우리의 여행을 더 쉽고, 즐겁게!

여행 종합 추천 시스템

EASY TRIP

step1 맛,고스톱

It's being made by 데이터해적단

(곽주원, 김기범, 이산하, 장지수)

PART 1

우리 동료가 되지 않을래?

데이터 해적단원 모집 (2명)

함께 할 스터디원 모집합니다 !

프로젝트 발표를 보시고,
함께 만들며 공부하고 싶다면 연락주세요 !

프로젝트 분야 : 부동산 & 여행 관련 서비스

스터디 내용
- 프로젝트에 필요한 제반 내용을
  적극적으로 공부하고 적용해봅니다.

PART 1

해결하고자 하는 문제정의 (5whys)

1. 저번에 여행 때 숙소, 맛집 잘못 골라서 여행 망치고 너무 속상해😢

   Why ? 네x버 검색 위에 뜬 거 고른건데.. 가격 대비 별로였어..

2. 처음엔 꼼꼼히 검색하다가 힘들어서 그냥 위에 뜬 거 골랐어..

   Why ? 검색결과도 리뷰도 너무 많아서 그냥 위에 뜬 것만 봤어

3. 그래도 나름 평점도 높고 리뷰 많고 괜찮았는데 왜 별로인거지?

   Why ? 사이트 광고, SNS 마케팅 많이 한 곳이 상위노출 되는거 같아..

검색데이터가 너무 많아 사이트에서 필터링해 상위 노출시키지만,

마케팅으로 상위 노출되는 경우도 많음.

PART 1

해결 서비스 기획 : 이지트립

해결 서비스 기획 : 데이터 기반 맛집,숙소 추천

- 여러 사이트에 있는 맛집 + 숙소 검색 결과 수집 및 정제
- 대화형 UI 를 통해 위치 및 키워드 기반의 정보 표시
- 한 눈에 볼 수 있고, 쉽게 확인할 수 있도록 지도 시각화

Step 1. 맛집 추천시스템 → 맛집고민, 그만! 맛,고스톱

맛,고스톱 개요(1)

- 목표 : 여러 사이트들이 공통적으로 추천하는 맛집은 어디?

1. 데이터 수집 (크롤링)

- 네이버, 카카오맵, 구글 등 검색사이트
- 망고플레이트, 트립어드바이저 등 여행사이트

2. 정제 및 모델링

- 가중평점을 만들어 평점을 통일시킴
- CBF : 키워드별 추천시스템 (현재, 망고플레이트 데이터에만 적용)
- CF : 각 사이트를 유저로 가정하고 적용

  e.g. 구글 검색에 없는 식당의 예측평점을 다른 사이트를 통해 적용

맛,고스톱 개요(2)

3. 프로젝트 중간 평가 (ft.팀플의 필요성과 어려움..)

- 정말 많은 데이터 필요 + 데이터 수집하며 방향 개선
- 지속적인 커뮤니케이션 중요 (방향설정 or 진행하며 이슈 공유)
- 데이터 수집 ↔ 모델 적용을 오가며 어떻게 수집, 적용할지 계속 고민

4. 향후 진행 방향

- 맛,고스톱에서 구축된 모델을 숙소 추천시스템에도 적용

PART 1

수집자료 활용법 #1 평점랭킹

평점 총합 높은 식당 정렬
(망고플레이트 200개 식당 기준)

- 4개 사이트 평점자료 합침 (merge)
- 각 사이트별 평점 합친 total 컬럼 생성(sum)

→ 많은 사이트로부터 높은 평점을 받은 식당을 한 눈에 확인 가능

	Title	Mango	Naver	Google	Kakao	Total	Type
2	해목	4.5	4.44	4.3	3.9	17.14	정통 일식 일반 일식 특히츠마부시 민물장어덮밥 특카이센동 해산물덮밥 카이센동 해산물...
10	고옥	4.4	4.48	4.2	3.7	16.78	돈부리 일본 카레 벤토
149	에쎄떼	3.9	4.57	4.3	3.7	16.47	카페 디저트
153	기장손칼국수	3.9	4.40	4.0	4.0	16.30	국수 면 요리

PART 1

네이버 자료수집

네 이 버

부산 맛집

300

50개 x 6페이지

크롤링 코드

네이버 지도 왼쪽 검색결과 iframe 접근

```
driver.switch_to.frame("searchIframe")
```

페이지 넘기기 & 스크롤 내리기

```
# ActionChanins 모듈
from selenium.webdriver.common.action_chains import ActionChains

# for문으로 페이지 넘기기 > 스크롤 내리기 반복
for i in range(1,11):
    driver.find_element_by_link_text(str(i)).click()
    for j in range(3,55,3):
        element = driver.find_elements_by_css_selector('.OXiLu')[j]
        ActionChains(driver).move_to_element(element)
            .key_down(Keys.PAGE_DOWN).key_up(Keys.PAGE_DOWN).perform()
```

PART 1

Data 컬럼 전처리 코드

```
df['data'][0:10]

0                영업 전방문자리뷰 7,721저장수 58,000+
1    영업 전\n별점\n4.22방문자리뷰 3,450저장수 54,000+
2      영업 전\n별점\n4.44방문자리뷰 922저장수 52,000+
3    영업 전\n별점\n4.46방문자리뷰 4,595저장수 51,000+
```

숫자데이터만 추출

1. 별점 (rating)

2. 방문자리뷰 (review)

3. 저장수 (save)

```
d = df[~df['data'].str.contains('별점')].index
refined_df = df.drop(index=d, axis=0) # 별점 없는 데이터 제거

data_list = refined_df['data'].values.tolist() # 리스트로 변경

for i in range(len(data_list)): # for문으로 전처리
    data_list[i] = data_list[i].split('별점')[1]
    data_list[i] = data_list[i].split('\n')[1]   # 별점 숫자 앞 제거
    data_list[i] = data_list[i].replace('방문자리뷰', '').replace('블로그리뷰','')    # 나머지 한글 제거
    data_list[i] = data_list[i].split(' ')  # 별점, 방문자리뷰, 저장수 분리
```

title	rating	review	save
해운대암소갈비집	4.22	3,450	54,000+
해목	4.44	922	52,000+
신발원	4.46	4,595	51,000+
해성막창집 본점	4.20	1,432	47,000+

PART 2

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

원래 의도 한번에 가게 이름, 별점, 리뷰수, 업종을 조사하려고 했다. beautifulsoup가 익숙하지 않아.가게 이름만 뽑아내고 다시 재입력하는 코드를 돌려 이름, 별점, 리뷰수, 업종을 정리했다.

```
import os
from time import sleep

import pandas as pd
import chromedriver_autoinstaller
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import ElementNotInteractableException
from selenium.common.exceptions import StaleElementReferenceException
from bs4 import BeautifulSoup

path = chromedriver_autoinstaller.install()
driver = webdriver.Chrome(path)
```

PART 2

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

원래 의도 한번에 가게 이름, 별점, 리뷰수, 업종을 조사하려고 했다. beautifulsoup가 익숙하지 않아.가게 이름만 뽑아내고 다시 재입력하는 코드를 돌려 이름, 별점, 리뷰수, 업종을 정리했다.

```
try:
    #driver.find_element_by_xpath('//*[@id="info.search.place.more"]').send_keys(Keys.ENTER)
    #sleep(1)

        # 2~ 5페이지 읽기
    for i in range(2, 6):
            # 페이지 넘기기
        xPath = '//*[@id="info.search.page.no' + str(i) + '"]'
        driver.find_element_by_xpath(xPath).send_keys(Keys.ENTER)
        sleep(1)

        html = driver.page_source
        soup = BeautifulSoup(html, 'html.parser')
        place_lists += soup.select('.placelist > .PlaceItem') # 장소 목록 list
```

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

```
# 다음 넘어가는 것 누르고 6페이지 가게 리스트 뽑기
# 6페이지에서 리스트가 뽑히지 않고 넘어가지도 않는다.
# sleep(1) 잠시 대기하면서 6페이지 읽기 문제 해결
driver.find_element_by_xpath('//*[@id="info.search.page.next"]').send_keys(Keys.ENTER)
sleep(1)

html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
place_lists += soup.select('.placelist > .PlaceItem') # 장소 목록 list
```

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

```
# 평점 크롤링
    try:
        rate = driver.find_element_by_css_selector("#info\.search\.place\.list >
li.PlaceItem.clickArea.PlaceItem-ACTIVE > div.rating.clickArea > span.score > em").text
    except Exception as e1:
        print("정보 없음")
        rate="정보 없음"
        pass
    # 리뷰 수
    try:
        rateNum = driver.find_element_by_css_selector("#info\.search\.place\.list >
li.PlaceItem.clickArea.PlaceItem-ACTIVE > div.rating.clickArea > span.score > a").text
    except Exception as e1:
        print("정보 없음")
        rateNum="정보 없음"
    # 식당 종류
    try:
        cate=driver.find_element_by_css_selector('#info\.search\.place\.list >
li.PlaceItem.clickArea.PlaceItem-ACTIVE > div.head_item.clickArea > span').text
    except Exception as e1:
        print("정보 없음")
        cate="정보 없음"
```

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

```
# 뽑아낸 리스트를 가져와 가게이름을 저장한다
df=[]
for i, place in enumerate(place_lists):
    place_name = place.select('.head_item > .tit_name > .link_name')[0].text
    df.append(place_name)
```

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

알게된 점1

```
# enumerate 오류
# kakao_keyword에는 문자형 데이터가 들어가야한다.
# enumerate를 사용하면 자료형이 tuple 형태로 변환된다.
# kakao_keyword에 원하는 값이 입력되지 않는다.
for i, keyword in enumerate(df['kakao_keyword'].tolist()):
    print(keyword, end=" ")
    try:
        kakao_map_search_url = f"https://map.kakao.com/?q={keyword}"
        driver.get(kakao_map_search_url)
        time.sleep(1)

# 강사님 답변
# enumrate 제거
for keyword in df['kakao_keyword'].tolist():
    print(keyword, end=" ")
    try:
        kakao_map_search_url = f"https://map.kakao.com/?q={keyword}"
        driver.get(kakao_map_search_url)
        time.sleep(1.5)
```

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

알게 된 점2

스크롤 내리는 코드
driver.execute_script("window.scrollTo(0, 400)")
주의할 점
400자리를 변경해주는데 모니터 크기, 배율에 따라 스크롤 내려가는 정도가 달라진다.
너무 적게 내려가거나 또는 많이 내려가서 읽혀지지 않아 오류가 발생할 수 있다.
400에서 100 단위로 위아래 재설정하면서 맞춰준다.

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

알게 된 점3

광고창 지우기 코드

```
# 광고창 iframe 접근하기
driver.switch_to.frame("google_ads_iframe_/395211568/init/desktop_all_0")

# '다시 보지 않기' 버튼 클릭하기
element = '#ad > div > button.ad_btn.ad_block_btn'
driver.find_element_by_css_selector(element).click()
```

selenium과 BeautifulSoup 이용 카카오맵 상위 225개 맛집 크롤링하기

Q.5개의 사이트 중 점수가 가장 후한 곳은 어디인가 ?

A.

1위 naver 4.451457
2위 trip 4.226848
3위 mango 4.067983
4위 google4.018847
5위 kakao 3.650640

데이터 크롤링

In [1]:

```
import chromedriver_autoinstaller
path = chromedriver_autoinstaller.install()

# 경고 무시
import warnings
warnings.filterwarnings('ignore')

from selenium import webdriver

driver = webdriver.Chrome(path)

# 구글로 이동
driver.get('https://www.google.com')

# 창 최대
driver.maximize_window()

# 검색어 입력
keyword = '부산광역시 맛집'

# 검색창으로 이동
element = driver.find_element_by_css_selector(".a4bIc > input")

element.clear()

element.click()

element.send_keys(keyword)

element.submit()

# 장소 더보기 클릭하여 지도 전체 리스트 확인
driver.find_element_by_link_text('장소 더보기').click()
```

In [6]:

```
from tqdm import tqdm_notebook

import re

import time

import pandas as pd

dict = {}

base = 0

# 첫 페이지로 이동
driver.find_element_by_css_selector('#mKlEF > span > svg').click()

# 페이지 수 설정
for i in tqdm_notebook(range(15)):

    try:
        # 한 페이지에 있는 가게 이름 링크 갯수
        n = len(driver.find_elements_by_css_selector('.dbg0pd.0SrXXb.eDlkBe > span:nth-child(1)'))

        for i in range(n):
            # 이름 링크 클릭 >> 상세 페이지
            driver.find_elements_by_css_selector('.dbg0pd.0SrXXb.eDlkBe > span:nth-child(1)')[i].click()
            time.sleep(2)

            try:
                store_info = {}

                # 가게 이름
                name = driver.find_element_by_css_selector('.SPZz6b > h2 > span').text
                store_info['name'] = name

                # 평점
                rating = float(driver.find_element_by_css_selector('div:nth-child(2) > div:nth-child(1) > div > div > span.Aq14fc').text)
                store_info['rating'] = rating

                # 리뷰
                review = int(re.findall('Wd+', # 숫자만 추출
                                        re.sub(',','', # 콤마 공백으로 대체
                                               driver.find_element_by_css_selector('.hqzQac > span > a > span').text))[0])
```

```
                store_info['review'] = review

                # 음식 종류
                try:
                    category = driver.find_element_by_css_selector('div:nth-child(2) > div:nth-child(2) > div > span:nth-child(2)').text
                    store_info['category'] = category
                except: # 밀려나서 수집이 안되는 경우
                    category = driver.find_element_by_css_selector('.kp-header > div > div:nth-child(2) > div:nth-child(2) > div > span').text
                    store_info['category'] = category

                dict[base + i] = store_info

            # 평점이 없거나 리뷰가 없는 가게는 수집하지 않고 다음 가게 수집 계속 진행
            except:
                continue

        # 다음 페이지 넘기기
        if driver.find_element_by_css_selector('#pnnext > span.SJajHc.NVbCr'):
            driver.find_element_by_css_selector('#pnnext > span.SJajHc.NVbCr').click()
            time.sleep(2)

    except:
        continue

    base += n

#-----------------------------------------------------------------------------
# 마지막 페이지

# 한 페이지에 있는 가게 이름 링크 갯수
n = len(driver.find_elements_by_css_selector('.dbg0pd.0SrXXb.eDIkBe > span:nth-child(1)'))

for i in range(n):
    # 이름 링크 클릭 >> 상세 페이지
    driver.find_elements_by_css_selector('.dbg0pd.0SrXXb.eDIkBe > span:nth-child(1)')[i].click()
    time.sleep(2)

    try:
        store_info = {}

        # 가게 이름
        name = driver.find_element_by_css_selector('.SPZz6b > h2 > span').text
        store_info['name'] = name
```

```
        # 평점
        rating = float(driver.find_element_by_css_selector('div:nth-child(2) > div:nth-child(1) > div > div > span.Aq14fc').text)
        store_info['rating'] = rating

        # 리뷰
        review = int(re.findall('₩d+',
                                re.sub(',','',
                                       driver.find_element_by_css_selector('.hqzQac > span > a > span').text))[0])
        store_info['review'] = review

        # 음식 종류
        try:
            category = driver.find_element_by_css_selector('div:nth-child(2) > div:nth-child(2) > div > span:nth-child(2)').text
            store_info['category'] = category
        except:
            category = driver.find_element_by_css_selector('.kp-header > div > div:nth-child(2) > div:nth-child(2) > div > span').text
            store_info['category'] = category

        dict[base + i] = store_info

    except:
        continue

# 선 평점, 후 리뷰 정렬 데이터 프레임
df = pd.DataFrame.from_dict(dict,'index').sort_values(['rating', 'review'],
                                                      ascending=[False,False]).reset_index(drop=True)

# 중복 행 제거
df = df.drop_duplicates(['name','rating','review','category'])

# csv로 저장
df.to_csv(f"구글 {keyword}.csv")

pd.set_option('display.max_rows', None) # 전체 행 보기
pd.set_option('display.max_columns', None) # 전체 열 보기

df
```

```
  0%|          | 0/15 [00:00<?, ?it/s]
```

Out[6]:

	name	rating	review	category
0	영진어묵 본점	5.0	6	식품가공업체
1	다무치아	5.0	3	음식점
2	부산식당	5.0	2	한식당
3	산수맛집	5.0	1	음식점
4	자성화맛집코다리네부산수정점	5.0	1	음식점
5	대교식당	5.0	1	한식당
6	캠퍼스더큰닭	5.0	1	음식점
7	강가면옥	5.0	1	음식점

In [3]:
```
stores = pd.read_csv('구글 부산광역시 맛집.csv').reset_index(drop=True).drop('Unnamed: 0',axis=1)
print(stores.shape)
```

```
(261, 4)
```

데이터 전처리

In [119]:
```
import pandas as pd


# 망고플레이트에서 크롤링한 부산 맛집 리스트
mango_konlpy = pd.read_csv('mango_revise.csv').drop('Unnamed: 0', axis=1) # csv 파일 읽어 오기
mango_konlpy.head()
```

Out[119]:

	Title	Point	Review	View	Star	Type	Site	weighted_rating
0	비비비당	4.6	74	136578	3782	카페 디저트 오늘의 차 우전녹차 특말차 복분자 냉 오미자차	mango	4.485213
1	신발원	4.5	166	259881	5903	딤섬 만두	mango	4.452742
2	해목	4.5	157	324680	5477	정통 일식 일반 일식 특히츠마부시 민물장어덮밥 특카이센동 해산물덮밥 카이센동 해...	mango	4.450339
3	할매국밥	4.5	88	128996	2812	탕 찌개 전골	mango	4.418611
4	톤쇼우	4.6	38	26176	867	까스 요리	mango	4.413966

In [120]:
```
from konlpy.tag import Okt

main_pos = []

# 망고플레이트 부산 맛집의 음식 타입을 품사 태깅
for sentence in mango_konlpy.Type:

    pos = Okt().pos(sentence)

    # 명사로 나눠서 담기, 가끔 '오늘의' 같이 조사가 붙은 명사가 있었기 때문에 품사 태깅함
    main_words = [word_pos[0] for word_pos in pos if word_pos[1] in ("Noun")]

    # 제외 처리
    stopwords = ['이재', '오늘', '일반', '요리', '기타', '세트', '달인', '모듬', '평일', '런치', '성인', '디너', '주말', '매일']
    main_words = [t for t in main_words if t not in stopwords]

    # 공백으로 연결
    main_words_str = ' '.join(main_words)
    main_pos.append(main_words_str)

# 품사 태깅한 열 7번째에 추가
mango_konlpy.insert(6, 'main_pos_stem', main_pos, True)
mango_konlpy.head()
```

Out[120]:

	Title	Point	Review	View	Star	Type	main_pos_stem	Site	weighted_rating
0	비비비당	4.6	74	136578	3782	카페 디저트 오늘의 차 우전녹차 특말차 복분자 냉 오미자차	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차	mango	4.485213
1	신발원	4.5	166	259881	5903	딤섬 만두	딤섬 만두	mango	4.452742
2	해목	4.5	157	324680	5477	정통 일식 일반 일식 특히츠마부시 민물장어덮밥 특카이센동 해산물덮밥 카이센동 해...	정통 일식 일식 츠마 부시 민물장어 밥 특 카이센동 해산물 밥 카이센동 해산물 밥 ...	mango	4.450339
3	할매국밥	4.5	88	128996	2812	탕 찌개 전골	탕 찌개 전골	mango	4.418611
4	톤쇼우	4.6	38	26176	867	까스 요리	까스	mango	4.413966

In [121]:
```
# 잘못 분리된 문자 치환
mango_konlpy.main_pos_stem = mango_konlpy.main_pos_stem.str.replace('왕만', '왕만두')
mango_konlpy.main_pos_stem = mango_konlpy.main_pos_stem.str.replace('껍데기', ' 껍데기')
mango_konlpy.head()
```

Out[121]:

	Title	Point	Review	View	Star	Type	main_pos_stem	Site	weighted_rating
0	비비비당	4.6	74	136578	3782	카페 디저트 오늘의 차 우전녹차 특말차 복분자 냉 오미자차	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차	mango	4.485213
1	신발원	4.5	166	259881	5903	딤섬 만두	딤섬 만두	mango	4.452742
2	해목	4.5	157	324680	5477	정통 일식 일반 일식 특히츠마부시 민물장어덮밥 특카이센동 해산물덮밥 카이센동 해...	정통 일식 일식 츠마 부시 민물장어 밥 특 카이센동 해산물 밥 카이센동 해산물 밥 ...	mango	4.450339
3	할매국밥	4.5	88	128996	2812	탕 찌개 전골	탕 찌개 전골	mango	4.418611
4	톤쇼우	4.6	38	26176	867	까스 요리	까스	mango	4.413966

In [122]:
```
# 필요한 열 추출
mango_konlpy = mango_konlpy[['Title','Point','Review','weighted_rating','View','Star','main_pos_stem','Site']]
mango_konlpy.head()
```

Out[122]:

	Title	Point	Review	weighted_rating	View	Star	main_pos_stem	Site
0	비비비당	4.6	74	4.485213	136578	3782	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차	mango
1	신발원	4.5	166	4.452742	259881	5903	딤섬 만두	mango
2	해목	4.5	157	4.450339	324680	5477	정통 일식 일식 츠마 부시 민물장어 밥 특 카이센동 해산물 밥 카이센동 해산물 밥 ...	mango
3	할매국밥	4.5	88	4.418611	128996	2812	탕 찌개 전골	mango
4	톤쇼우	4.6	38	4.413966	26176	867	까스	mango

In [123]:
```
# 열 이름 변경
mango_konlpy.rename(columns={'main_pos_stem':'Type'}, inplace=True)
mango_konlpy.head()
```

Out[123]:

	Title	Point	Review	weighted_rating	View	Star	Type	Site
0	비비비당	4.6	74	4.485213	136578	3782	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차	mango
1	신발원	4.5	166	4.452742	259881	5903	딤섬 만두	mango
2	해목	4.5	157	4.450339	324680	5477	정통 일식 일식 츠마 부시 민물장어 밥 특 카이센동 해산물 밥 카이센동 해산물 밥 ...	mango
3	할매국밥	4.5	88	4.418611	128996	2812	탕 찌개 전골	mango
4	톤쇼우	4.6	38	4.413966	26176	867	까스	mango

In [124]:
```
mango_konlpy.to_csv('mango_konlpy.csv') # csv 파일로 저장
```

가중평점

In [11]:
```
import pandas as pd

g_stores = pd.read_csv('google_stores.csv').drop('Unnamed: 0',axis=1)
g_stores.head(3)
```

Out[11]:

	Title	Point	Review	Type	Site
0	영진어묵 본점	5.0	6	식품가공업체	google
1	다무치아	5.0	3	음식점	google
2	부산식당	5.0	2	한식당	google

In [12]:
```
# 고정값
C = g_stores['Point'].mean() # 전체 평균 평점
m = g_stores['Review'].quantile(0.6) # 최소 리뷰 수 : 상위 60%
```

In [13]:
```
# 변동값
def weighted_rating(record):
    v = record['Review'] # 가게 별 리뷰 수
    R = record['Point'] # 가게 별 평균 평점

    return ((v/(v+m))*R) + ((m/(m+v))*C)
```

In [14]:
```
# 가중평점 적용하고 정렬 (이하 동일한 방식으로 적용)
g_stores['weighted_rating'] = g_stores.apply(weighted_rating, axis=1)
g_stores = g_stores.sort_values(by=["weighted_rating"], ascending=False)
g_stores.head(3)
```

Out[14]:

	Title	Point	Review	Type	Site	weighted_rating
16	영진돼지국밥	4.5	1999	돼지국밥 전문점	google	4.456327
17	수변최고돼지국밥 본점	4.5	1222	돼지국밥 전문점	google	4.432497
26	이재모 피자	4.4	2724	피자 전문점	google	4.374052

협업 필터링

CF-KNN(item)

In [526]:
```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore') # 경고 무시

# 사이트별 맛집 목록 읽어오기
mango = pd.read_csv('mango_konlpy.csv').drop('Unnamed: 0', axis=1)
kakao = pd.read_csv('kakao_revise.csv').drop('Unnamed: 0', axis=1)
google = pd.read_csv('goo_revise.csv').drop('Unnamed: 0', axis=1)
naver = pd.read_csv('naver_revise.csv').drop('Unnamed: 0', axis=1)
tripadvisor = pd.read_csv('Tripadvisor_revise.csv').drop('Unnamed: 0', axis=1)
```

- site(사이트) = userId(사용자)
- title(상호명) = movieId & title
- point(가중평점) = rating(평점)
- type(음식 종류) = genres

In [527]:
```
# '공공데이터포털' 가게 음식 종류 데이터
stores = pd.read_csv('busan.csv').drop('Unnamed: 0', axis=1)

print(stores.shape)
stores.head()
```

```
(141878, 2)
```

Out[527]:

	title	type
0	미광장	모텔 여관 여인숙
1	BHC치킨	후라이드 양념치킨
2	스텔라	의류
3	도란도란	한식 백반 한정식
4	럭키사진관	사진관

In [ ]:
```

```

리뷰 10개 이상인 평점

In [528]:
```
print(mango.shape) # 모든 가게 목록
mango = mango[mango.Review >= 10]
print(mango.shape) # 리뷰 10개 이상 가게 목록
mango.head(3)
```

```
(200, 8)
(152, 8)
```

Out[528]:

	Title	Point	Review	weighted_rating	View	Star	Type	Site
0	비비비당	4.6	74	4.485213	136578	3782	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차	mango
1	신발원	4.5	166	4.452742	259881	5903	딤섬 만두	mango
2	해목	4.5	157	4.450339	324680	5477	정통 일식 일식 츠마 부시 민물장어 밥 특 카이센동 해산물 밥 카이센동 해산물 밥 ...	mango

In [529]:
```
print(kakao.shape)
kakao = kakao[kakao.Review >= 10]
print(kakao.shape)
kakao.head(3)
```

```
(370, 6)
(339, 6)
```

Out[529]:

	Title	Point	Review	Type	Site	weighted_rating
0	홍유단 남포본점	4.7	224	중화요리	kakao	4.501401
1	겐짱카레	4.6	142	일식	kakao	4.337854
2	프랭클린커피로스터스	4.8	64	커피전문점	kakao	4.267306

In [530]:
```
print(google.shape)
google = google[google.Review >= 10]
print(google.shape)
google.head(3)
```

```
(261, 6)
(216, 6)
```

Out[530]:

	Title	Point	Review	Type	Site	weighted_rating
0	영진돼지국밥	4.5	1999	돼지국밥 전문점	google	4.456327
1	수변최고돼지국밥 본점	4.5	1222	돼지국밥 전문점	google	4.432497
2	이재모 피자	4.4	2724	피자 전문점	google	4.374052

In [531]:
```
naver['Site'] = 'naver'
print(naver.shape)
naver = naver[naver.review >= 10]
print(naver.shape)
naver.head(3)
```

```
(287, 6)
(287, 6)
```

Out[531]:

	title	rating	review	save	weighted_rating	Site
0	해운대 달인막창	4.82	6056	17,000+	4.772242	naver
1	뚜벅스 광안점	4.83	1469	8,000+	4.685803	naver
2	오후의홍차	4.71	4282	15,000+	4.664902	naver

In [532]:
```
print(tripadvisor.shape)
tripadvisor = tripadvisor[tripadvisor.Review >= 10]
print(tripadvisor.shape)
tripadvisor.head(3)
```

```
(210, 5)
(132, 5)
```

Out[532]:

	Title	Review	Rate	site	weighted_rating
0	장수삼	184	5.0	tripadvisor	4.909958
1	고릴라브루잉	108	5.0	tripadvisor	4.858272
2	복순도가F1963	35	5.0	tripadvisor	4.684095

사용자-아이템 평점 행렬로 변환

필요한 열 추출

In [533]:
```
mango = mango[['Site', 'Title', 'weighted_rating']]
mango.head(3)
```

Out[533]:

	Site	Title	weighted_rating
0	mango	비비비당	4.485213
1	mango	신발원	4.452742
2	mango	해목	4.450339

In [534]:
```
kakao = kakao[['Site', 'Title', 'weighted_rating']]
kakao.head(3)
```

Out[534]:

	Site	Title	weighted_rating
0	kakao	홍유단 남포본점	4.501401
1	kakao	겐짱카레	4.337854
2	kakao	프랭클린커피로스터스	4.267306

In [535]:
```
google = google[['Site', 'Title', 'weighted_rating']]
google.head(3)
```

Out[535]:

	Site	Title	weighted_rating
0	google	영진돼지국밥	4.456327
1	google	수변최고돼지국밥 본점	4.432497
2	google	이재모 피자	4.374052

In [536]:
```
naver = naver[['Site', 'title', 'weighted_rating']]
naver.head(3)
```

Out[536]:

	Site	title	weighted_rating
0	naver	해운대 달인막창	4.772242
1	naver	뚜벅스 광안점	4.685803
2	naver	오후의홍차	4.664902

In [537]:
```
tripadvisor = tripadvisor[['site', 'Title', 'weighted_rating']]
tripadvisor.head(3)
```

Out[537]:

	site	Title	weighted_rating
0	tripadvisor	장수삼	4.909958
1	tripadvisor	고릴라브루잉	4.858272
2	tripadvisor	복순도가F1963	4.684095

In [538]:
```
# 열 이름 변경하여 통일
naver.rename(columns={'title':'Title'}, inplace=True)
naver.head(3)
```

Out[538]:

	Site	Title	weighted_rating
0	naver	해운대 달인막창	4.772242
1	naver	뚜벅스 광안점	4.685803
2	naver	오후의홍차	4.664902

In [539]:
```
tripadvisor.rename(columns={'site':'Site'}, inplace=True)
tripadvisor.head(3)
```

Out[539]:

	Site	Title	weighted_rating
0	tripadvisor	장수삼	4.909958
1	tripadvisor	고릴라브루잉	4.858272
2	tripadvisor	복순도가F1963	4.684095

데이터 합치기

In [540]:
```
ratings = pd.concat([mango, kakao, google, naver, tripadvisor]).reset_index(drop=True)

# 열 이름 변경
ratings.rename(columns={'Site':'userId', 'Title':'title', 'weighted_rating':'rating'}, inplace=True)
print(ratings.shape)
ratings.head(3)
```

(1126, 3)

Out[540]:

	userId	title	rating
0	mango	비비비당	4.485213
1	mango	신발원	4.452742
2	mango	해목	4.450339

In [541]:
```
# pivot_table 메소드를 사용해서 행렬 변환
ratings_matrix = ratings.pivot_table('rating', index='userId', columns='title')
ratings_matrix
```

Out[541]:

title	(주)대한민국맛집	(주)원조개금밀면	168도시락국{168계단도시락국}	18번완당집	1984나폴리	303화덕	33게이트	3found	50년전통할매국밥	Bumbu Bali 1	...	홍유단남포본점	홍콩반점 0410 부산역점	화교대반점	화국반점	화국반점(華國飯店)	황산밀면집	흙시루	희와제
userId																			
google	4.037789	3.913688	4.007687	NaN	NaN	NaN	NaN	NaN	4.184727	NaN	...	NaN	4.009598	3.946265	NaN	3.688144	3.962069	NaN	Na
kakao	NaN	NaN	NaN	3.336509	3.479711	2.919558	NaN	NaN	NaN	NaN	...	4.501401	NaN	NaN	3.046711	NaN	NaN	3.04516	3.3864
mango	NaN	NaN	NaN	NaN	4.029512	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	3.908542	NaN	NaN	NaN	Na
naver	NaN	NaN	NaN	4.386620	NaN	NaN	4.488892	4.597138	4.376229	NaN	...	NaN	NaN	NaN	4.314576	NaN	NaN	NaN	4.4781
tripadvisor	NaN	NaN	NaN	3.861037	NaN	NaN	NaN	NaN	NaN	4.498204	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 783 columns

In [542]:
```
# type이 들어있는 데이터랑 합치기
ratings_stores = pd.merge(ratings, stores, on='title')

# 중복 행 제거
ratings_stores = ratings_stores.drop_duplicates(['userId','title']).reset_index(drop=True)
print(ratings_stores.shape)
ratings_stores.head(4)
```

(549, 4)

Out[542]:

	userId	title	rating	type
0	mango	신발원	4.452742	중국음식 중국집
1	kakao	신발원	3.826330	중국음식 중국집
2	naver	신발원	4.458244	중국음식 중국집
3	tripadvisor	신발원	4.086166	중국음식 중국집

In [543]:
```
# 다시 columns='title' 로 title 컬럼으로 pivot 수행 # 사용자-아이템 평점 행렬
ratings_matrix = ratings_stores.pivot_table('rating', index='userId', columns='title')
ratings_matrix
```

Out[543]:

title	18번완당집	1984나폴리	303화덕	33게이트	가미	가야밀면	가야포차 선지국밥	갈비곳간	갈삼구이	개미집	...	헤이든	호랑이젤라떡	홍가네양곱창	홍성방	홍소족발	홍유단	흙시루	희와제
userId																			
google	NaN	NaN	NaN	NaN	NaN	NaN	4.179804	NaN	NaN	NaN	...	NaN	NaN	NaN	3.731405	NaN	NaN	NaN	Na
kakao	3.336509	3.479711	2.919558	NaN	NaN	NaN	NaN	3.926047	3.660968	NaN	...	3.036774	3.625340	NaN	NaN	NaN	3.460994	3.04516	3.3864
mango	NaN	4.029512	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
naver	4.386620	NaN	NaN	4.488892	NaN	NaN	NaN	NaN	4.568980	NaN	...	4.410330	4.440429	NaN	NaN	NaN	NaN	NaN	4.4781
tripadvisor	3.861037	NaN	NaN	NaN	4.454513	4.314555	NaN	NaN	NaN	4.019745	...	NaN	NaN	4.324472	NaN	4.163815	NaN	NaN	Na

5 rows × 358 columns

In [544]:
```
# NaN 값을 모두 0 으로 변환
ratings_matrix = ratings_matrix.fillna(0)
ratings_matrix
```

Out[544]:

title	18번완당집	1984나폴리	303화덕	33게이트	가미	가야밀면	가야포차 선지국밥	갈비곳간	갈삼구이	개미집	...	헤이든	호랑이젤라떡	홍가네양곱창	홍성방	홍소족발	홍유단	흙시루	희와제
userId																			
google	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.179804	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	3.731405	0.000000	0.000000	0.00000	0.0000
kakao	3.336509	3.479711	2.919558	0.000000	0.000000	0.000000	0.000000	3.926047	3.660968	0.000000	...	3.036774	3.625340	0.000000	0.000000	0.000000	3.460994	3.04516	3.3864
mango	0.000000	4.029512	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.0000
naver	4.386620	0.000000	0.000000	4.488892	0.000000	0.000000	0.000000	0.000000	4.568980	0.000000	...	4.410330	4.440429	0.000000	0.000000	0.000000	0.000000	0.00000	4.4781
tripadvisor	3.861037	0.000000	0.000000	0.000000	4.454513	4.314555	0.000000	0.000000	0.000000	4.019745	...	0.000000	0.000000	4.324472	0.000000	4.163815	0.000000	0.00000	0.0000

5 rows × 358 columns

In [ ]:

In [ ]:

가게 간 유사도 산출

In [545]:
```
# 사용자-아이템 평점 행렬로 transpose 한다.
ratings_matrix_T = ratings_matrix.transpose() # 전치 행렬

print(ratings_matrix_T.shape)
ratings_matrix_T.head(3)
```

```
(358, 5)
```

Out[545]:

userId	google	kakao	mango	naver	tripadvisor
title					
18번완당집	0.0	3.336509	0.000000	4.38662	3.861037
1984나폴리	0.0	3.479711	4.029512	0.00000	0.000000
303화덕	0.0	2.919558	0.000000	0.00000	0.000000

In [546]:
```
# 가게 간 코사인 유사도 산출
from sklearn.metrics.pairwise import cosine_similarity

item_sim = cosine_similarity(ratings_matrix_T, ratings_matrix_T)

# cosine_similarity() 로 반환된 넘파이 행렬을 가게명을 매핑하여 DataFrame으로 변환
item_sim_df = pd.DataFrame(data=item_sim,
                           index=ratings_matrix.columns,
                           columns=ratings_matrix.columns)

print(item_sim_df.shape)
item_sim_df.head(3)
```

(358, 358)

Out[546]:

title	18번완당집	1984나폴리	303화덕	33게이트	가미	가야밀면	가야포차선지국밥	갈비곳간	갈삼구이	개미집	...	헤이든	호랑이젤라떡	홍가네양곱창	홍성방	홍소족발	홍유단	흙시루	희와제과	흰여울비치	흰
title																					
18번완당집	1.000000	0.324064	0.495824	0.651877	0.573772	0.573772	0.0	0.495824	0.818754	0.573772	...	0.818101	0.818530	0.573772	0.0	0.573772	0.495824	0.495824	0.819012	0.651877	0.
1984나폴리	0.324064	1.000000	0.653585	0.000000	0.000000	0.000000	0.0	0.653585	0.408685	0.000000	...	0.370662	0.413346	0.000000	0.0	0.000000	0.653585	0.653585	0.394227	0.000000	0.
303화덕	0.495824	0.653585	1.000000	0.000000	0.000000	0.000000	0.0	1.000000	0.625297	0.000000	...	0.567121	0.632429	0.000000	0.0	0.000000	1.000000	1.000000	0.603177	0.000000	0.

3 rows × 358 columns

In [547]:
```
# 자기 것 빼고 유사 확인해보기
item_sim_df["마가만두"].sort_values(ascending=False)[1:10]
```

Out[547]:
```
title
고옥            0.876821
기장손칼국수        0.872831
뚱보집           0.869837
부산족발          0.867244
밀양순대돼지국밥      0.865658
금수복국          0.865542
포항돼지국밥        0.865330
신창국밥          0.861477
해운대원조할매국밥     0.861188
Name: 마가만두, dtype: float64
```

아이템 기반 인접 이웃 협업 필터링으로 개인화된 가게 추천

In [548]:
```
# 평점 벡터(행 벡터)와 유사도 벡터(열 벡터)를 내적(dot)해서 예측 평점을 계산하는 함수 정의
def predict_rating(ratings_arr, item_sim_arr):
    ratings_pred = ratings_arr.dot(item_sim_arr) / np.array([np.abs(item_sim_arr).sum(axis=1)])
    return ratings_pred
```

In [549]:
```
# 원본 데이터
ratings_matrix
```

Out[549]:

title	18번완당집	1984나폴리	303화덕	33게이트	가미	가야밀면	가야포차선지국밥	갈비곳간	갈삼구이	개미집	...	헤이든	호랑이젤라떡	홍가네양곱창	홍성방	홍소족발	홍유단	흙시루	희와제
userId																			
google	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.179804	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	3.731405	0.000000	0.000000	0.00000	0.0000
kakao	3.336509	3.479711	2.919558	0.000000	0.000000	0.000000	0.000000	3.926047	3.660968	0.000000	...	3.036774	3.625340	0.000000	0.000000	0.000000	3.460994	3.04516	3.3864
mango	0.000000	4.029512	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.0000
naver	4.386620	0.000000	0.000000	4.488892	0.000000	0.000000	0.000000	0.000000	4.568980	0.000000	...	4.410330	4.440429	0.000000	0.000000	0.000000	0.000000	0.00000	4.4781
tripadvisor	3.861037	0.000000	0.000000	0.000000	4.454513	4.314555	0.000000	0.000000	0.000000	4.019745	...	0.000000	0.000000	4.324472	0.000000	4.163815	0.000000	0.00000	0.0000

5 rows × 358 columns

In [550]:
```
# 예측 데이터
item_sim_df.head(3)
```

Out[550]:

title	18번완당집	1984나폴리	303화덕	33게이트	가미	가야밀면	가야포차선지국밥	갈비곳간	갈삼구이	개미집	...	헤이든	호랑이젤라떡	홍가네양곱창	홍성방	홍소족발	홍유단	흙시루	희와제과	흰여울비치	흰
title																					
18번완당집	1.000000	0.324064	0.495824	0.651877	0.573772	0.573772	0.0	0.495824	0.818754	0.573772	...	0.818101	0.818530	0.573772	0.0	0.573772	0.495824	0.495824	0.819012	0.651877	0.
1984나폴리	0.324064	1.000000	0.653585	0.000000	0.000000	0.000000	0.0	0.653585	0.408685	0.000000	...	0.370662	0.413346	0.000000	0.0	0.000000	0.653585	0.653585	0.394227	0.000000	0.
303화덕	0.495824	0.653585	1.000000	0.000000	0.000000	0.000000	0.0	1.000000	0.625297	0.000000	...	0.567121	0.632429	0.000000	0.0	0.000000	1.000000	1.000000	0.603177	0.000000	0.

3 rows × 358 columns

In [551]:
```
ratings_pred = predict_rating(ratings_matrix.values , item_sim_df.values)
ratings_pred
```

Out[551]:
```
array([[0.34895055, 0.34839453, 0.25765939, ..., 0.295689  , 0.32391849,
        0.32391849],
       [2.0307833 , 3.42988093, 3.56941125, ..., 2.25189576, 1.27390027,
        1.27390027],
       [1.01200027, 2.5919702 , 1.68406901, ..., 1.07508776, 0.62303925,
        0.62303925],
       [2.73801281, 1.26320568, 1.29780548, ..., 3.10814303, 4.45196173,
        4.45196173],
       [0.90436367, 0.36829389, 0.30572312, ..., 0.29050587, 0.27921006,
        0.27921006]])
```

In [552]:
```
# 데이터프레임으로 변환
ratings_pred_matrix = pd.DataFrame(data=ratings_pred,
                                   index= ratings_matrix.index,
                                   columns = ratings_matrix.columns)
print(ratings_pred_matrix.shape)
ratings_pred_matrix.head()
```

(5, 358)

Out[552]:

title	18번완당집	1984나폴리	303화덕	33게이트	가미	가야밀면	가야포차 선지국밥	갈비곳간	갈삼구이	개미집	...	헤이든	호랑이젤라떡	홍가네양곱창	홍성방	홍소족발	홍유단	흙시루	희와
userId																			
google	0.348951	0.348395	0.257659	0.323918	0.637443	0.637443	4.019480	0.257659	0.294748	0.637443	...	0.297196	0.294441	0.637443	4.019480	0.637443	0.257659	0.257659	0.295
kakao	2.030783	3.429881	3.569411	1.273900	0.817697	0.817697	0.315334	3.569411	2.284495	0.817697	...	2.199674	2.295119	0.817697	0.315334	0.817697	3.569411	3.569411	2.251
mango	1.012000	2.591970	1.684069	0.623039	0.662783	0.662783	0.320225	1.684069	1.090156	0.662783	...	1.050950	1.095066	0.662783	0.320225	0.662783	1.684069	1.684069	1.075
naver	2.738013	1.263206	1.297805	4.451962	0.760798	0.760798	0.403864	1.297805	3.063349	0.760798	...	3.179899	3.048752	0.760798	0.403864	0.760798	1.297805	1.297805	3.108
tripadvisor	0.904364	0.368294	0.305723	0.279210	4.223300	4.223300	0.291677	0.305723	0.290882	4.223300	...	0.289903	0.291005	4.223300	0.291677	4.223300	0.305723	0.305723	0.290

5 rows × 358 columns

-> 가게 별 예측평점이 나옴

예측 평점 정확도를 판단하기 위해 오차 함수인 RMSE를 이용

In [553]:
```
from sklearn.metrics import mean_squared_error

# 사이트(사용자)가 평점을 부여한 가게에 대해서만 예측 성능 평가 MSE 를 구함.
def get_mse(pred, actual):
    # Ignore nonzero terms
    pred = pred[actual.nonzero()].flatten()
    actual = actual[actual.nonzero()].flatten()
    return mean_squared_error(pred, actual)

print('아이템 기반 모든 인접 이웃 MSE: ', get_mse(ratings_pred, ratings_matrix.values ))
```

아이템 기반 모든 인접 이웃 MSE: 1.9425707060733026

top-n 유사도를 가진 데이터들에 대해서만 예측 평점 계산

In [554]:
```
def predict_rating_topsim(ratings_arr, item_sim_arr, top_n):
    # 사용자-아이템 평점 행렬 크기만큼 0으로 채운 예측 행렬 초기화
    pred = np.zeros(ratings_arr.shape)

    # 사용자-아이템 평점 행렬의 열 크기만큼 Loop 수행.
    for col in range(ratings_arr.shape[1]):
        # 유사도 행렬에서 유사도가 큰 순으로 n개 데이터 행렬의 index 반환
        top_n_items = [np.argsort(item_sim_arr[:, col])[:-top_n-1:-1]]
        # 개인화된 예측 평점을 계산
        for row in range(ratings_arr.shape[0]):
            pred[row, col] = item_sim_arr[col, :][tuple(top_n_items)].dot(ratings_arr[row, :][tuple(top_n_items)].T)
            pred[row, col] /= np.sum(np.abs(item_sim_arr[col, :][tuple(top_n_items)]))
    return pred
```

In [555]:
```
ratings_pred = predict_rating_topsim(ratings_matrix.values , item_sim_df.values, top_n=10)
print('아이템 기반 인접 TOP-10 이웃 MSE: ', get_mse(ratings_pred, ratings_matrix.values))
print('아이템 기반 인접 TOP-10 이웃 RMSE: ', np.sqrt(get_mse(ratings_pred, ratings_matrix.values)))

# 계산된 예측 평점 데이터는 DataFrame으로 재생성
ratings_pred_matrix = pd.DataFrame(data=ratings_pred, index= ratings_matrix.index,
                                   columns = ratings_matrix.columns)
ratings_pred_matrix
```

```
아이템 기반 인접 TOP-10 이웃 MSE:  0.10213119900537873
아이템 기반 인접 TOP-10 이웃 RMSE:  0.31957972245650806
```

Out[555]:

title	18번완당집	1984나폴리	303화덕	33게이트	가미	가야밀면	가야포차 선지국밥	갈비곳간	갈삼구이	개미집	...	헤이든	호랑이젤라떡	홍가네양곱창	홍성방	홍소족발	홍유단	흙시루	희와
userId																			
google	0.740059	0.000000	0.000000	0.000000	0.000000	0.000000	4.035621	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	4.035621	0.000000	0.000000	0.000000	0.000
kakao	3.033818	3.572645	3.545528	0.000000	0.000000	0.000000	0.000000	3.545528	3.611978	0.000000	...	3.310934	3.656053	0.000000	0.000000	0.000000	3.545528	3.545528	3.361
mango	1.195420	4.095866	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
naver	4.374396	0.000000	0.000000	4.436162	0.000000	0.000000	0.000000	0.000000	4.492069	0.000000	...	4.442095	4.488468	0.000000	0.000000	0.000000	0.000000	0.000000	4.449
tripadvisor	3.732981	0.000000	0.000000	0.000000	4.260291	4.260291	0.000000	0.000000	0.000000	4.260291	...	0.000000	0.000000	4.260291	0.000000	4.260291	0.000000	0.000000	0.000

5 rows × 358 columns

-> 최종적인 가게 별 예측 평점 데이터가 만들어짐

사용자(사이트)에게 가게 추천

In [556]:
```
# 사용자가 높은 평점을 준 가게 확인
user_rating_id = ratings_matrix.loc['mango']
user_rating_id[user_rating_id > 0].sort_values(ascending=False)[:5]
```

Out[556]:
```
title
신발원       4.452742
해목         4.450339
할매국밥      4.418611
톤쇼우       4.413966
블랙업커피     4.348947
Name: mango, dtype: float64
```

사용자가 이용하지 않은 가게 추천

user_rating이 0보다 크면 기존에 이용한 가게라는 점을 이용해서 계산

In [557]:
```
def get_unvisited_foods(ratings_matrix, userId):
    # userId로 입력받은 사용자의 모든 가게정보 추출하여 Series로 반환
    # 반환된 user_rating 은 가게명(title)을 index로 가지는 Series 객체
    user_rating = ratings_matrix.loc[userId,:]

    # user_rating이 0보다 크면 기존에 이용한 가게, 대상 index를 추출하여 list 객체로 만듬
    already_visited = user_rating[user_rating > 0].index.tolist()

    # 모든 가게명을 list 객체로 만듬
    foods_list = ratings_matrix.columns.tolist()

    # list comprehension으로 already_visited에 해당하는 food는 foods_list에서 제외
    unvisited_list = [ food for food in foods_list if food not in already_visited]

    return unvisited_list
```

In [558]:
```
# pred_df : 앞서 계산된 가게 별 예측 평점
# unvisited_list : 사용자가 이용하지 않은 가게들
# top_n : 상위 n개를 가져온다.

def recomm_food_by_userid(pred_df, userId, unvisited_list, top_n=10):
    # 예측 평점 DataFrame에서 사용자id index와 unvisited_list로 들어온 가게명 컬럼을 추출하여
    # 가장 예측 평점이 높은 순으로 정렬함.
    recomm_foods = pred_df.loc[userId, unvisited_list].sort_values(ascending=False)[:top_n]
    return recomm_foods
```

mango가 이용하지 않은 가게 예측 평점

In [559]:
```
# 사용자가 이용하지 않는 가게명 추출
unvisited_list = get_unvisited_foods(ratings_matrix, 'mango')

# 아이템 기반의 인접 이웃 협업 필터링으로 가게 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'mango', unvisited_list, top_n=5)

# 평점 데이터를 DataFrame으로 생성
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['pred_score'])
recomm_foods
```

Out[559]:

title	pred_score
영진돼지국밥	1.665547
초량밀면	1.585834
거인통닭	1.584969
해운대암소갈비집	1.195461
18번완당집	1.195420

In [560]:
```
ratings_matrix['초량밀면']
```

Out[560]:
```
userId
google        3.812421
kakao         3.197718
mango         0.000000
naver         4.330589
tripadvisor   4.044233
Name: 초량밀면, dtype: float64
```

kakao가 이용하지 않은 가게 예측 평점

In [561]:
```
# 사용자가 이용하지 않는 가게명 추출
unvisited_list = get_unvisited_foods(ratings_matrix, 'kakao')

# 아이템 기반의 인접 이웃 협업 필터링으로 가게 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'kakao', unvisited_list, top_n=10)

# 평점 데이터를 DataFrame으로 생성
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['pred_score'])
recomm_foods
```

Out[561]:

title	pred_score
원향재	2.188041
원산면옥	1.943213
다이도코로	1.547991
동해물회	1.547942
부산명물횟집	1.336239
옥성반점	1.256382
풍원장미역국정찬	1.068103
다이닝룸	1.068103
포르타나	1.068103
사보이	1.068103

In [562]:
```
ratings_matrix['부산명물횟집']
```

Out[562]:
```
userId
google        3.930067
kakao         0.000000
mango         0.000000
naver         4.363552
tripadvisor   4.136117
Name: 부산명물횟집, dtype: float64
```

In [ ]:

naver가 이용하지 않은 가게 예측 평점

In [563]:
```
# 사용자가 이용하지 않는 가게명 추출
unvisited_list = get_unvisited_foods(ratings_matrix, 'naver')

# 아이템 기반의 인접 이웃 협업 필터링으로 가게 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'naver', unvisited_list, top_n=10)

# 평점 데이터를 DataFrame으로 생성
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['pred_score'])
recomm_foods
```

Out[563]:

title	pred_score
마가만두	1.752543
짱떡볶이	1.645559
해운대원조할매국밥	1.644805
신창국밥	1.643834
스완양분식	1.643209
포항돼지국밥	1.205006
금수복국	1.204553
다리집	1.204096
밀양순대돼지국밥	1.203766
해성막창집	1.202516

In [564]:
```
ratings_matrix['마가만두']
```

Out[564]:
```
userId
google        4.087101
kakao         3.741779
mango         4.341966
naver         0.000000
tripadvisor   4.146751
Name: 마가만두, dtype: float64
```

In [ ]:

google이 이용하지 않은 가게 예측 평점

In [565]:
```
# 사용자가 이용하지 않는 가게명 추출
unvisited_list = get_unvisited_foods(ratings_matrix, 'google')

# 아이템 기반의 인접 이웃 협업 필터링으로 가게 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'google', unvisited_list, top_n=10)

# 평점 데이터를 DataFrame으로 생성
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['pred_score'])
recomm_foods
```

Out[565]:

title	pred_score
이가네떡볶이	1.557108
쌍둥이돼지국밥	1.555255
신발원	1.550371
원산면옥	1.109085
해운대암소갈비집	0.741411
고릴라브루잉	0.740896
18번완당집	0.740059
포항돼지국밥	0.385683
금수복국	0.385570
밀양순대돼지국밥	0.385342

In [566]:
```
ratings_matrix['이가네떡볶이']
```

Out[566]:
```
userId
google        0.000000
kakao         3.499880
mango         4.185909
naver         4.321205
tripadvisor   4.094869
Name: 이가네떡볶이, dtype: float64
```

In [ ]:

tripadvisor가 이용하지 않은 가게 예측 평점

In [567]:
```
# 사용자가 이용하지 않는 가게명 추출
unvisited_list = get_unvisited_foods(ratings_matrix, 'tripadvisor')

# 아이템 기반의 인접 이웃 협업 필터링으로 가게 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'tripadvisor', unvisited_list, top_n=10)

# 평점 데이터를 DataFrame으로 생성.
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['pred_score'])
recomm_foods
```

Out[567]:

title	pred_score
에쎄떼	1.588274
해목	1.587417
평산옥	1.585733
내호냉면	1.585706
동해물회	1.093314
다이도코로	1.093237
옥성반점	1.093166
영진돼지국밥	0.796664
짱떡볶이	0.394043
해운대원조할매국밥	0.393888

In [568]:
```
ratings_matrix['에쎄떼']
```

Out[568]:
```
userId
google        4.182932
kakao         3.563853
mango         4.007000
naver         4.493709
tripadvisor   0.000000
Name: 에쎄떼, dtype: float64
```

지도 시각화

In [53]:
```
import pandas as pd

# '공공데이터포털'에 있는 부산 상권 정보
busan = pd.read_csv('소상공인시장진흥공단_상가(상권)정보_부산_202112.csv')

# 열 이름 변경
busan.rename(columns={'상호명':'title'}, inplace=True)

# 상위 10개
print(busan.shape)
busan.head()
```

```
(141878, 39)
```

Out[53]:

	상가업소번호	title	지점명	상권업종대분류코드	상권업종대분류명	상권업종중분류코드	상권업종중분류명	상권업종소분류코드	상권업종소분류명	표준산업분류코드	...	건물관리번호	건물명	도로명주소	구우편번호	신우편번호	동정보	층정보	호정보	경도	위도
0	23206623	미광장	NaN	O	숙박	O02	모텔/여관/여인숙	O02A01	모텔/여관/여인숙	I55112	...	2611012600100250001004473	NaN	부산광역시 중구 보수대로44번길 5	600074	48974.0	NaN	NaN	NaN	129.023668	35.100979
1	20418637	BHC 치킨 동래점	Q	음식	Q05	닭/오리요리	Q05A08	후라이드/양념치킨	I56193	...	2626010500101850000016529	NaN	부산광역시 동래구 동래로147번길 18	607020	47802.0	NaN	NaN	NaN	129.087156	35.205267

In [54]:
```
# 망고플레이트 기준으로 다른 사이트에 리뷰가 있는 가게 리스트
mango = pd.read_csv('중복가게.csv').drop('Unnamed: 0', axis=1) # csv 파일 읽어 오기
print(mango.shape)
mango.head()
```

(379, 4)

Out[54]:

	userId	title	rating	type
0	mango	비비비당	4.485213	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차
1	kakao	비비비당	3.661010	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차
2	naver	비비비당	4.465512	카페 디저트 차 우전 녹차 특 말차 복분자 냉 오미자차
3	mango	신발원	4.452742	딤섬 만두
4	kakao	신발원	3.850168	딤섬 만두

In [55]:
```
# 상호명을 기준으로 '부산 상권 정보', '가게 리스트' 병합
ratings_stores = pd.merge(busan, mango, on='title')
print(ratings_stores.shape) # 행 갯수가 늘어난 이유는 상호명을 기준으로 하여 같은 가게인데 경도 위도가 다른 중복된 가게 목록이 있기 때문
ratings_stores.head()
```

(384, 42)

Out[55]:

	상가업소번호	title	지점명	상권업종대분류코드	상권업종대분류명	상권업종중분류코드	상권업종중분류명	상권업종소분류코드	상권업종소분류명	표준산업분류코드	...	구우편번호	신우편번호	동정보	층정보	호정보	경도	위도	userId	rating	type
0	28490295	모모스커피	NaN	Q	음식	Q12	커피점/카페	Q12A01	커피전문점/카페/다방	I56220	...	609320	46311.0	NaN	1	NaN	129.086376	35.219218	mango	4.020862	카페디저트
1	28490295	모모스커피	NaN	Q	음식	Q12	커피점/카페	Q12A01	커피전문점/카페/다방	I56220	...	609320	46311.0	NaN	1	NaN	129.086376	35.219218	kakao	4.099568	카페디저트
2	28504927	백설대학	NaN	Q	음식	Q04	분식	Q04A01	라면김밥분식	I56194	...	606080	49102.0	NaN	NaN	NaN	129.070072	35.079008	mango	4.040333	한식
3	28504927	백설대학	NaN	Q	음식	Q04	분식	Q04A01	라면김밥분식	I56194	...	606080	49102.0	NaN	NaN	NaN	129.070072	35.079008	kakao	3.179077	한식
4	28504927	백설대학	NaN	Q	음식	Q04	분식	Q04A01	라면김밥분식	I56194	...	606080	49102.0	NaN	NaN	NaN	129.070072	35.079008	naver	4.448316	한식

5 rows × 42 columns

In [56]:
```
# 필요한 열 추출
ratings_stores = ratings_stores[['title','위도','경도','상권업종소분류명','userId','rating']]

# 사이트 이름과 상호명을 기준으로 중복 행 제거
ratings_stores = ratings_stores.drop_duplicates(['userId','title']).reset_index(drop=True)

# 가중평점 소수점 둘째자리까지
ratings_stores.rating = ratings_stores.rating.round(2)

print(ratings_stores.shape)
ratings_stores.head()
```

```
(228, 6)
```

Out[56]:

	title	위도	경도	상권업종소분류명	userId	rating
0	모모스커피	35.219218	129.086376	커피전문점/카페/다방	mango	4.02
1	모모스커피	35.219218	129.086376	커피전문점/카페/다방	kakao	4.10
2	백설대학	35.079008	129.070072	라면김밥분식	mango	4.04
3	백설대학	35.079008	129.070072	라면김밥분식	kakao	3.18
4	백설대학	35.079008	129.070072	라면김밥분식	naver	4.45

In [57]:
```
# 사이트 이름과 가중평점을 연결하고 값을 문자열로 변환하여 열 추가
ratings_stores['user_rating'] = ratings_stores[['userId','rating']].apply(lambda row: ' : '.join(row.values.astype(str)), axis=1)
print(ratings_stores.shape)
ratings_stores.head()
```

(228, 7)

Out[57]:

	title	위도	경도	상권업종소분류명	userId	rating	user_rating
0	모모스커피	35.219218	129.086376	커피전문점/카페/다방	mango	4.02	mango : 4.02
1	모모스커피	35.219218	129.086376	커피전문점/카페/다방	kakao	4.10	kakao : 4.1
2	백설대학	35.079008	129.070072	라면김밥분식	mango	4.04	mango : 4.04
3	백설대학	35.079008	129.070072	라면김밥분식	kakao	3.18	kakao : 3.18
4	백설대학	35.079008	129.070072	라면김밥분식	naver	4.45	naver : 4.45

In [58]:
```
# 한 가게에 있는 사이트 가중평점을 하나로 연결
ratings_stores = ratings_stores.groupby(['title','위도','경도','상권업종소분류명',])['user_rating'].apply('  /  '.join).reset_index()
print(ratings_stores.shape)
ratings_stores.head()
```

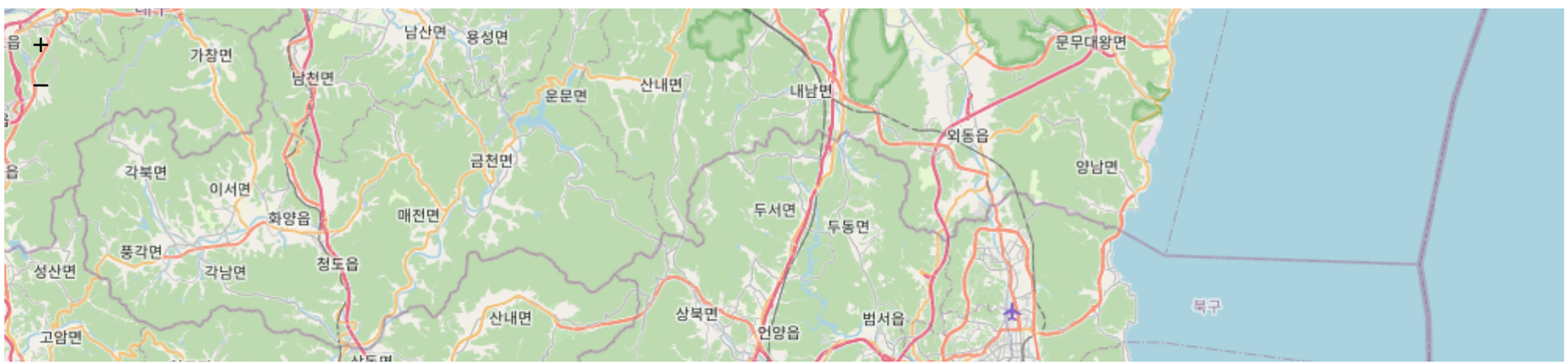
(89, 5)

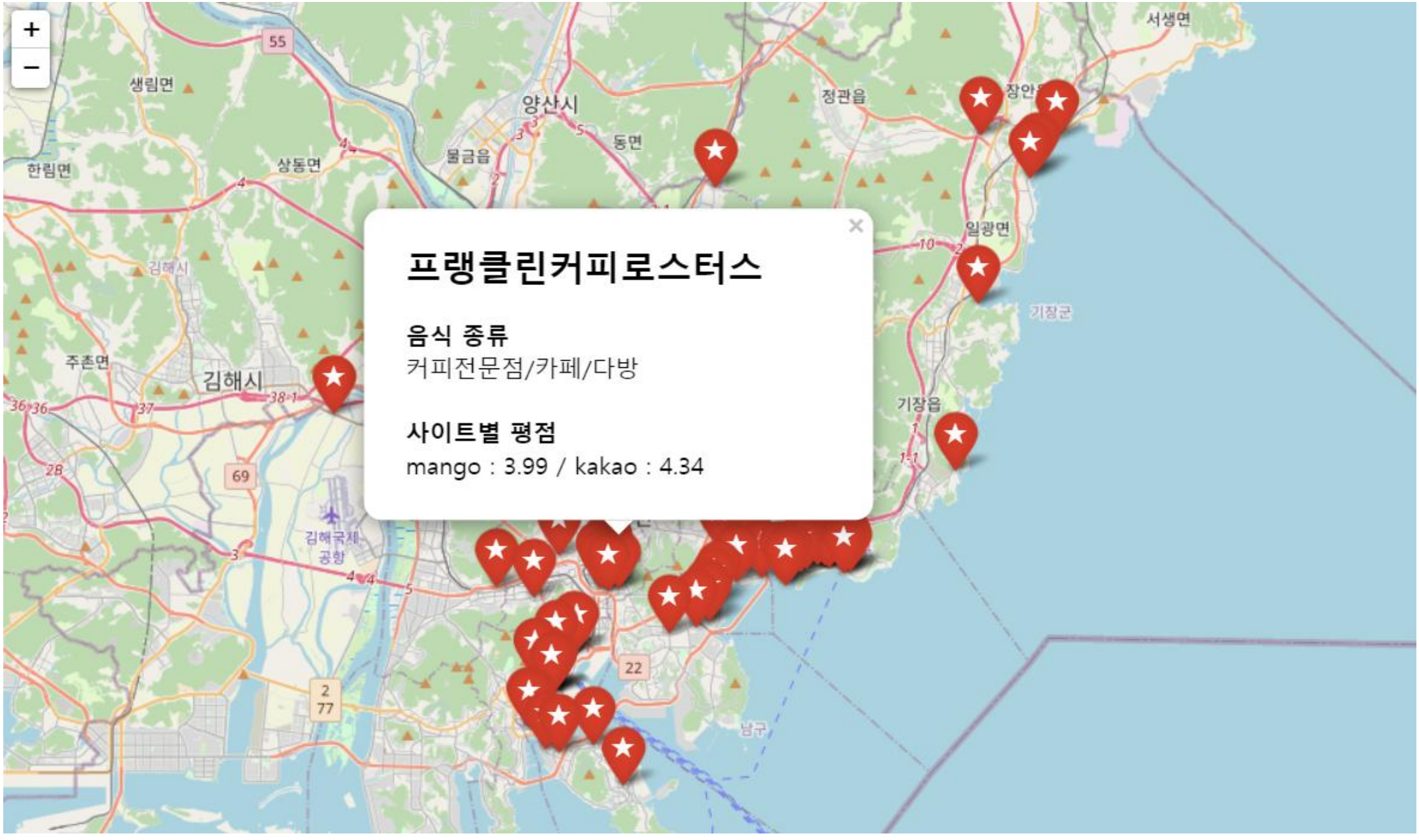
Out[58]:

	title	위도	경도	상권업종소분류명	user_rating
0	1984나폴리	35.304089	129.112414	커피전문점/카페/다방	mango : 4.03 / kakao : 3.56
1	거대곰탕	35.153023	129.059626	설렁탕집	mango : 4.07 / kakao : 3.98 / naver : 4.47
2	겐짱카레	35.100763	129.029007	음식점-일식	mango : 3.91 / kakao : 4.38
3	고래사	35.162510	129.159503	식료품점	mango : 4.01 / kakao : 3.7
4	고옥	35.310151	129.260781	한식/백반/한정식	mango : 4.33 / kakao : 3.67 / google : 4.1...

In [59]:
```
import folium

# 기준점 설정
m = folium.Map(location = [35.16067972581369, 129.12464303902127], # [위도, 경도]
               zoom_start = 11,
               width=1000,
               height=700)

for lat, long, store, category, point in zip(ratings_stores['위도'],
                                             ratings_stores['경도'],
                                             ratings_stores['title'], # 상호명
                                             ratings_stores['상권업종소분류명'], # 음식 카테고리
                                             ratings_stores['user_rating']): # 사이트별 가중평점
    # HTML 코드 작성
    iframe = folium.IFrame("<h2><b>" + store + "</h2></b>"+ "<b>음식 종류</b>" + "<br>" + category + "<br>" + "<br>" + "<b>사이트별 평점</b>" + "<br>" + point)

    # 팝업 내용과 크기 설정
    popup = folium.Popup(iframe,
                         min_width=300,
                         max_width=300)

    folium.Marker(location = [lat, long],
                  tooltip=store, # 커서를 갖다 대면 뜨는 내용
                  popup=popup, # 클릭하면 나오는 팝업
                  icon=folium.Icon('red', icon='star') # 표시되는 아이콘 모양 설정
                 ).add_to(m)
m
```

Out[59]:




+
−
55
생림면
양산시
정관읍
장안
서생면
한림면
상동면
물금읍
동면
일광면
김해시
기장군
주촌면
김해시
기장읍
69
김해국제
공항
22
2
77
남구
프랭클린커피로스터스
음식 종류
커피전문점/카페/다방
사이트별 평점
mango : 3.99 / kakao : 4.34

PART 1

수집

망고플레이트

부산 맛집

200

20개 X 10페이지

PART 1

수집

사용된 코드

스크롤 : driver.execute_script("window.scrollTo(0, 400)")

뒤로 가기 : driver.back()

쉬기 : time.sleep(0.5)

창 끄기 : driver.close()

창 띄우기 : chromedriver_autoinstaller.install(), webdriver.Chrome()

특정 페이지 띄우기 : driver.get(" https://www.mangoplate.com/search/{0}?keyword={0}&page={1}".format(keyword, page+1))

창 최대화 : maximize_window()

클릭하기 : click()

Iframe에 접근 : driver.switch_to.frame()

가져오기 : driver.find_element_by_css_selector(element)

텍스트 뽑기 : .text

PART 2

정제

Type 열
불필요한 문자 제거

```
data['Type'].str.findall('[가-힣]+’)

data['Type'].apply(lambda x : (' ').join(x))

data['Type'].str.replace('원', '')
```

View 열 정제
숫자형으로 변환

```
data['View'].str.replace(',', ‘’)

data['View'].astype(int)
```

Star 열 정제
숫자형으로 변환

```
[str(x) for x in data['Star’]]

data['Star'].str.replace(',', ‘’)

data['Star'].astype(int)
```

PART 3

모델 : 1개 사이트(망고)

CBF 추천 알고리즘

5.1 CountVectorizer로 학습

CountVectorizer로 학습 : ngram_range=(1,2)

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect1 = CountVectorizer(min_df=1, ngram_range=(1, 2))  # min_df: 단어장에 들어갈 최소빈도, ngram_range: 1 <= n <= 2
type_mat1 = count_vect1.fit_transform(data['Type'])
print(type_mat1.shape)
print(type_mat1[:5])

# 200개 가게에 대한 307개 유형의 '유형 매트릭스'가 생성되었다.
```

```
(200, 307)
  (0, 236)      1
  (0, 36)       1
  (0, 154)      1
  (0, 178)      1
  (0, 257)      1
```

모델

추천 ver1: 코사인 유사도 이용해서 Type이 유사한 가게 추천

6.0.1 가게 '비비비당'에 대해 유형 유사성, 가중평점 반영한 추천 가게 10개를 뽑아보자

```
similar_foods = find_sim_food_ver4(data, type_sim_sorted_ind, '비비비당', top_n=10)
similar_foods.iloc[:, 1:]
```

	Title	Point	Review	weighted_rating	View	Star	Type	Site
29	보성녹차	4.2	44	4.156406	44856	1487	카페 디저트	mango
38	흰여울비치	4.3	7	4.122593	2323	66	카페 디저트	mango
42	샬롯	4.2	13	4.115455	11922	430	카페 디저트	mango
50	구프	4.2	7	4.096667	2924	145	카페 디저트	mango
51	블랙업커피	4.1	95	4.093130	83681	2278	카페 디저트	mango
129	모모스커피	4.0	38	4.020862	21292	544	카페 디저트	mango
131	카페38.5	3.9	7	4.018889	1405	20	카페 디저트	mango
136	어라이크커피	3.9	8	4.014643	8603	290	카페 디저트	mango
143	어바웃제이	3.9	9	4.010690	6251	141	카페 디저트	mango
144	쿠루미과자점	3.9	9	4.010690	4313	120	카페 디저트	mango

PART 3

모 델

추천 ver1: 코사인 유사도 이용해서 Type이 유사한 가게 추천

6.0.4 가게 '이재모피자'에 대해 유형 유사성, 가중평점 반영한 추천 가게 10개를 뽑아보자

```
similar_foods = find_sim_food_ver4(data, type_sim_sorted_ind, '이재모피자', top_n=10)
similar_foods.iloc[:, 1:]
```

	Title	Point	Review	weighted_rating	View	Star	Type	Site
15	포르타나	4.5	13	4.233636	25505	608	양식 마르 게리 후라이드 치킨	mango
31	버거샵	4.3	11	4.145484	3621	127	브런치 버거 샌드위치 클래식 버거 치즈버거 베이컨 치즈 감자 튀김 하리 토스	mango
59	빨간떡볶이	4.1	34	4.085370	66290	1580	한식	mango
60	수변최고돼지국밥	4.1	30	4.084200	18581	498	탕 찌개 전골	mango
61	슈발츠발트	4.1	27	4.083191	37976	972	카페 디저트 아메리카노 플랫 화이트 말차 프레 소 레몬 레이어 케이크	mango
63	이너프	4.1	23	4.081628	56636	1924	카페 디저트	mango
64	본가제일면가	4.1	23	4.081628	21333	552	국수 면 물 밀면 비빔면 회 밀면 곡 밀면 물 왕만두 알	mango
65	봉샌드	4.1	21	4.080732	18231	452	브런치 버거 샌드위치	mango
66	김유순대구뽈찜전문집	4.1	20	4.080250	7997	223	해산물	mango
67	기와집대구탕	4.1	18	4.079211	21836	536	해산물	mango

PART 3

모델

추천 ver2 :먼저 Type이 유사한 20개 선정 후 가중평점(1) 순 10 가게

5.5.1 가게 '비비비당'에 대해 유형 유사성, 가중평점 반영한 추천 가게 10개를 뽑아보자

```
similar_foods = find_sim_food_ver2(data, type_sim_sorted_ind, '비비비당', top_n=10)
similar_foods
```

	Title	Point	View	Review	Star	Type	weighted_rate
53	블랙업커피	0.375	0.257446	0.387234	0.36416	카페 디저트	1.383840
38	보성녹차	0.500	0.137820	0.170213	0.23760	카페 디저트	1.045633
62	이너프	0.375	0.174116	0.080851	0.30752	카페 디저트	0.937487
25	흰여울비치	0.625	0.006769	0.012766	0.01024	카페 디저트	0.654775
43	샬롯	0.500	0.036345	0.038298	0.06848	카페 디저트	0.643123
41	구프	0.500	0.008621	0.012766	0.02288	카페 디저트	0.544267
138	손목서가	0.125	0.061001	0.068085	0.08448	카페 디저트	0.338566
108	Cafe de 220VOLT	0.250	0.019220	0.021277	0.02480	카페 디저트	0.315297
117	카페덕미	0.250	0.003756	0.008511	0.00576	카페 디저트	0.268027
166	그릿비 일광	0.125	0.047749	0.029787	0.06352	카페 디저트	0.266056

모 델

가중평점(1) 계산

```
# 1) MinMaxScale를 통해 0~1로 값의 규모를 통일시킨다.
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data[['Point', 'View', 'Review', 'Star']] = scaler.fit_transform(data[['Point', 'View', 'Review', 'Star’]])


# 2) 행 기준으로 점수를 더한다.
data['weighted_rate'] = data['Point'] + data['View'] + data['Review'] + data['Star']
```

모델

추천 ver3 : 먼저 Type이 유사한 20개 선정 후 가중평점(2) 순 10 가게 추천

5.6.1 가게 '비비비당'에 대해 유형 유사성, 가중평점(2) 반영한 추천 가게 10개를 뽑아보자

```
similar_foods = find_sim_food_ver3(data, type_sim_sorted_ind, '비비비당', top_n=10)
similar_foods
```

	Title	Point	View	Review	Star	Type	weighted_rate	weighted_rate2
38	보성녹차	0.500	0.137820	0.170213	0.23760	카페 디저트	1.045633	4.501786
43	샬롯	0.500	0.036345	0.038298	0.06848	카페 디저트	0.643123	3.884000
25	흰여울비치	0.625	0.006769	0.012766	0.01024	카페 디저트	0.654775	3.728947
53	블랙업커피	0.375	0.257446	0.387234	0.36416	카페 디저트	1.383840	3.676168
41	구프	0.500	0.008621	0.012766	0.02288	카페 디저트	0.544267	3.531579
62	이너프	0.375	0.174116	0.080851	0.30752	카페 디저트	0.937487	3.524286
115	수월경화	0.250	0.002884	0.004255	0.00640	카페 디저트	0.263539	3.211765
117	카페덕미	0.250	0.003756	0.008511	0.00576	카페 디저트	0.268027	3.172222
108	Cafe de 220VOLT	0.250	0.019220	0.021277	0.02480	카페 디저트	0.315297	3.076190
148	어라이크커피	0.125	0.026119	0.017021	0.04608	카페 디저트	0.214220	2.855000

모 델

가중평점(2) 계산

```
def weighted_rate_average(record):
    v = record['Review’] # 가게별 리뷰 개수
    R = record['Point’] # 개별 가게에 대한 평점
    return ( (v/(v+m)) * R ) + ( (m/(m+v)) * C )

C = data[ ' Point ' ].mean() # 전체 가게에 대한 평균 평점
m = data['Review'].quantile(0.6) # 리뷰 개수 상위 60%

data['weighted_rate2'] = data.apply(weighted_rate_average, axis=1)
data['weighted_rate2'] = data['weighted_rate2'] * 10 # 이미 0~1사이 이므로 보기 쉽게 10을 곱함
```

PART 1

수집

Tripadvisor

트립어드바이저

부산 맛집

210

30개 X 7페이지

PART 1

수집

사용된 코드(망고플레이트에서 사용된 코드는 제외)

- alt 속성 가져오기 : .get_attribute('alt’)
- class 이름 가져오기 : driver.find_elements_by_class_name()

PART 2

정제

Rate 열

불필요한 문자 제거

```
rate_list = []
for x in df_all['Rate']:
    y = x[8:]
    rate_list.append(y)

df_all['Rate'] = rate_list
```

Review 열

불필요한 문자 제거

```
review_list = []
for x in df_all['Review']:
    y = x[:-5]
    review_list.append(y)

df_all['Review'] = review_list
```

PART 3

모델 : 5개 사이트(망고, 구글, 카카오, 네이버, 트립)

CF 추천 알고리즘 - MF

2.1 google가 이용하지 않은 가게 예측 평점

```
# 사용자가 이용하지 않는 가게명 추출
unseen_list = get_unseen_foods(ratings_matrix, 'google')

# 아이템 기반의 인접 이웃 협업 필터링으로 영화 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'google', unseen_list, top_n=10)

# 평점 데이타를 DataFrame으로 생성.
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['google_pred_score'])
recomm_foods
```

Title	google_pred_score
고릴라브루잉	3.369816
장수삼	3.252676
오후의홍차	3.168791
해운대 달인막창	3.155588
칸다소바	3.145131
목구멍	3.139114
나가하마만게츠	3.129934
톤쇼우	3.122001
문토스트	3.121736
삼진어묵	3.121091

모 델

CF 추천 알고리즘 - MF

2.2 kakao가 이용하지 않은 가게 예측 평점

```
# 사용자가 이용하지 않는 가게명 추출
unseen_list = get_unseen_foods(ratings_matrix, 'kakao')

# 아이템 기반의 인접 이웃 협업 필터링으로 영화 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'kakao', unseen_list, top_n=10)

# 평점 데이타를 DataFrame으로 생성.
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['kakao_pred_score'])
recomm_foods
```

Title	kakao_pred_score
포르타나	3.637620
디젤앤카멜리아스	3.620617
다이닝룸	3.611133
풍원장미역국정찬	3.574533
전포명가떡집 (휴업중)	3.548800
원향재	3.514992
봉샌드	3.514505
풍원장 꼬막정식	3.493012
사보이	3.476737
해운대31cm해물칼국수	3.469151

모 델

CF 추천 알고리즘 - MF

2.3 naver가 이용하지 않은 가게 예측 평점

```
# 사용자가 이용하지 않는 가게명 추출
unseen_list = get_unseen_foods(ratings_matrix, 'naver')

# 아이템 기반의 인접 이웃 협업 필터링으로 영화 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'naver', unseen_list, top_n=10)

# 평점 데이타를 DataFrame으로 생성.
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['naver_pred_score'])
recomm_foods
```

Title	naver_pred_score
프랭클린커피로스터스	4.538275
목구멍	4.537936
겐짱카레	4.471980
마가만두	4.435869
블랙업커피	4.406302
할매국밥	4.390464
짱떡볶이	4.379833
모모스커피	4.370962
보성녹차	4.367928
삼진어묵	4.348149

모 델

CF 추천 알고리즘 - MF

2.4 tripadvisor가 이용하지 않은 가게 예측 평점

```
# 사용자가 이용하지 않는 가게명 추출
unseen_list = get_unseen_foods(ratings_matrix, 'tripadvisor')

# 아이템 기반의 인접 이웃 협업 필터링으로 영화 추천
recomm_foods = recomm_food_by_userid(ratings_pred_matrix, 'tripadvisor', unseen_list, top_n=10)

# 평점 데이타를 DataFrame으로 생성.
recomm_foods = pd.DataFrame(data=recomm_foods.values, index=recomm_foods.index, columns=['trip_pred_score'])
recomm_foods
```

Title	trip_pred_score
해목	4.336124
평산옥	4.242518
비비비당	4.234761
할매국밥	4.186285
치킨버거클럽	4.148581
나가하마만게츠	4.111232
선창횟집	4.105925
목구멍	4.078425
칸다소바	4.075209
동경밥상	4.072610