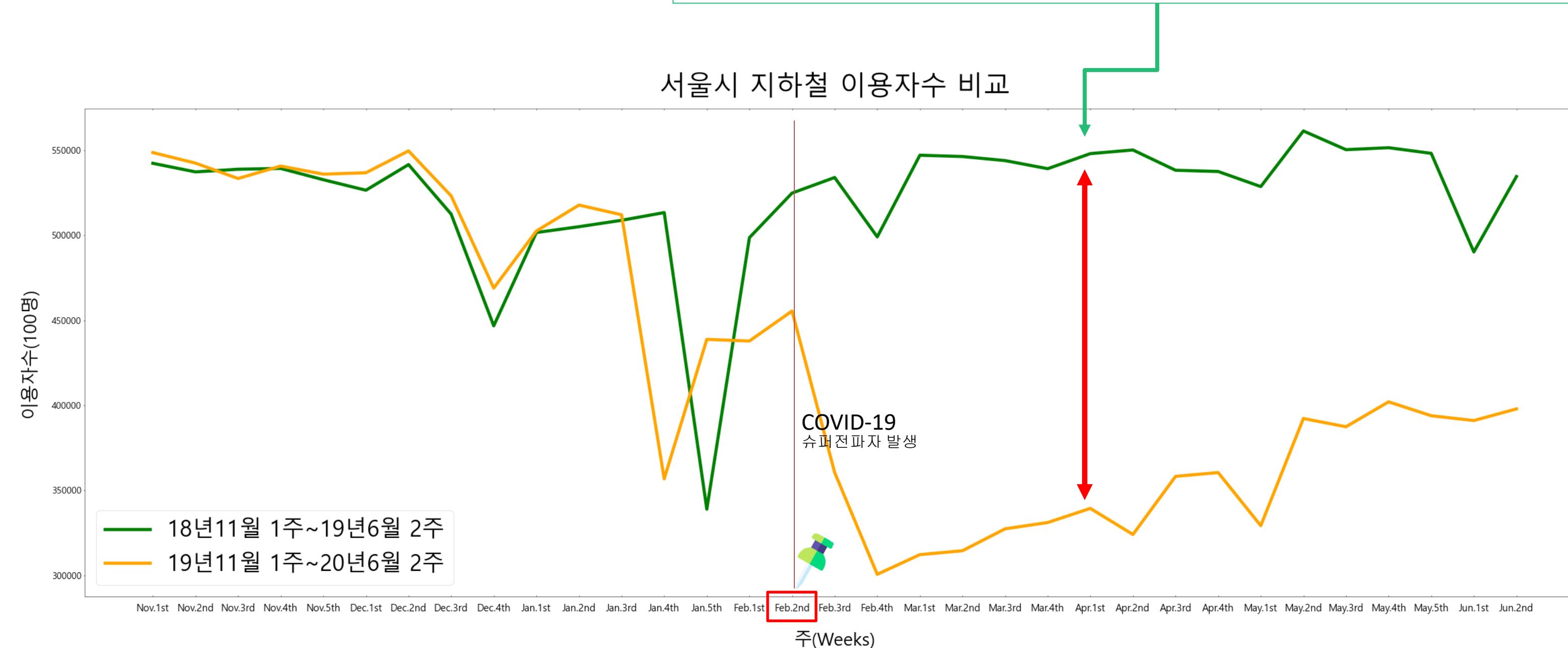
[아무튼, 주말] 코로나야 따라올 테면 따라와봐… “오늘부터
자전거로 출근합니다”

코로나 시대 자전거 열풍

COVID-19로 변화한
서울 시민들의
대중 교통 생활

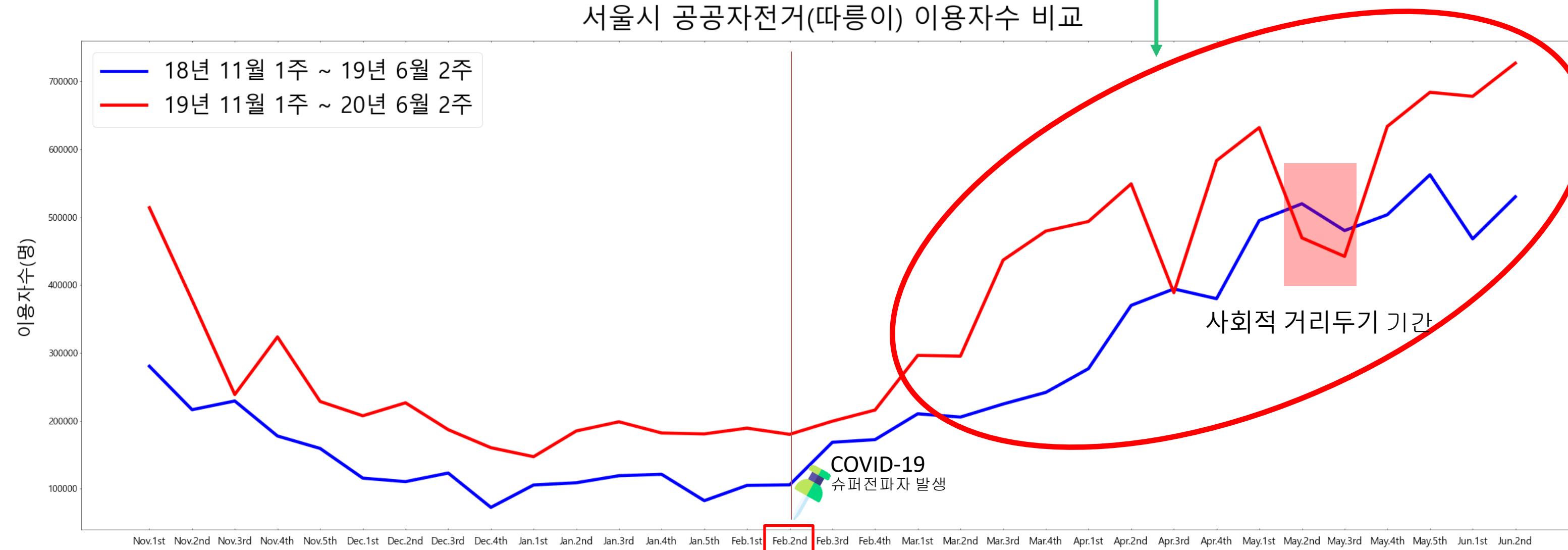
지하철 선호도 급감

2018년 11월부터 2019년 6월까지 서울 지하철 이용자 수와
2019년 11월부터 2019년 6월까지 서울 지하철 이용자 수를 비교하여
같은 기간 동안 이용자 수가 얼마나 변화했는지 보여준다.



따릉이 선호도 급증

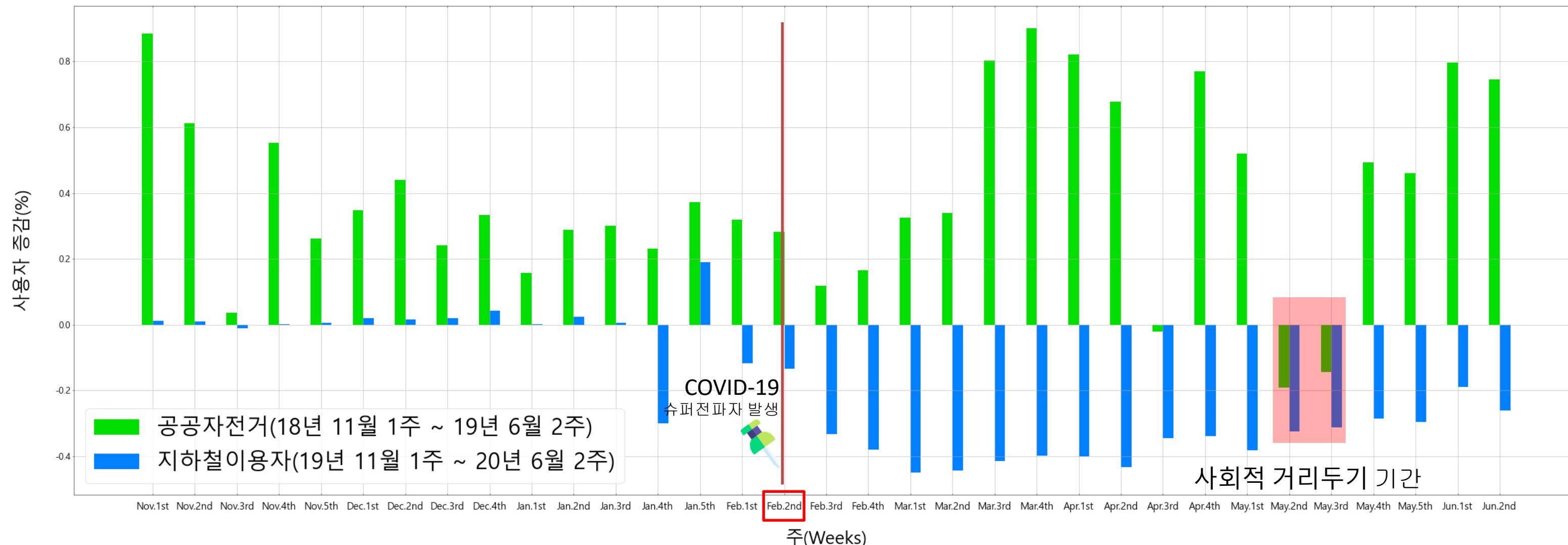
2018년 11월부터 2019년 6월까지 서울 공공자전거 이용자 수와
2019년 11월부터 2020년 6월까지 서울 공공자전거 이용자 수를 비교하여
같은 기간 동안 이용자 수가 얼마나 변화했는지 보여준다.



따릉이/지하철 이용자 증감비율

같은 기간 동안의
서울 지하철 이용자 수의 증감률과
서울 공공자전거 이용자 수의 증감률을 통해
COVID-19 슈퍼 전파자 이후
지하철과 따릉이 상반된 선호 정도를 알 수 있다.

주차별 서울시 공공자전거/지하철 이용자 증감비율



대중교통 선호도 변화

Covid-19 이후 사람들의 대중교통 선호도 변화로 지하철 이용객이 급감하고 따릉이 이용자가 급증해 새로운 대중교통 환경이 조성되었다.

지하철 이용자 수 변화

사회적 거리두기
밀폐, 밀집 공간 기피

이용자 수 감소

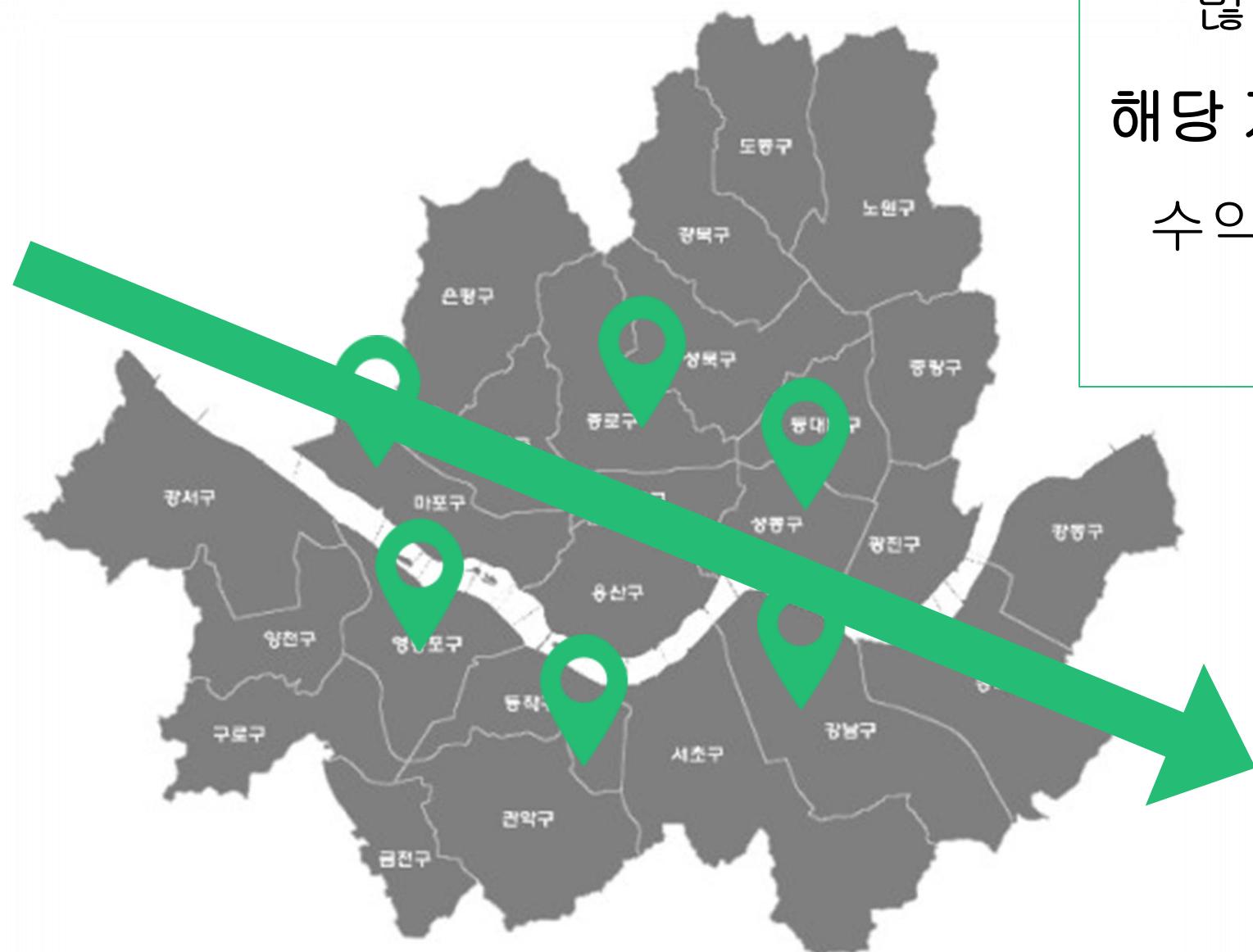
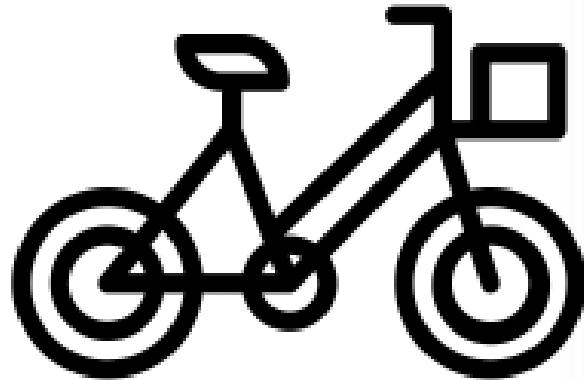
따릉이 이용자 수 변화

생활 속 거리두기 습관화
개방 공간 선호도 증가

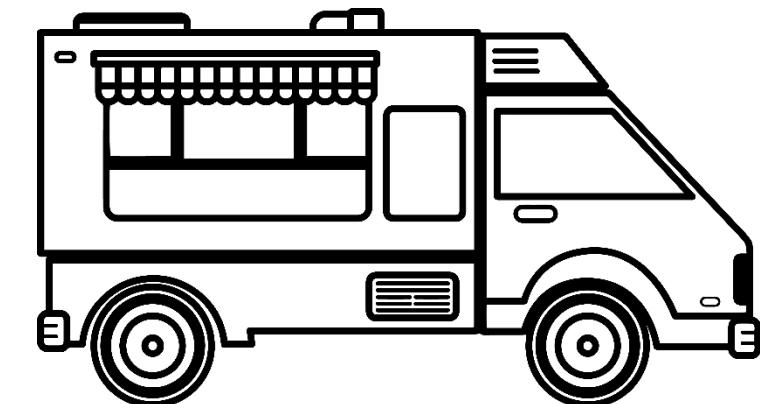
이용자 수 급증



따릉이 이동 경로를 활용한 입지선정



서울시 공공자전거가
많이 이동한 경로를 추적해
해당 지점에 푸드트럭을 위치시켜
수익성 있는 장사를 할 수 있는
공간을 모색한다.



PINEAT

자전거가 이동하는 최적 경로를 활용해

최다 누적 경로를 확인하고

최다 누적 위치에 푸드트럭 입지를 선정한 후

해당 입지 상권의 예상 매출과 업종을 추천

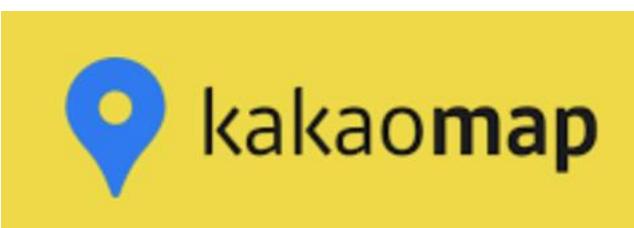
PINEAT 모델

따릉이 최다 빈도 경로 분석에 따른 입지선정

네비게이션 제작 과정



+



위도/경도 데이터 활용

OpenStreetMap로부터 얻은 데이터프레임 형식의 데이터에
Google Maps API의 Elevation 데이터를 추가한다.
위경도 데이터를 활용하여 역학적 에너지 공식을 바탕으로
Edge에 역학적 에너지 소비량이라는 새로운 열을 추가한다.
이것을 가중치로 두어 역학적 에너지 소모가 낮은 길로 안내한다.

Edge +
(선)

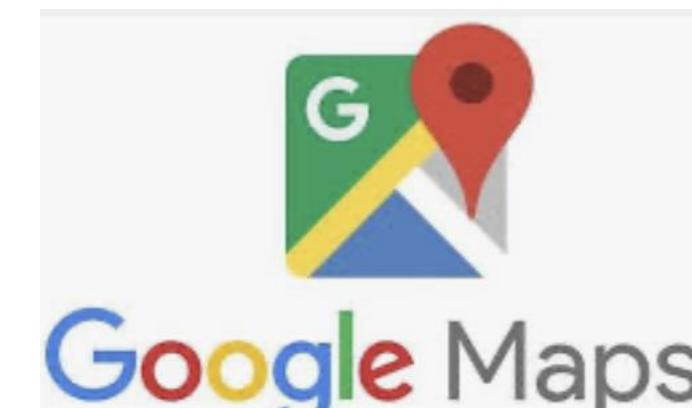
역학적 에너지 소비량

$$\begin{aligned} & \mu m_{bike} g * \cos(\arctan(grade)) * length \\ & + m_{bike} g * length * \sin(\arctan(grade)) \\ & + \frac{1}{2} \rho A C_d (u * \cos(\arctan(grade)))^2 \end{aligned}$$

$$\begin{aligned} \mu &= 0.015 & \text{마찰로 인해} \\ g &= 9.81 & \text{소비된 에너지량} \\ \rho &= 1.225 & \text{중력으로 인해} \\ A &= 0.441 * 0.370 & \text{소비된 에너지량} \\ m_{bike} &= 18 & \text{공기저항으로 인해} \\ u &= 4.53 & \text{소비된 에너지량} \\ length &: \text{도로 길이} & \\ grade &: \text{도로 경사} & \end{aligned}$$

국가참조표준센터
한국인 인체 치수
성인 남성 만
25~29세
평균 상체 면적

Node +
(모서리)



Elevation(고도) 활용

OSMNX와 개발에 필요한 모듈 import

```
import numpy as np
import osmnx as ox, networkx as nx, geopandas as gpd, matplotlib.pyplot as plt
import networkx as nx
import seaborn as sns
import folium
import folium.plugins
import pandas as pd
from folium import plugins
from folium.plugins import MarkerCluster
import branca.colormap as cm
import json
import urllib.request
```

도로망 내려받는 과정

OSMNX의 NetworkX 기능을 활용해
OpenStreetMap의 도로망을 가져옴

```
Seoul_Map = ox.graph_from_place('서울특별시', network_type = 'bike')
a = ox.elevation.add_node_elevations(Seoul_Map, 'AIzaSyCPwXwcoODP8gJdEKcfNka4AtiIEFgVWP8', max_locations_per_batch=350
                                     , pause_duration=0.02)
b = ox.elevation.add_edge_grades(Seoul_Map, add_absolute=True)
Seoul_Node, Seoul_Edge = ox.graph_to_gdfs(Seoul_Map)

Seoul_Edge['energy'] = 0

#에너지 공식을 이용하여 새로운 열을 만들었고, 이것을 네비게이션 기능에 사용할 계획입니다.
for i in range(len(Seoul_Edge)):
    Seoul_Edge['energy'][i] = 0.015*9.81*18*math.cos(math.atan(Seoul_Edge['grade'][i]))*Seoul_Edge['length'][i]
    + 18*9.81*Seoul_Edge['length'][i]*math.sin(math.atan(Seoul_Edge['grade'][i])) + 0.5*1*1.225*(0.441*0.370)
    *(4.53*math.cos(math.atan(Seoul_Edge['grade'][i])))**2
```

서울특별시에서 자전거로 갈 수 있는
모든 도로망의 정보를 받아옵니다.

각 노드들에 elevation을 추가합니다.

각 엣지들에 grade를 추가합니다.

Grade 추가

26만개 Edge에 Grade 정보 추가

	osmid	name	highway	oneway	length	bridge	geometry	width	tunnel	ref	lanes	maxspeed	service	access	junction
0	[26965306, 26965302, 406000767]	[용봉 교, 고 산자 로]	secondary	False	753.981	yes	LINESTRING (127.03458 37.55393, 127.03476 37.5...)			NaN	NaN	NaN	NaN	NaN	NaN
1	218655250	고산 자로	secondary	False	74.555	NaN	LINESTRING (127.03458 37.55393, 127.03445 37.5...)			NaN	NaN	NaN	NaN	NaN	NaN
2	226905209	고산 자로	secondary	True	197.494	NaN	LINESTRING (127.03458 37.55393, 127.03449 37.5...)			NaN	NaN	NaN	NaN	NaN	NaN
3	26271799	증랑 천자 전거 길	cycleway	False	726.665	NaN	LINESTRING (127.05140 37.55611, 127.05138 37.5...)	3		NaN	NaN	NaN	NaN	NaN	NaN
4	238163912	청계 천자 전거 길	cycleway	False	601.101	NaN	LINESTRING (127.05140 37.55611, 127.05139 37.5...)	3		NaN	NaN	NaN	NaN	NaN	NaN
...
15463	844577493	NaN	unclassified	False	52.670	NaN	LINESTRING (127.10180 37.54827, 127.10184			NaN	NaN	NaN	NaN	NaN	NaN

	osmid	name	highway	oneway	length	bridge	geometry	grade	grade_abs	width	tunnel	ref	lanes	maxspeed	service
0	[26965306, 26965302, 406000767]	[용봉 교, 고 산자 로]	secondary	False	753.981	yes	LINESTRING (127.03458 37.55393, 127.03476 37.5...)	-0.008	0.008	NaN	NaN	NaN	NaN	NaN	NaN
1	218655250	고산 자로	secondary	False	74.555	NaN	LINESTRING (127.03458 37.55393, 127.03445 37.5...)	0.014	0.014	NaN	NaN	NaN	NaN	NaN	NaN
2	226905209	고산 자로	secondary	True	197.494	NaN	LINESTRING (127.03458 37.55393, 127.03449 37.5...)	-0.048	0.048	NaN	NaN	NaN	NaN	NaN	NaN
3	26271799	증랑 천자 전거 길	cycleway	False	726.665	NaN	LINESTRING (127.05140 37.55611, 127.05138 37.5...)	0.002	0.002	3	NaN	NaN	NaN	NaN	NaN
4	238163912	청계 천자 전거 길	cycleway	False	601.101	NaN	LINESTRING (127.05140 37.55611, 127.05139 37.5...)	0.003	0.003	3	NaN	NaN	NaN	NaN	NaN
...
15463	844577493	NaN	unclassified	False	52.670	NaN	LINESTRING (127.10180 37.54827, 127.10184	0.002	0.002	NaN	NaN	NaN	NaN	NaN	NaN



Elevation 추가

10만개 Node에 Elevation 정보 추가

Seoul_Node

	y	x	osmid	highway	ref	geometry
287287152	37.553925	127.034582	287287152	NaN	NaN	POINT (127.03458 37.55393)
287298869	37.556112	127.051404	287298869	NaN	NaN	POINT (127.05140 37.55611)
287298930	37.550722	127.035954	287298930	NaN	NaN	POINT (127.03595 37.55072)
287719331	37.568552	127.046246	287719331	NaN	NaN	POINT (127.04625 37.56855)
291714636	37.541511	127.019567	291714636	NaN	NaN	POINT (127.01957 37.54151)
...
7878704815	37.548116	127.102003	7878704815	NaN	NaN	POINT (127.10200 37.54812)
7878704817	37.548536	127.101644	7878704817	NaN	NaN	POINT (127.10164 37.54854)
7878704820	37.548267	127.101802	7878704820	NaN	NaN	POINT (127.10180 37.54827)
7899814964	37.546147	127.107480	7899814964	NaN	NaN	POINT (127.10748 37.54615)
7899814969	37.545449	127.108269	7899814969	NaN	NaN	POINT (127.10827 37.54545)



```
a = ox.elevation.add_node_elevations(Seoul_Map, 'AlzaSyCPwXwco0DP8gJdEKcfNKA4AtiEFgVWP8', max_locations_per_batch=350  
                                     , pause_duration=0.02)  
b = ox.elevation.add_edge_grades(Seoul_Map, add_absolute=True)  
Seoul_Node, Seoul_Edge = ox.graph_to_gdfs(Seoul_Map)
```

Seoul_Node

	y	x	osmid	elevation	highway	ref	geometry
287287152	37.553925	127.034582	287287152	23.814	NaN	NaN	POINT (127.03458 37.55393)
287298869	37.556112	127.051404	287298869	8.115	NaN	NaN	POINT (127.05140 37.55611)
287298930	37.550722	127.035954	287298930	7.313	NaN	NaN	POINT (127.03595 37.55072)
287719331	37.568552	127.046246	287719331	11.311	NaN	NaN	POINT (127.04625 37.56855)
291714636	37.541511	127.019567	291714636	14.239	NaN	NaN	POINT (127.01957 37.54151)
...
7878704815	37.548116	127.102003	7878704815	25.260	NaN	NaN	POINT (127.10200 37.54812)
7878704817	37.548536	127.101644	7878704817	25.793	NaN	NaN	POINT (127.10164 37.54854)
7878704820	37.548267	127.101802	7878704820	25.171	NaN	NaN	POINT (127.10180 37.54827)
7899814964	37.546147	127.107480	7899814964	19.030	NaN	NaN	POINT (127.10748 37.54615)
7899814969	37.545449	127.108269	7899814969	17.096	NaN	NaN	POINT (127.10827 37.54545)

Kakao MAP API 활용

```
def getLatLng(addr): #카카오 API를 이용하여 주소를 위도 경도를 반환합니다.  
    try:  
        url = 'https://dapi.kakao.com/v2/local/search/keyword.json?query=' + addr  
        headers = {"Authorization": "KakaoAK 84bab228f2f7a5f35ca89b1c459849ec"}  
        result = json.loads(str(requests.get(url, headers=headers).text))  
        x = float(result['documents'][0]['x']) # 경도 - x축 기준  
        y = float(result['documents'][0]['y']) # 위도 - y축 기준  
        return (y,x)  
  
    except:  
        return -1 #위도 경도 반환 실패인 경우 -1을 반환합니다.
```

Kakao MAP API를 활용한
위/경도 정보 반환

위/경도 반환 및 가까운 Node 찾기

```
: start = getLatLng('세종대학교')
arrive = getLatLng('서울역')
print(start)
print(arrive)
```

(37.551920581618205, 127.0735094176659)
(37.5546788388674, 126.970606917394)

출발지에 '세종대학교',
도착지에 '서울역'을 검색하여
위/경도 반환

```
: orig_node = ox.get_nearest_node(Seoul_Map, (start[0], start[1])) #출발지
dest_node = ox.get_nearest_node(Seoul_Map, (arrive[0], arrive[1])) #도착지
```

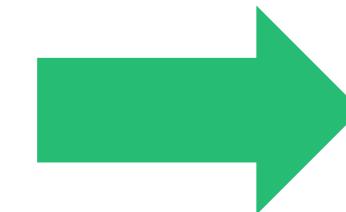
```
: print(orig_node)
print(dest_node)
```

get_nearest_node 함수를 활용하여
입력된 출발지와 도착지에서
가장 가까운 Node를 찾음

1168013947
3391061977

네비게이션 구현

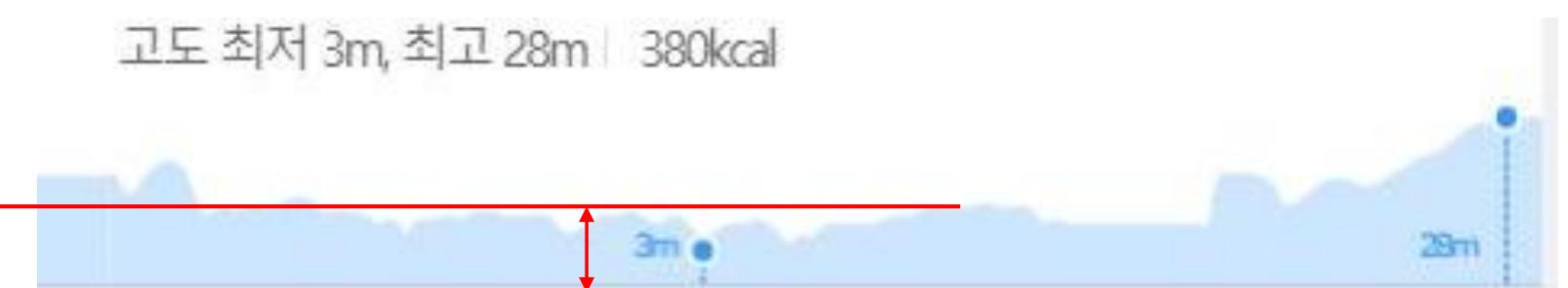
왼쪽의 경우 다른 조건을 무시하고 단순히 거리만을 가중치로 둔 경우이며
오른쪽의 경우 역학적 에너지를 가중치로 둔 경우이다.



고도 최저 11m, 최고 46m | 296kcal



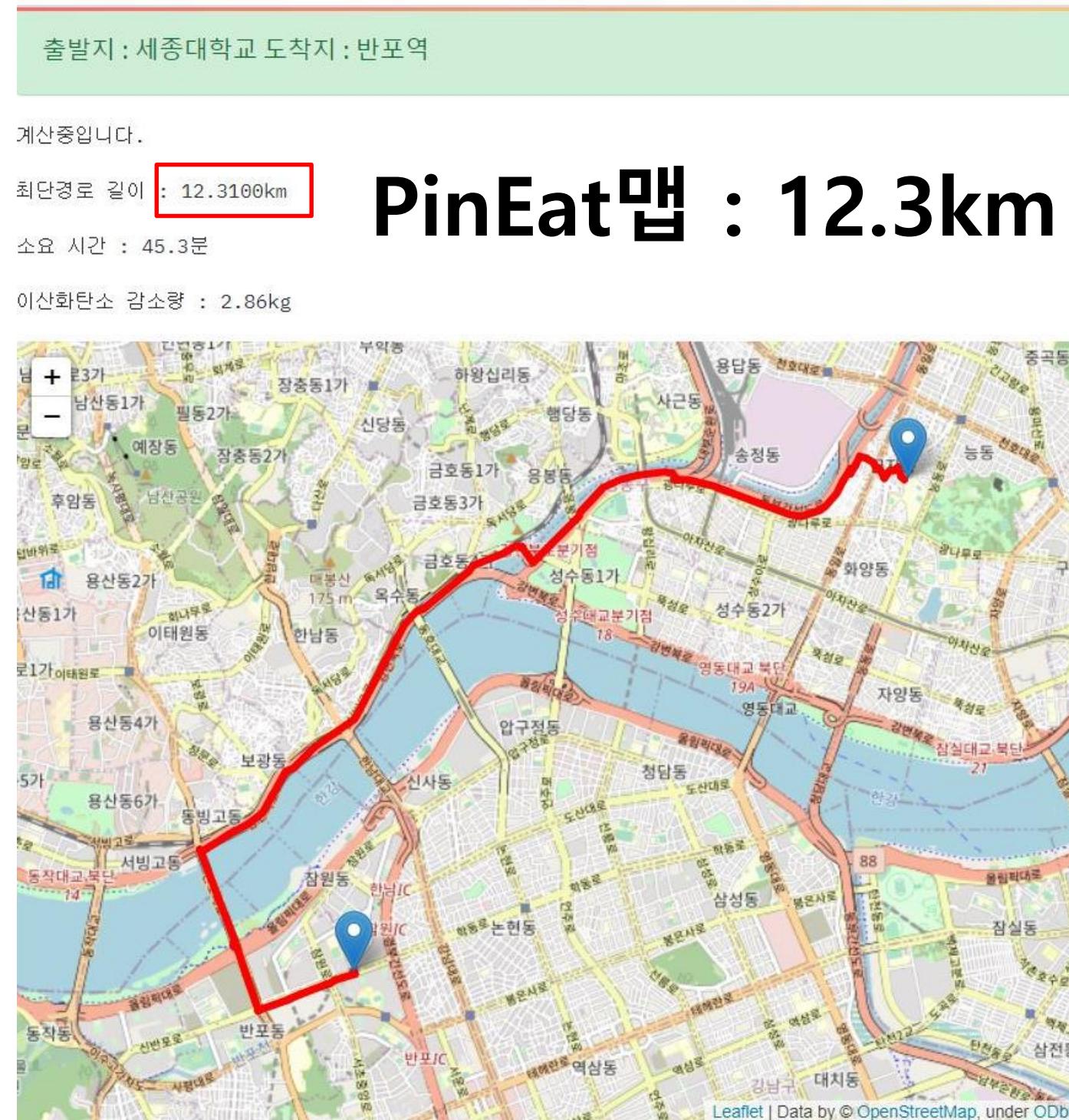
고도 최저 3m, 최고 28m | 380kcal



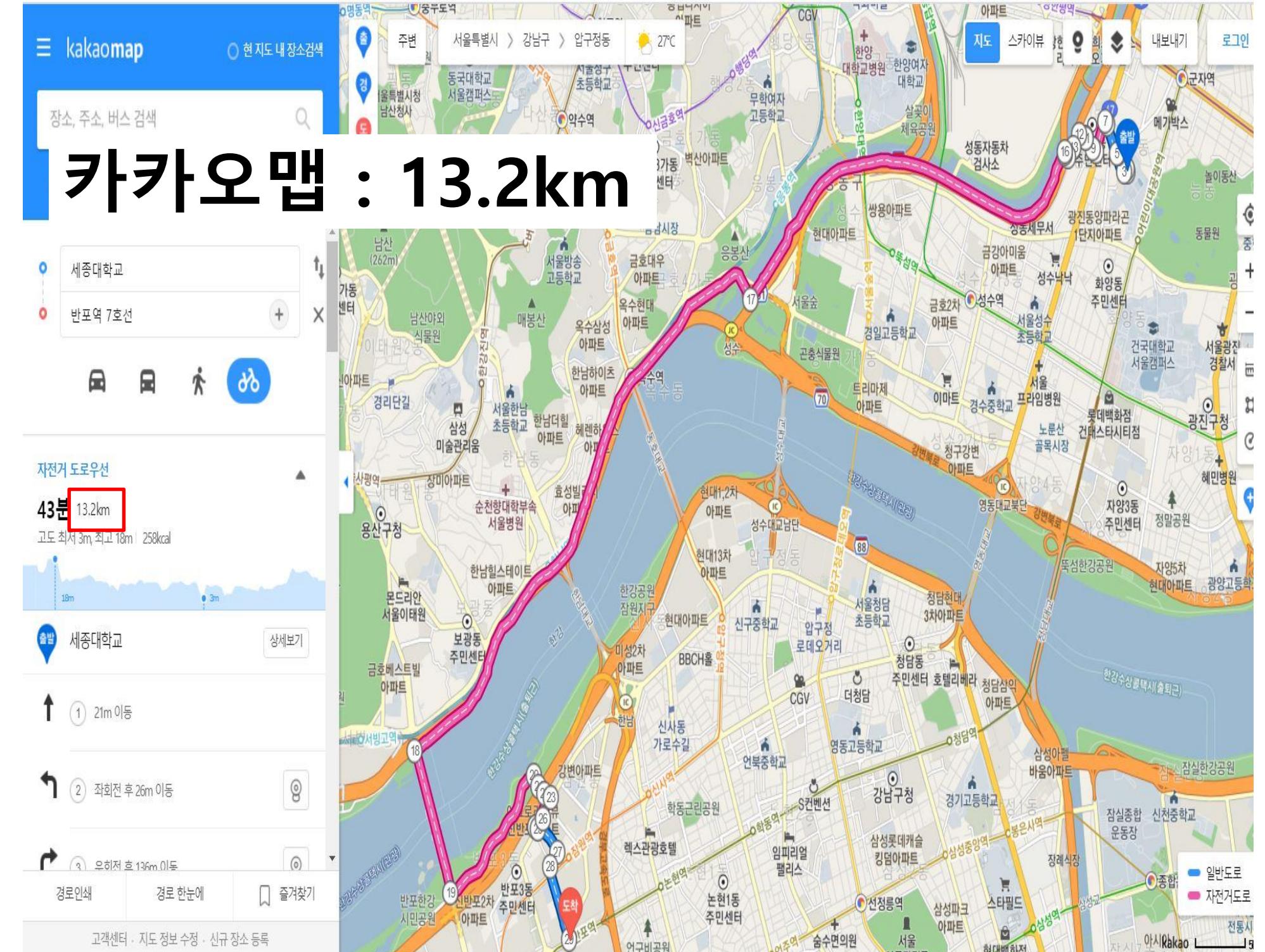
경사도 차이로
에너지 소모량 차이 발생

네비게이션 정확도 측정(세종대학교 -> 반포역)

카카오맵 대비 PinEat맵
정확도 : 93%

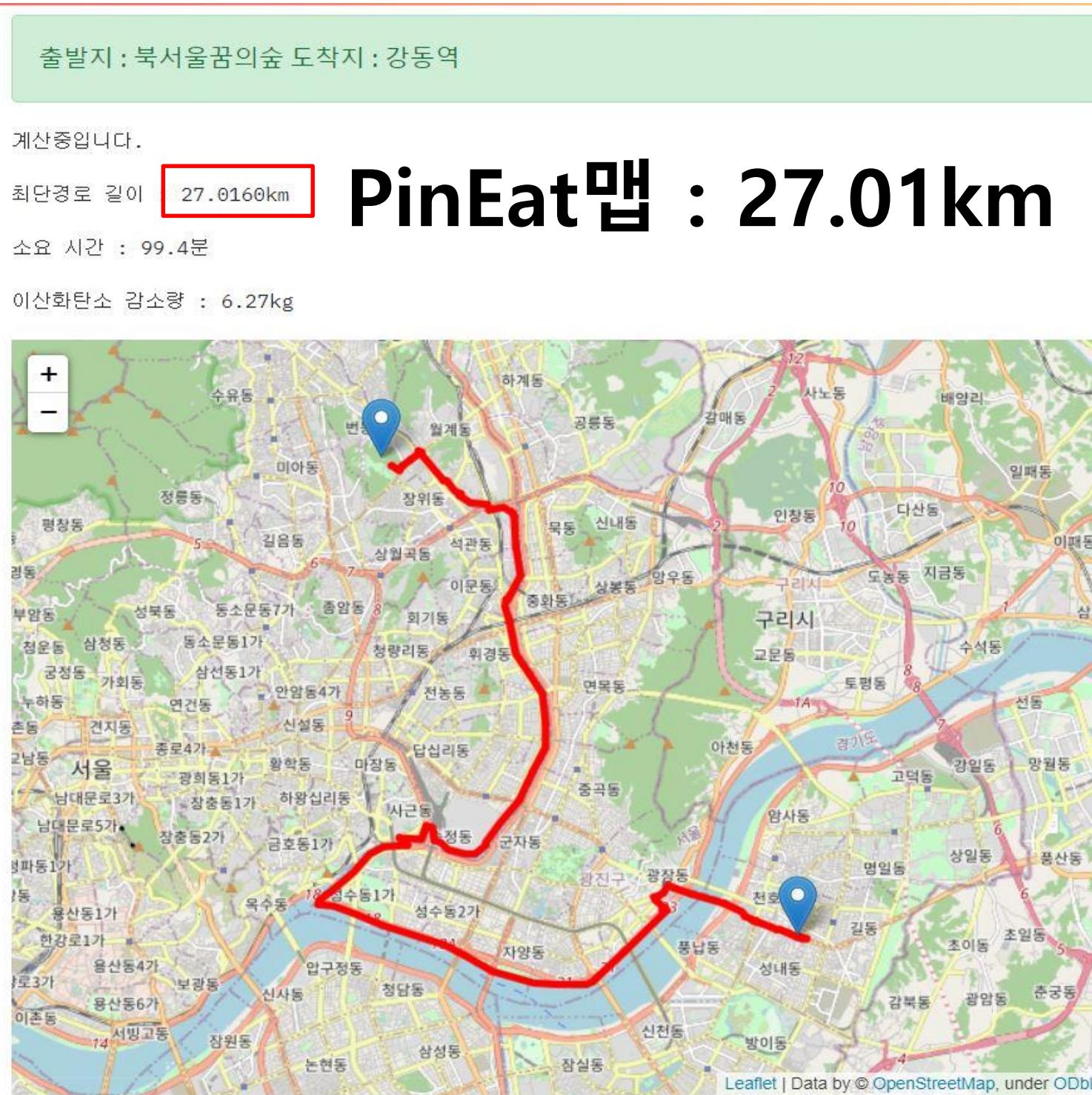


PinEat맵 : 12.3km

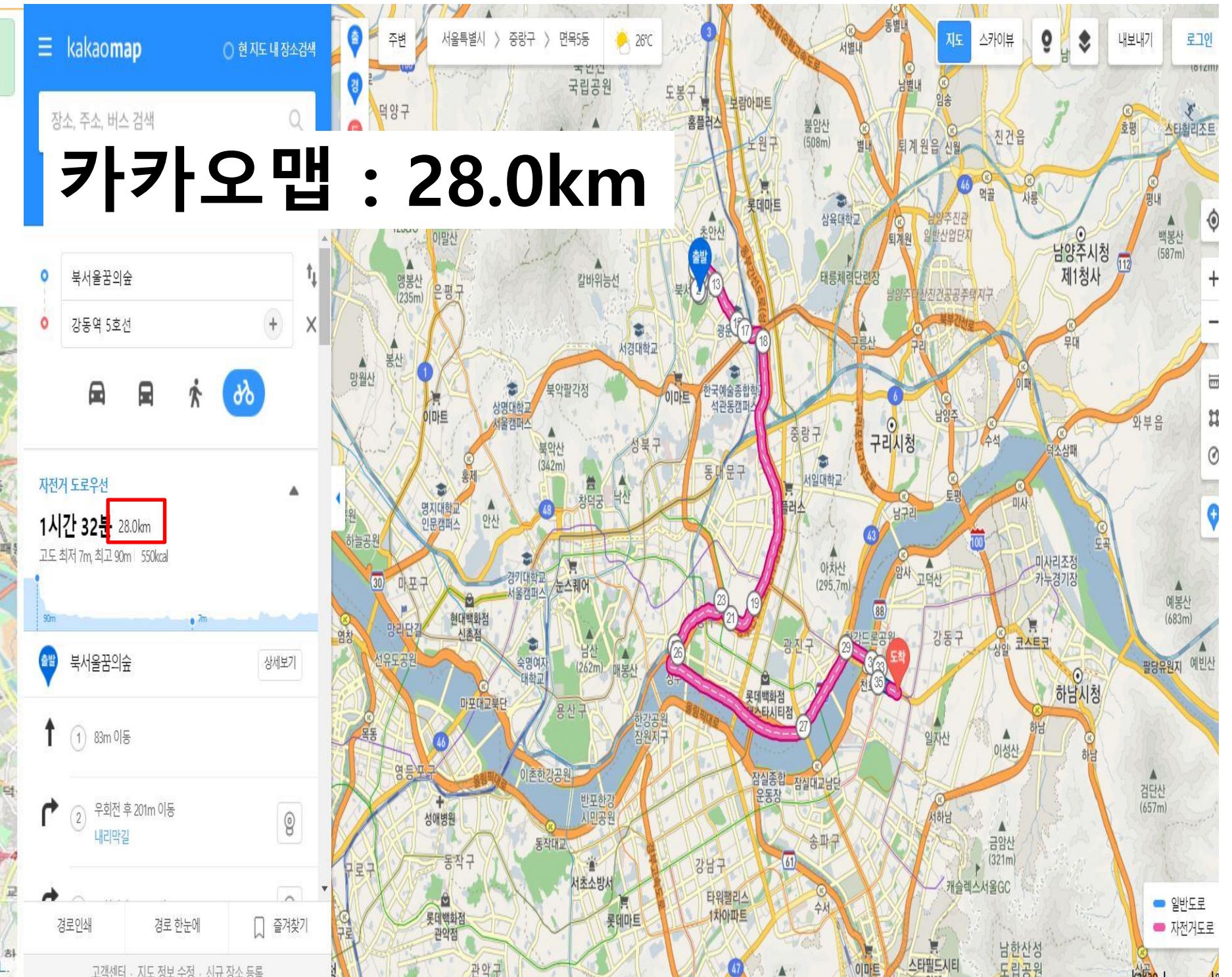


네비게이션 정확도 측정(북서울 꿈의 숲 -> 강동역)

카카오맵 대비 PinEat맵
정확도 : 96%

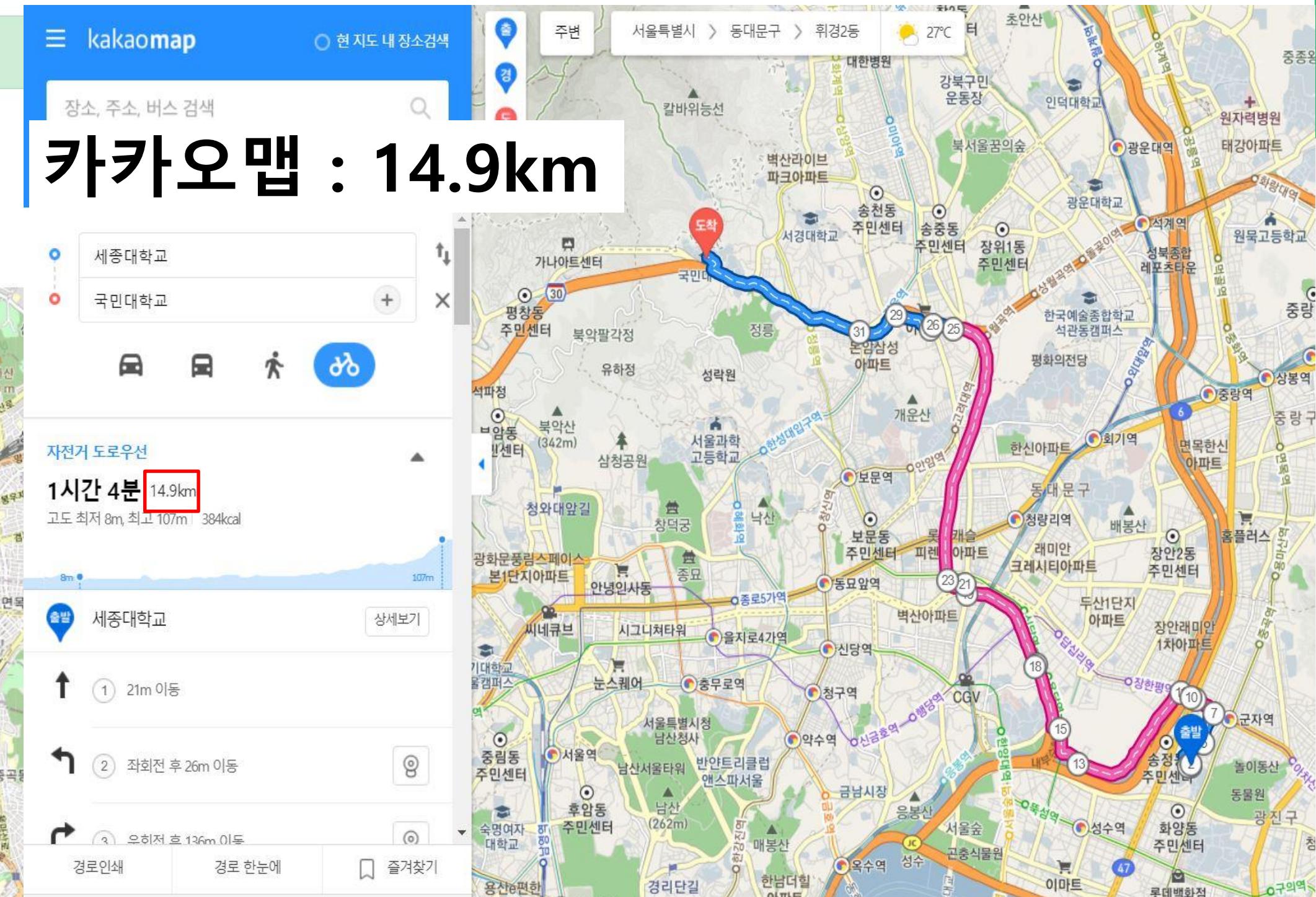
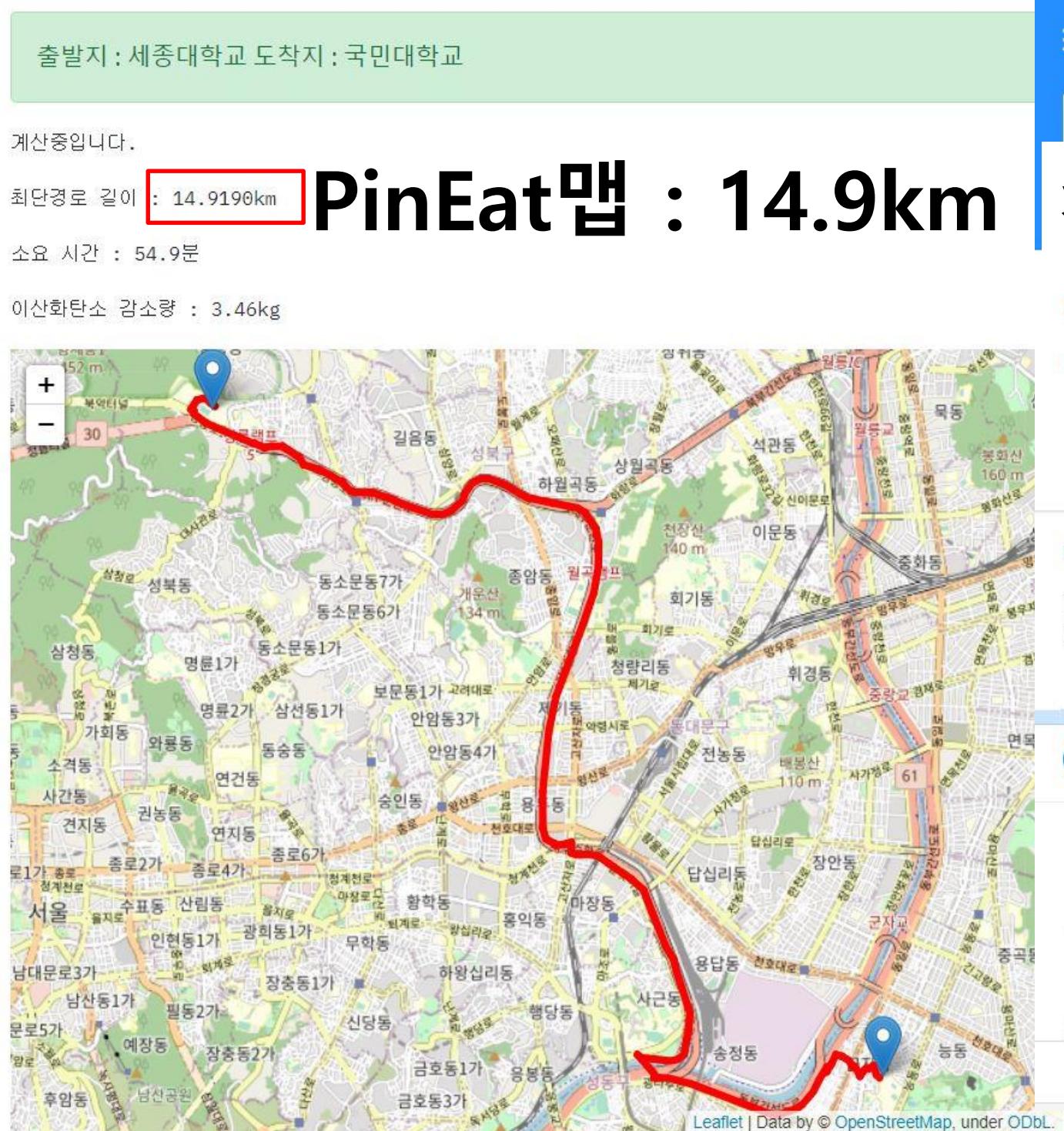


PinEat맵 : 27.01km



네비게이션 정확도 측정(세종대학교 -> 국민대학교)

카카오맵 대비 PinEat맵
정확도 : 100%



경로에 해당하는 Node

```
path = nx.shortest_path(Seoul_Map, orig_node, dest_node, weight = 'energy')  
shortcut = Seoul_Node.loc[path]  
print(shortcut)
```

	y	x	osmid	elevation	highway	ref	#
1168013947	37.551887	127.073240	1168013947	14.631	NaN	NaN	
1168014024	37.552141	127.072919	1168014024	13.603	NaN	NaN	
1168014013	37.552972	127.072551	1168014013	11.681	NaN	NaN	
4630953075	37.552827	127.072268	4630953075	12.618	NaN	NaN	
3846712512	37.552688	127.071997	3846712512	13.113	NaN	NaN	
...	
2322074430	37.559460	126.969443	2322074430	28.739	NaN	NaN	
3893667695	37.559593	126.971291	3893667695	31.464	NaN	NaN	
5593950685	37.557297	126.971890	5593950685	26.478	NaN	NaN	
3391063393	37.555974	126.972274	3391063393	26.831	NaN	NaN	
3391061977	37.554642	126.972060	3391061977	26.209	NaN	NaN	

shortest_path 함수를 활용하여
Weight에 가중치를 입력하고 그
가중치에 근거한 최단 경로 검색

지나간 Node 저장 및 출력
(최다 빈도의 Node를 입지 선정에 활용)

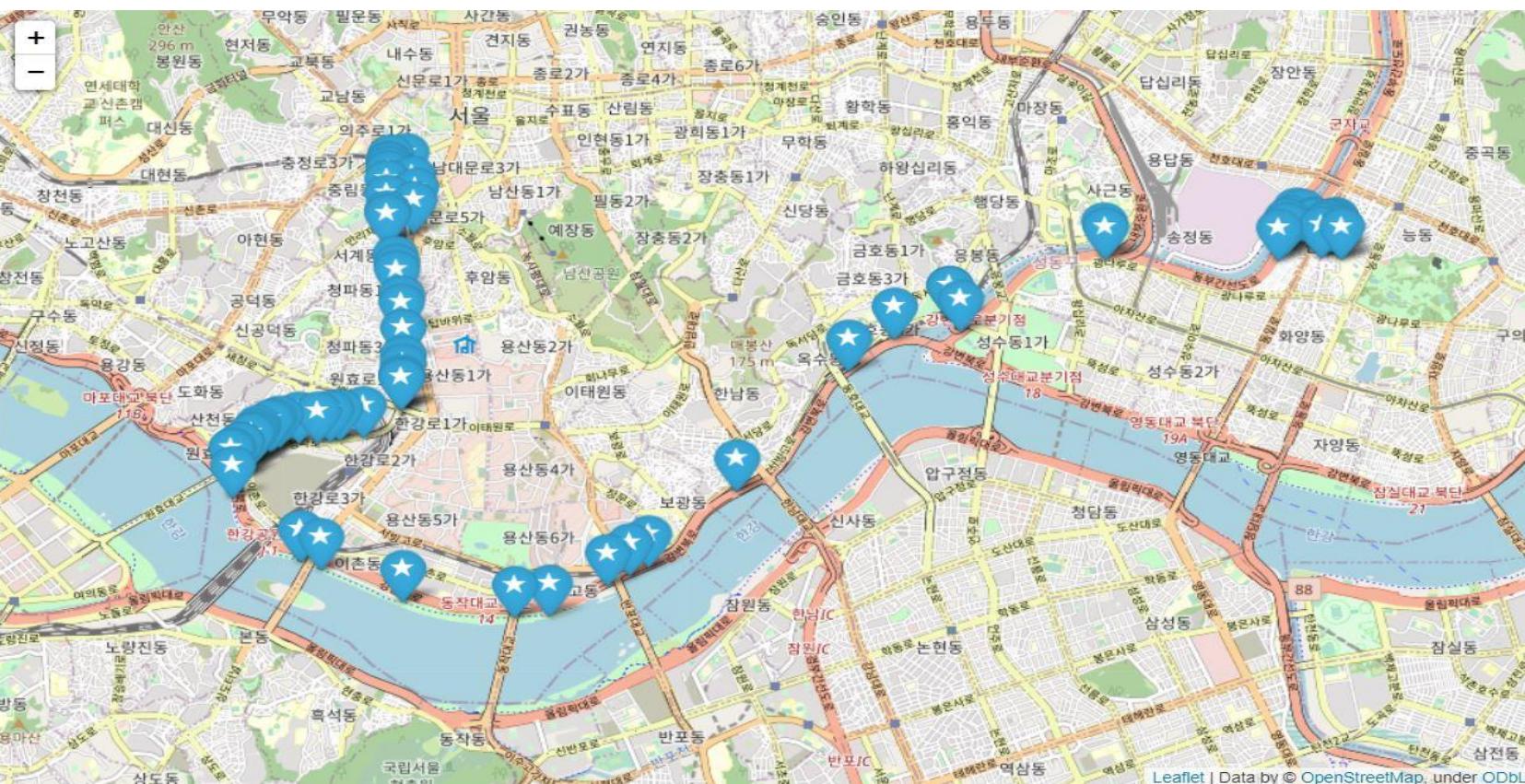
경로에 해당하는 Node 기록

#지나간 노드들 출력

```
map = folium.Map(location=[37.5502, 126.982], zoom_start=11.5)

for i in range(len(gdf_nodes.loc[path])):
    folium.Marker([a.i loc[i][0], a.i loc[i][1]], color = 'blue',
                  icon = folium.Icon(color='red', icon='star')).add_to(map)

map
```



‘세종대학교’에서
‘서울역’까지 최소 에너지 소비
경로를 따라 지나간 Node
출력

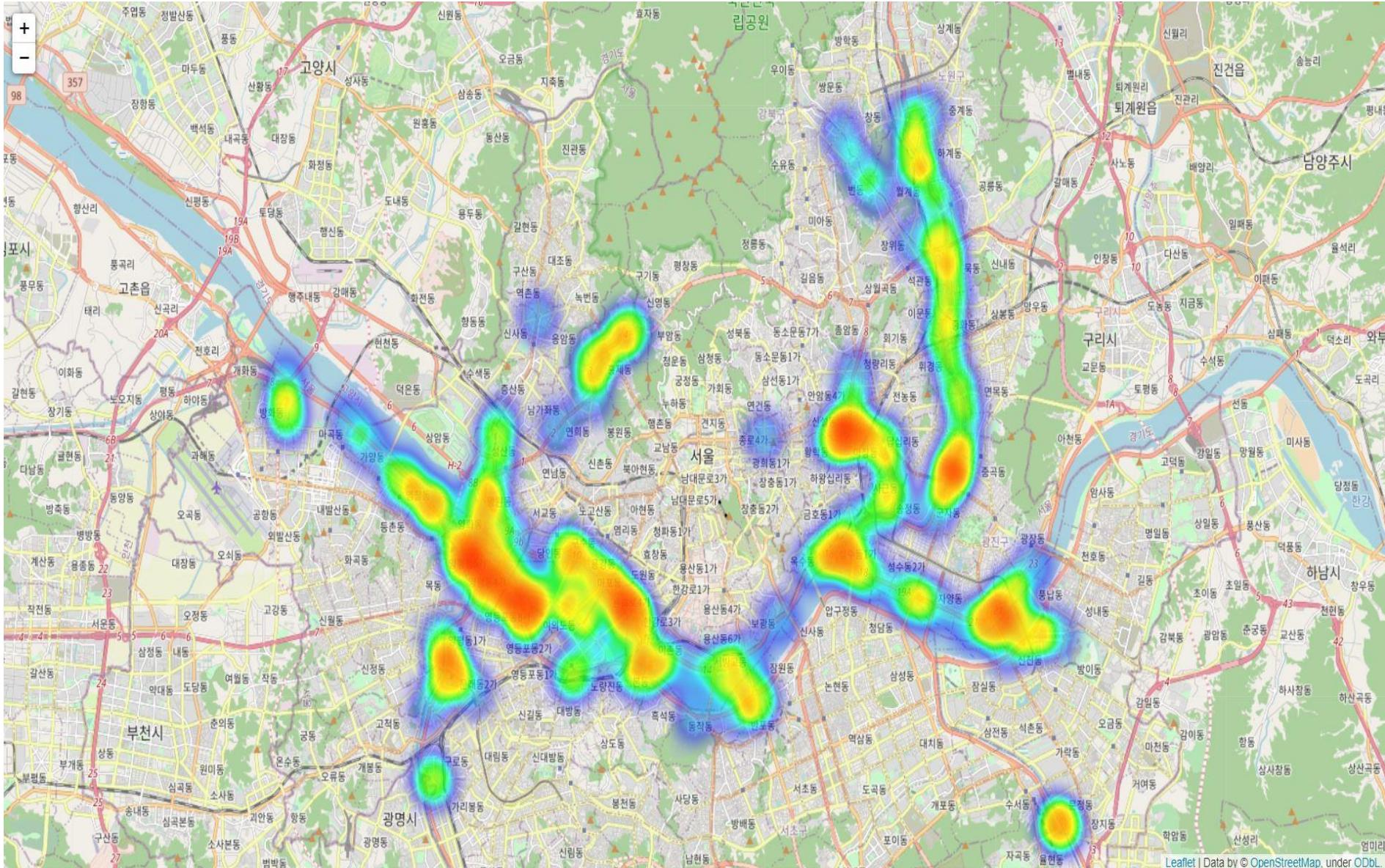
시작/도착 데이터 기록

```
for i in range(150000):
    orig_node = ox.get_nearest_node(Seoul_Map, (start_save[i][0], start_save[i][1])) #시작점
    dest_node = ox.get_nearest_node(Seoul_Map, (arrive_save[i][0], arrive_save[i][1])) #도착지
    path = nx.shortest_path(Seoul_Map, orig_node, dest_node, weight = 'energy')
    shortcut = Seoul_Node.loc[path]

    for j in range(len(shortcut.index)):
        if(shortcut.index[j] in pocket):
            pocket[shortcut.index[j]] +=1
```

서울시 공공자전거의 15만개 출발/도착 정보를
PinEat 네비게이션에 적용해
최소 에너지 소비 경로에 따라
자전거가 지나간 Node를 누적

누적 경로를 반영한 히트맵



최소 에너지 소비 알고리즘을 통해
서울시 공공자전거로 이동 가능한
서울 전역 약 10만개의 지점을
15만개의 승하차 데이터로
카운트하여 최다 빈도 Node 분석

최다 빈도 경로의 상권 분석

DeepLearning을 활용한 업종 추천 및 예상 매출 분석

딥러닝 모델 생성 과정

0.input_grade.csv [C:\Users\WDHA.CNM\Desktop\] - 엑셀

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
상권_구분	상권_코드	서비스_업종	매출단가	점포수	남성_매출	여성_매출	연령대_10	연령대_20	연령대_30	연령대_40	연령대_50	연령대_60	총_유동인구	남성_유동인	여성_유동인	연령대_10	연령대_20	연령대_30	연령대_40	연령대_50	연령대_60	총_직장_인	남
D	1063	2	45161	6	64	36	1	18	29	26	15	11	763095	453533	310456	22748	90189	176526	208629	160090	96804	4898	
A	70	2	56701	2	63	37	1	35	30	19	8	6	508467	233643	274826	15904	270184	124550	47342	31842	18640	603	
A	108	2	18610	1	63	37	1	25	35	23	10	6	153337	93181	60156	2235	35232	36603	31861	32721	14684	2242	
A	61	6	18394	2	33	67	2	63	24	6	5	0	524375	243844	280531	64156	86440	99297	81566	67026	125889	178	
D	1103	8	13440	21	36	65	2	32	31	19	10	6	56902	28758	28144	1629	16562	16196	10771	7075	4671	22562	
U	1496	8	13440	21	36	65	2	32	31	19	10	6	56902	28758	28144	1629	16562	16196	10771	7075	4671	22562	
A	632	6	27948	1	61	39	4	35	26	20	11	4	380836	198441	182398	6706	52836	88362	85888	80637	66412	140	
A	282	3	60570	2	60	40	0	9	23	22	23	23	96409	43898	52513	9056	24195	19151	15926	14937	13140	56	
D	1195	2	16186	1	64	36	2	39	27	18	10	5	109158	55583	53575	4069	36571	18755	18409	17775	13581	668	
D	1024	3	84441	12	72	29	0	19	31	16	17	17	736520	392537	343985	24464	212297	146678	109287	129011	114785	3503	
A	495	2	32528	1	38	62	2	47	32	11	7	2	194244	83764	110482	19523	94837	42654	19872	10733	6626	171	
D	1164	7	26848	4	56	44	2	28	33	22	12	4	710345	371867	338479	28130	102201	173190	181171	133874	91777	11310	
A	858	2	19931	1	65	35	1	17	26	29	17	11	25540	12405	13138	1053	7095	5033	3874	4381	4104	1289	
A	1008	2	24580	4	64	36	1	21	36	23	13	7	180930	88812	92119	6144	23362	39373	39715	42895	29438	594	
D	1100	2	24563	2	63	37	1	26	33	23	12	6	145313	83954	61360	2123	40739	43836	27049	17749	13818	1533	
A	491	1	28259	12	54	46	1	32	31	18	12	6	498871	226816	272055	37895	126991	118438	87424	55675	72449	2761	
D	1129	2	20683	1	61	39	1	24	35	25	10	5	73945	35983	37965	1038	15096	23647	18480	10854	4831	1021	
D	1057	2	19148	6	61	39	4	19	21	23	17	16	870462	463385	407077	44724	145396	199560	209050	156337	115393	3480	
D	1133	4	56152	15	46	54	1	30	37	15	13	6	64980	33648	31332	592	15656	19516	12959	9308	6949	1143	
D	1253	4	12759	4	41	59	2	22	28	29	17	3	1028482	512866	515616	128790	343836	169867	148969	137228	99794	89	
D	1133	4	60473	17	46	54	1	30	41	15	10	4	85305	42969	42336	1474	23907	25170	16387	10826	7543	1791	
D	1063	7	36578	5	64	36	2	13	19	31	23	13	415761	229597	186161	21827	83287	90099	88767	67139	64637	5151	
R	1480	2	33450	2	59	41	1	25	37	22	10	4	309585	170296	139291	23838	102028	72565	44289	38128	28741	404	
D	1106	7	45039	4	60	40	2	32	35	19	9	3	100279	58373	41906	1148	32275	30790	17768	10564	7737	5932	
A	642	2	19363	1	63	37	1	26	36	22	10	4	97521	53225	44293	2279	13666	25208	25418	20836	10114	387	
D	1167	4	63505	8	46	54	1	22	34	23	14	7	360267	194584	165685	11386	77459	93127	69858	62807	45635	4905	
A	879	4	28854	1	41	59	1	59	24	9	7	1	245136	131616	113520	4460	33640	54167	58342	59067	35458	2013	
A	446	4	14031	2	49	51	2	30	34	19	10	5	91123	44899	46225	5785	18498	19152	18388	17369	11928	395	
A	73	2	19770	3	47	54	4	50	21	14	9	2	871801	246663	625139	101469	481143	113030	73430	58478	44256	929	
A	45	4	28761	11	47	54	2	30	29	21	11	6	39167	17901	21266	4740	9079	9252	7508	5188	3398	2984	
R	1480	2	28325	2	62	38	1	28	34	19	12	6	276383	149513	126867	20078	93133	59002	39602	35661	28913	362	
A	781	2	19183	3	61	40	2	54	23	10	8	4	384407	193377	191032	21613	197395	73768	35600	32185	23849	464	
A	411	8	10884	2	42	58	3	16	30	30	15	6	245882	109299	136585	18142	30959	53823	60156	53073	29731	57	
A	780	4	33387	1	50	50	1	50	26	13	8	2	179962	93191	86773	5735	42100	44302	36921	32818	18089	256	
A	408	2	27448	1	63	37	1	10	21	28	23	18	96974	38912	58061	7531	10341	17478	23740	23135	14751	131	
D	1108	1	23639	7	56	44	1	32	35	19	10	5	120303</										

딥러닝 모델 생성

In [1]:

```
1 import numpy as np
2 import pandas as pd
3 import tensorflow as tf
4 import keras
5 import keras.layers.advanced_activations
6 import pickle
7 import sklearn
8
9 from keras.models import load_model
10 from keras.models import Sequential
11 from keras.layers import Dense, BatchNormalization, Activation
12 from keras.layers import PReLU
13 from sklearn.model_selection import train_test_split
14 from sklearn.preprocessing import LabelEncoder, StandardScaler
15
16 # 필요한 모듈들을 불러온다. 사용된 모듈은 numpy, pandas, keras, tensorflow, scikit-learn이다.
```

딥러닝 모델 생성을 위한 모듈 import

딥러닝 모델 생성

```
1 f_type = df['상권_구분_코드'].astype(str)
2 e = LabelEncoder()
3 e.fit(f_type)
4 a = e.transform(f_type)
5
6 # 문자열인 상권_구분_코드를 scikit-learn의 LabelEncoder를 통해 정수로 치환해준다.
```

문자열 데이터 처리 : LabelEncoder

딥러닝 모델 생성

```
1 X = dataset[:,0:60]
2 Y = dataset[:,60]
3 f_type_1 = df['등급']
4 e_1 = LabelEncoder()
5 e_1.fit(f_type_1)
6 Y = e_1.transform(f_type_1)
7 Y_encoded = tf.keras.utils.to_categorical(Y)
8 X_train, X_test, Y_train, Y_test = train_test_split(X, Y_encoded, test_size=0.1, random_state=seed)
9
10 # 전처리한 데이터를 속성값 X와 클래스 Y로 구분해주고 다중분류모델을 위해 클래스인 '등급'을 one-hot encoding 해준다.
11 # 이후 훈련셋과 테스트셋을 9대 1로 나누어준다.
```

Training set, Test set 구분

One-Hot Encoding →

```
1 print(Y_encoded[1])
[1. 0. 0. 0. 0. 0. 0. 0.]
```

딥러닝 모델 생성

```
1 scaler=StandardScaler()  
2 X_train_scale = scaler.fit_transform(X_train)  
3 X_test_scale = scaler.transform(X_test)  
4 # 빠른 학습과 보다 정확한 모델을 위해 Scikit-learn의 StandardScaler를 통해 데이터분포를 scaling해준다  
5 # Scaling을 위해 fitting을 학습셋에 맞추고 그에 맞게 테스트셋을 scaling 해준다.
```

```
1 print(X_train_scale)
```

```
[[ -0.68057342 -0.01343272 -1.30433545 ... -0.05142444 -0.34501279  
-0.60679854]  
[ -0.68057342 -0.40796236 -1.30433545 ... -0.05142444 -0.34501279  
-0.80569092]  
[ 0.81164493  0.74396679 -1.30433545 ... -0.05142444 -0.34501279  
-0.01012141]  
...  
[ -0.68057342  0.57105565 -0.0784423 ... -0.05142444 -0.34501279  
-0.60679854]  
[ -0.68057342 -1.12152524  1.14745086 ... -0.05142444  1.69614195  
-0.20901379]  
[ -0.68057342 -0.07188155 -0.0784423 ... -0.05142444 -0.34501279  
-0.40790616]]
```

StandardScaler를 통해 보다 빠르고 정확한 모델 생성

딥러닝 모델 생성

```
1 model = Sequential()
2 model.add(Dense(256, input_dim=60))
3 model.add(BatchNormalization())
4 model.add(PReLU())
5 model.add(Dense(256))
6 model.add(BatchNormalization())
7 model.add(PReLU())
8 model.add(Dense(9))
9 model.add(BatchNormalization())
10 model.add(Activation('softmax'))
11 model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(lr=0.0001), metrics=['accuracy'])
12 model.fit(X_train_scale, Y_train, epochs=1000, batch_size=4096)
13
```

은닉층 : 2개

은닉층 노드 개수 : 512개

활성화 함수 : PReLU

손실함수 : categorical_crossentropy

최적화 함수 : Adam

출력층 활성화 함수 : softmax

입력층(Input Layer)

은닉층(Hidden Layer)

출력층(Output Layer)

딥러닝 모델 생성

```
1 model.evaluate(X_test_scale,Y_test)[1]  
2  
3 # 모델을 테스트셋으로 평가한다. 정확도는 88%임을 알 수 있다.  
9732/9732 [=====] - 1s 55us/step  
0.8858405351638794
```

딥러닝 모델 정확도 88%

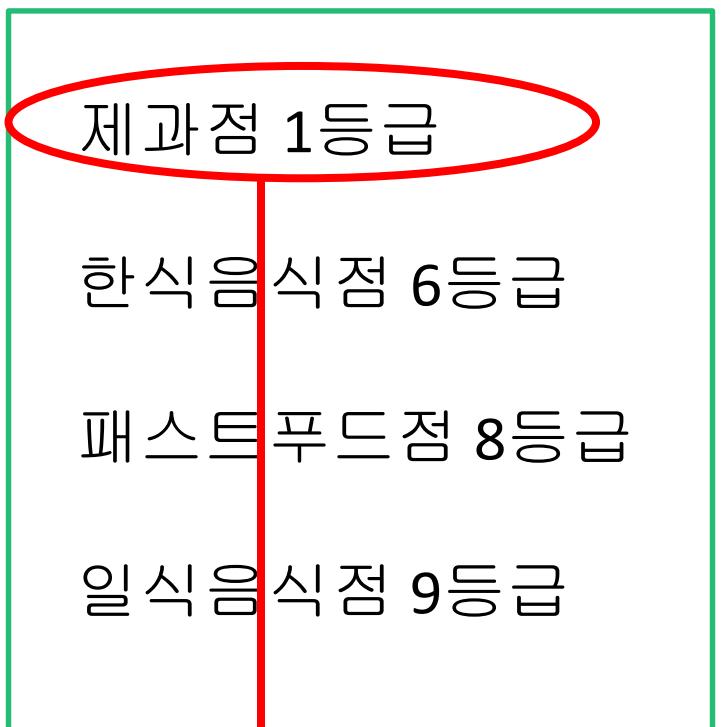
상권_구분_코드	상권_코드	서비스_업종_코드	매출_단가	점포수	남성_매출_비율	여성_매출_비율	연령대_10_매출_비율	연령대_20_매출_비율	연령대_30_매출_비율	백화점_점수	슈퍼마켓_수	극장_수	숙박_시설_수	공항_수	철도역_수	버스터미널_수	지하철_수	버스정거장_수	등급
0	0	55	9	3000	3	52	48	0	33	29	0	1	0	0	0	0	0	1.8	
1	0	55	1	3000	12	72	28	1	20	26	0	1	0	0	0	0	0	1.7	
2	0	55	9	4000	3	52	48	0	33	29	0	1	0	0	0	0	0	1.8	
3	0	55	1	4000	12	72	28	1	20	26	0	1	0	0	0	0	0	1.7	
4	0	55	9	5000	3	52	48	0	33	29	0	1	0	0	0	0	0	1.8	

딥러닝 모델을 통해 입지의 상권 데이터를 적용시켜 예상 매출 등급을 도출

상권 분석 딥러닝

예시

가재울로 6길



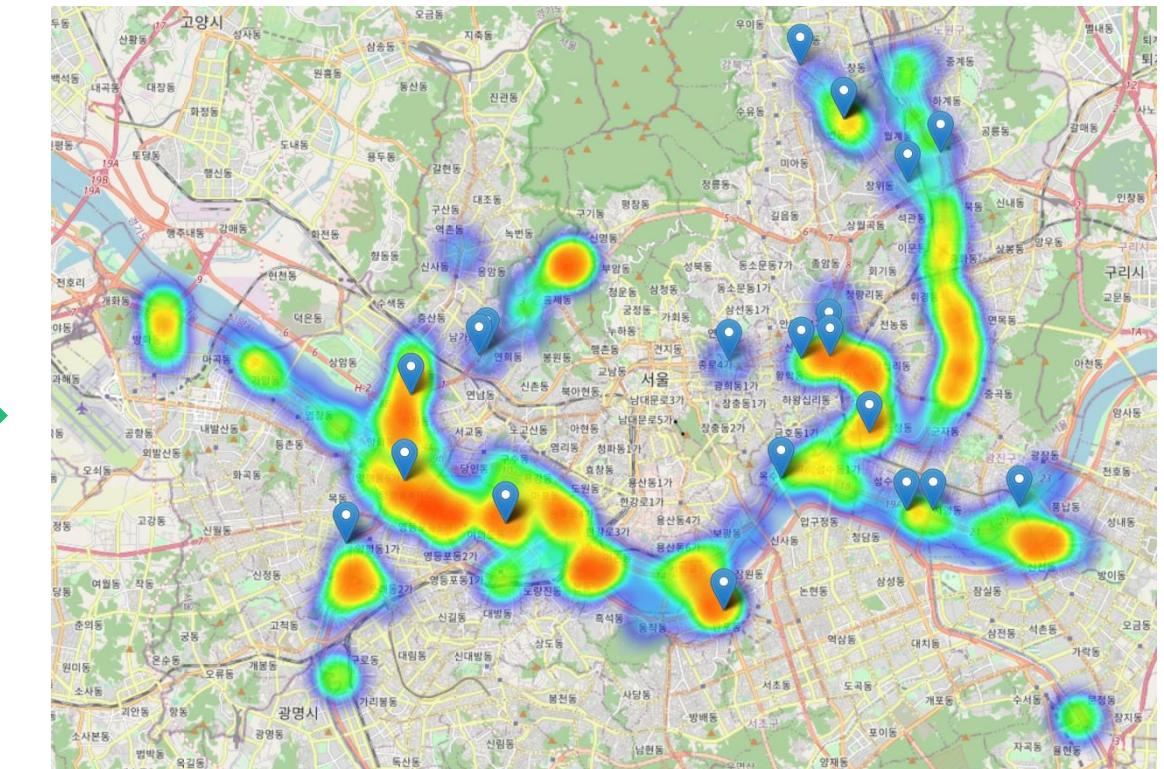
가재울로 6길
추천 업종 : 제과점
단가 : 3~8000원 선정

총 21개 지역

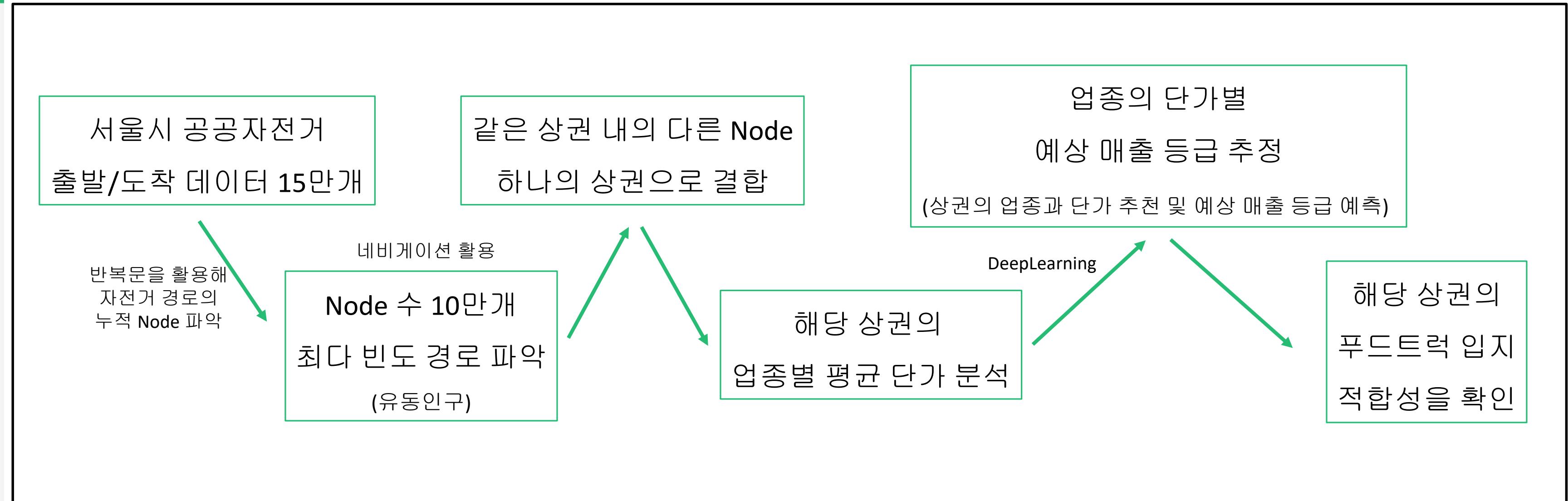
가산디지털단지역_2
가재울로6길
고속터미널역
난계로30길
능동로4길
당산역_1
동일로173가길
뚝섬로24길
모래내로15길
무학로16길
방울내로9안길
상원12길
서울 광진구 강변역
서울 영등포구 여의도역_2
서울 종로구 종로5가역_3
석계로1길
오목교역
천호대로17길
한림말3길
한천로109길
한천로132길



21개 위치
시각화



PinEat 푸드트럭 입지 선정 프로세스



데이터 분석



서울 열린데이터 광장

{ 서울시 지하철 역별 승하차 인원정보

{ 서울시 공공자전거 정보(이용현황, 대여이력정보, 실시간 대여 정보



우리 마을 가게
상권분석서비스

{ 분기별 상주인구, 직장인구, 추정 유동인구

{ 분기별 집객시설, 추정 매출 등



Kakao API

{ 실시간 위도, 경도 정보



OpenStreetMap

{ 실시간 서울 지도 정보



Google Maps API

{ 실시간 고도 정보

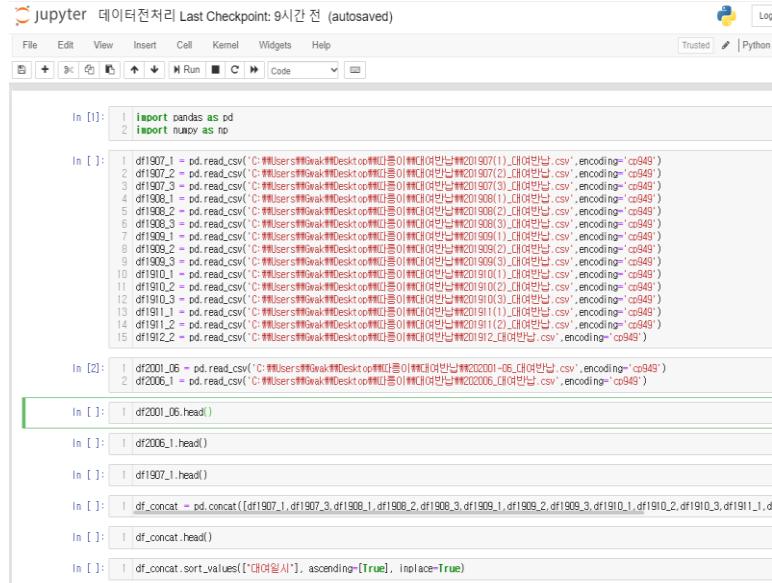
데이터 분석

데이터 수집

데이터 전처리

경로탐색 및 학습

시각화 및 분석



```
In [1]: import pandas as pd
import numpy as np

df1907_1 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190711_데이터만남.csv', encoding='cp949')
df1907_2 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190712_데이터만남.csv', encoding='cp949')
df1907_3 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190713_데이터만남.csv', encoding='cp949')
df1908_1 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190801_데이터만남.csv', encoding='cp949')
df1908_2 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190802_데이터만남.csv', encoding='cp949')
df1908_3 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190803_데이터만남.csv', encoding='cp949')
df1909_1 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190901_데이터만남.csv', encoding='cp949')
df1909_2 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190902_데이터만남.csv', encoding='cp949')
df1909_3 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20190903_데이터만남.csv', encoding='cp949')
df1910_1 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191001_데이터만남.csv', encoding='cp949')
df1910_2 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191002_데이터만남.csv', encoding='cp949')
df1910_3 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191003_데이터만남.csv', encoding='cp949')
df1911_1 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191101_데이터만남.csv', encoding='cp949')
df1911_2 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191102_데이터만남.csv', encoding='cp949')
df1911_3 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191103_데이터만남.csv', encoding='cp949')
df1912_1 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191201_데이터만남.csv', encoding='cp949')
df1912_2 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\20191202_데이터만남.csv', encoding='cp949')

df2001_06 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\2020_06_데이터만남.csv', encoding='cp949')
df2006_1 = pd.read_csv('C:\Users\Geek\Desktop\테이블데이터\여민남\202006_데이터만남.csv', encoding='cp949')

In [1]: df2001_06.head()

In [1]: df2006_1.head()

In [1]: df1907_1.head()

In [1]: df_concat = pd.concat([df1907_1, df1907_2, df1908_1, df1908_2, df1908_3, df1909_1, df1909_2, df1910_1, df1910_2, df1911_1, df1911_2, df1911_3, df1912_1, df1912_2])

In [1]: df_concat.sort_values(['대여일시'], ascending=[True], inplace=True)
```

자전거번호	대여일시	대여 대여소 번호	대여 대여소명	대여거치대	반납일시	반납대여소 번호	반납대여소명	반납거치대	이용시간	이용거리
0	SPB-04061	2020-01-01 0:01:00	429	송도병원	2	2020-01-01 0:04	372	악수역 3번출구 뒤	8	2.0
1	SPB-06686	2020-01-02 0:01:00	1637	KT 노원점 건물 앞	14	2020-01-01 0:05	1656	중앙아이즈 아파트 입구	9	1.350.0
2	SPB-15937	2020-01-03 0:01:00	1924	삼부르네상스파크빌	10	2020-01-01 0:05	1955	디자힐입구 고자로	7	4.800.0
3	SPB-14805	2020-01-04 0:01:00	437	대륭역 1번출구	1	2020-01-01 0:05	126	서강대 후문 옆	18	2.0
4	SPB-09038	2020-01-05 0:01:00	1168	마곡암밸리10단지 앞	5	2020-01-01 0:05	1152	마곡역교차로	2	4.660.0
...
104870	SPB-32717	2020-06-16 22:55	913	이마트 은평점	0	2020-06-16 23:03	931	역촌파출소	0	8.0
1048571	SPB-15338	2020-06-16 22:46	167	연가초등학교 옆	13	2020-06-16 23:03	175	풀연고열	20	17.0
1048572	SPB-31982	2020-06-16 22:48	559	왕십리역 4번 출구 건너편	0	2020-06-16 23:03	578	성동세무서 부근	0	16.0
1048573	SPB-24981	2020-06-16 22:10	1168	마곡암밸리10단지 앞	9	2020-06-16 23:03	1168	마곡암밸리10단지 앞	9	52.10290.0
1048574	SPB-14853	2020-06-16 22:35	2115	관악농협은행 지하철	13	2020-06-16 23:03	1822	서울시립종합체육관	3	28.0



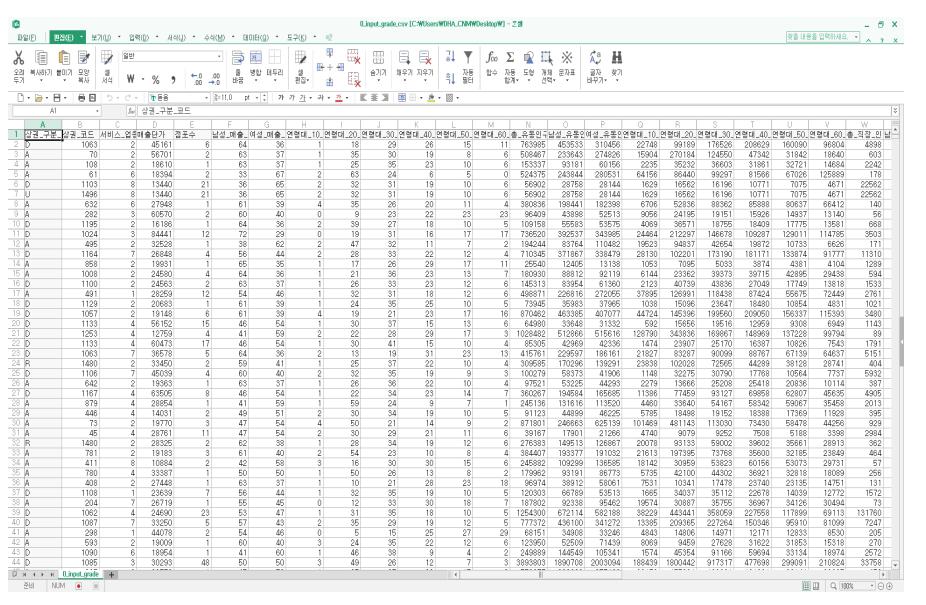
NumPy



Pandas

지하철 이용량 데이터

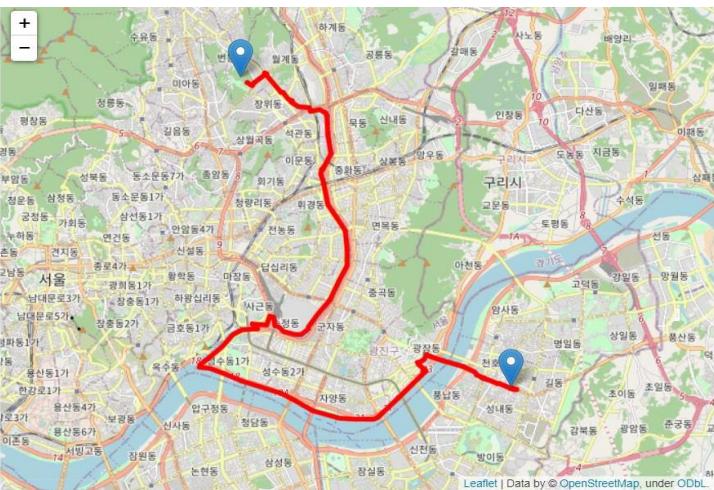
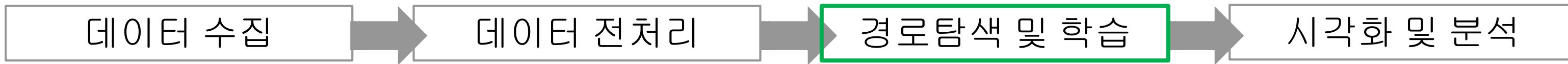
상권 분석 데이터 → 딥러닝 모델 생성



2097160 rows x 11 columns

공공자전거 승하차 데이터 등

데이터



GeoPandas

최적 경로 자전거용

네비게이션 구현

상권분석 딥러닝 모델

```
1 model = Sequential()
2 model.add(Dense(256, input_dim=60))
3 model.add(BatchNormalization())
4 model.add(PReLU())
5 model.add(Dense(256))
6 model.add(BatchNormalization())
7 model.add(PReLU())
8 model.add(Dense(9))
9 model.add(BatchNormalization())
10 model.add(Activation('softmax'))
11 model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(lr=0.0001), metrics=['accuracy'])
12 model.fit(X_train_scale, Y_train, epochs=1000, batch_size=4096)
```



데이터

데이터 수집

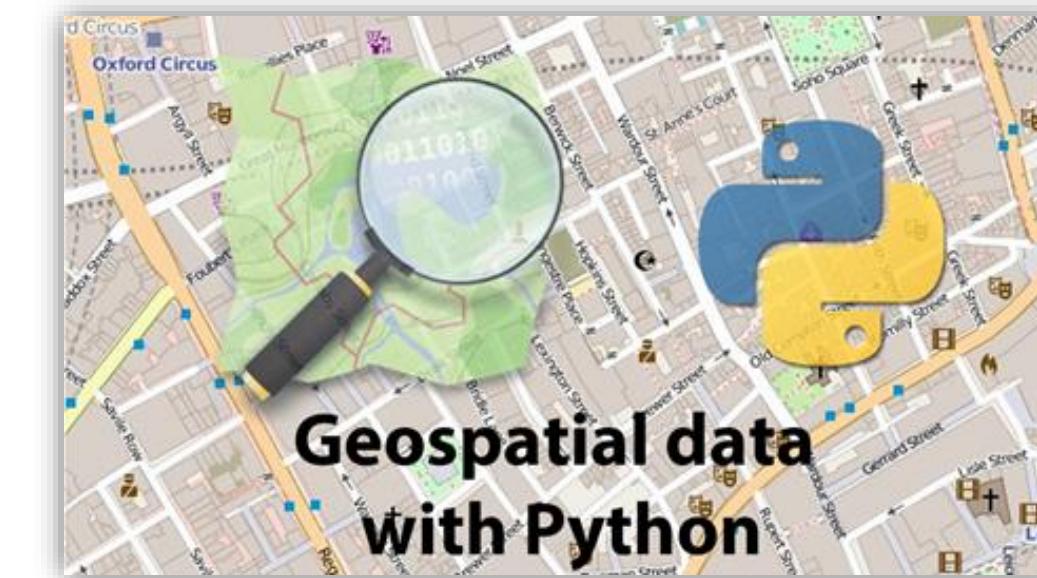
데이터 전처리

경로탐색 및 학습

시각화 및 분석

최다 빈도 노드 분석 히트맵 시각화

최다 빈도 노드(지역) 분석 데이터를 입력하여
해당 지역 매출 등급 예측



- Show the proper place for foodtruck

상권_구분_코드	상권_코드	서비스_업종_코드	매출_단가	점포_수	남성_매출_비율	여성_매출_비율	연령대_10_매출_비율	연령대_20_매출_비율	연령대_30_매출_비율	...	백화점_수	슈퍼마켓_수	극장_수	숙박_시설_수	공항_수	철도역_수	버스터미널_수	지하철역_수	버스정거장_수	등급
0	0	55	9	3000	3	52	48	0	33	29	...	0	1	0	0	0	0	0	0	8
1	0	55	1	3000	12	72	28	1	20	26	...	0	1	0	0	0	0	0	0	7
2	0	55	9	4000	3	52	48	0	33	29	...	0	1	0	0	0	0	0	0	8
3	0	55	1	4000	12	72	28	1	20	26	...	0	1	0	0	0	0	0	0	7
4	0	55	9	5000	3	52	48	0	33	29	...	0	1	0	0	0	0	0	0	8

활용 데이터 (Raw DATA)

#	출처	데이터 명	크기	단위	수집 기간
1	서울시 우리마을가게 상권분석서비스	상주인구	6.31 MB	분기별	2014년 ~ 2020년
2	서울시 우리마을가게 상권분석서비스	직장인구	5.1 MB	분기별	2014년 ~ 2020년
3	서울시 우리마을가게 상권분석서비스	추정유동인구	430.5 MB	분기별	2014년 ~ 2020년
4	서울시 우리마을가게 상권분석서비스	추정매출	442 MB	분기별	2014년 ~ 2020년
5	서울시 우리마을가게 상권분석서비스	집객시설	1.7 MB	분기별	2015년 ~ 2020년
6	서울 열린데이터 광장	서울시 지하철 호선별 역별 승하차 인원정보	15.7 MB	월별	2018년 11월 ~ 2020년 06월
7	서울 열린데이터 광장	서울시 공공자전거 이용현황	55.4 MB	월별	2018년 11월 ~ 2020년 6월
8	서울 열린데이터 광장	서울특별시 공공자전거 대여이력 정보	336.8 MB	월별	2020년 01월 ~ 2020년 06월

활용 API(Raw DATA)

API

API	제공자	설명	설명	설명
1	서울시 열린데이터 광장	서울특별시 공공자전거 실시간 대여 정보	API (누적 노드 수)	2일
2	OpenStreetMap	OSMNX	5.1 MB (서울지도정보)	실시간
3	KAKAO Developers	KAKAO Maps	430.5 MB (위경도 정보)	실시간
4	Google Cloud Platform	Google Maps Elevation	442 MB (고도 정보)	실시간

한계점 및 결론

한계점

1. 따릉이 이용자의 실제 이동 데이터가 아니라 예측된 최적 이동경로이기 때문에 현실과 차이 발생 가능성 존재(자전거 네비게이션을 활용해야함)
2. 푸드트럭 실 매출 데이터가 아닌 해당 지역의 상권 데이터로 부동산 매장과의 특성 차이로 입지 차이 발생 가능성 존재
3. 상권 분석 서비스의 데이터가 업종 내에 2개 이하의 점포를 보유할 경우 데이터를 제공하지 않아 분석 불가능

결 론

코로나19 사태 이외에도

전년도 대비 따릉이 사용이 증가함에 따라

따릉이 이용자가 많이 다니는 최적 경로 예측을 통해

푸드트럭 입지를 선정 후

입지의 신규영업장소 적합성을 딥러닝을 통해 입증

감사합니다