KFRs
(K-Food Recommendation Service)

외국인 관광객을 위한 딥러닝 기반 맞춤형 음식점 추천 서비스

[KFR]

김성우 김명준 서주원

목차

table of contents

1

프로젝트 개요

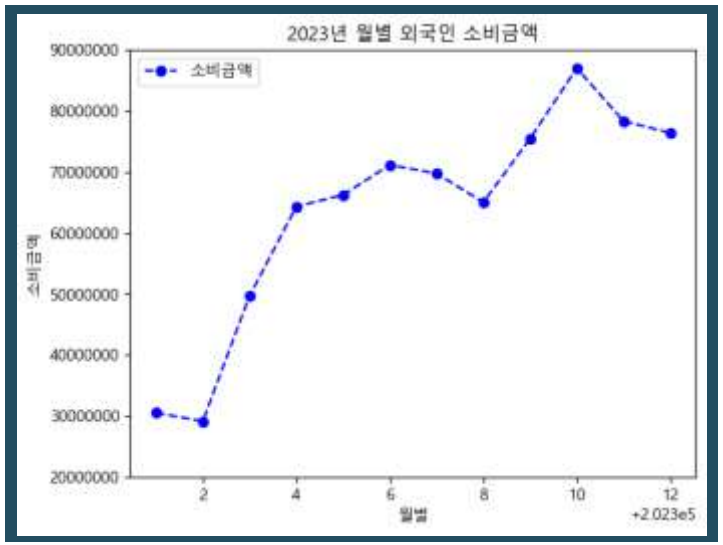
Part 1 **프로젝트 배경**

관광 현황 분석

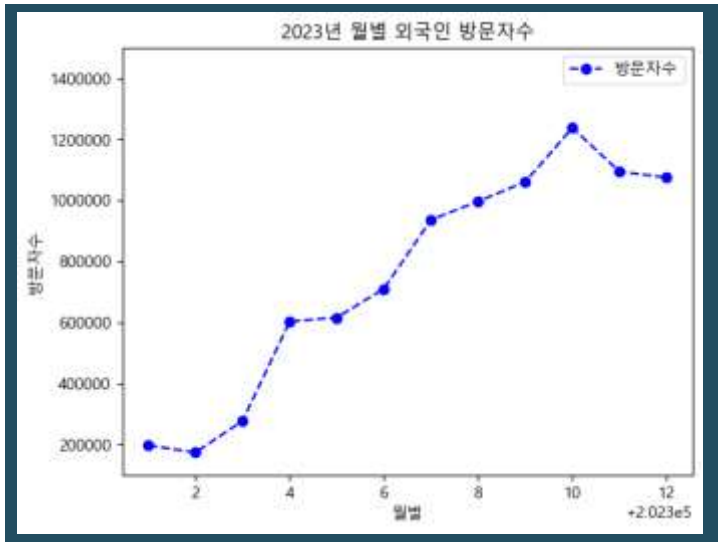
2023년 기준 외국인 방문자수 증가로 인한 외국인 수요 증대

- 한국관광데이터랩의 **'방한여행 행태 및 만족도 평가'**에 따르면 외국인 관광객의 재방문율은 최대 **77.7%**에 달함
- 또한 **'방한 목적'**을 **'관광'**을 목적으로 방한하는 외국인 관광객의 비율이 **70%** 이상으로 집계됨

외국인 소비금액
&
외국인 유입 증가



월별 외국인 소비 금액



월별 외국인 방문자수

관광 인프라 활성 및 외국인을 위한 다양한 인프라의 활성 필요

Part 1 **프로젝트 배경**

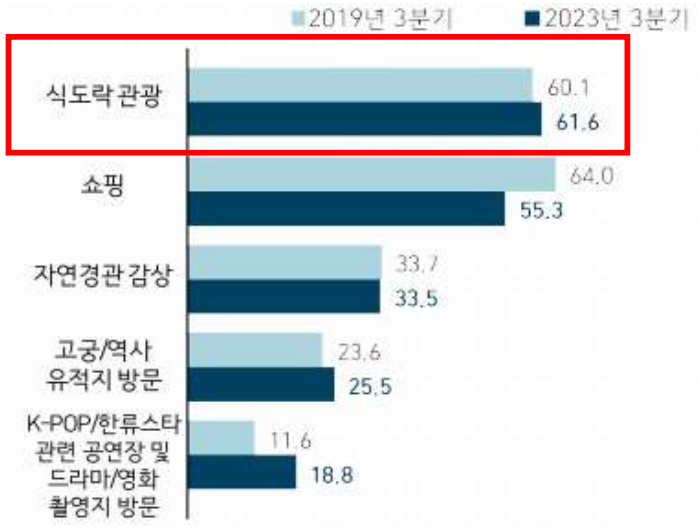
관광 현황 분석

외국인 관광객 주요 관광 활동으로 '식도락 관광'이 우세(2023년 3분기 기준)

- '2023년 외래관광객 조사 3분기' 보고서에 따르면 한국을 최종 관광목적지로 선택할때 고려한 주요 관광 활동으로 **식도락 관광**이 **61.6%**로 우세를 보임
- 그에 비해, 여행 준비 시 부족했던 한국 관련 정보는 '교통정보' 다음으로 **'음식 및 맛집 정보'**가 **15.8%**로 **2위**로 집계됨

[교통정보 - 21.7% / 음식 및 맛집 정보 - 15.8% / 방문지 정보 - 14.8% / 금융 정보 - 13.6%]

*출처 : 한국관광 데이터랩



방한 고려 관광 활동

부족했던 한국 관련 정보

분기별 방한 외래관광객 조사

2023년 분기별 방한 고려 관광 활동 TOP3

구분	1분기	2분기	3분기
식도락 관광	56.7%	60.2%	61.6%
쇼핑	61.7%	56.5%	55.3%
자연경관 감상	33.5%	37.6%	33.5%

2023년 분기별 부족했던 한국 관련 정보 TOP3

구분	1분기	2분기	3분기
교통정보	20.7%	22.1%	21.7%
음식 및 맛집 정보	16.6%	15.0%	15.8%
방문지 정보	14.1%	14.7%	14.8%

Part 1 **프로젝트 목적**

<2023년 외래관광객 조사 3분기> 자료에 따르면 **'식도락 관광'**을 목적으로 방한하는
외국인 관광객들은 꾸준히 증가하는 추세를 보임
반면에, '**음식 및 맛집 정보**'에 대한 **정보 부족 현상**도 높은 비율을 보이며
이를 해결하기 위해 외국인 관광객들을 대상으로 **음식 정보에 대한 인프라 개선**이 필요해 보임

따라서, 본 프로젝트는 딥러닝 기반의 **이미지 분류 모델**을 활용해
사용자가 **음식 이미지만으로도** 해당 음식과 관련된 **맛집** 및 **음식관련 정보**를 제공해주는
서비스를 개발해 외국인 관광객들로 하여금 **편의성**을 제공해주고자 함

이미지 분류 모델을 활용한

서울시 인기 맛집 추천 서비스 구현

한식 이미지 데이터 수집 → 네이버 리뷰 크롤링 및 적재 → 이미지 분류 모델 개발 → 음식 분류 서비스 제공 및 음식점 추천 → 외국인 관광객 한국 만족도 증가

2

프로젝트 프로세스

Part 2 **프로젝트 프로세스**

데이터 수집 : Ai-Hub 한식 40종 이미지 사진, 네이버 리뷰 크롤링 등

↓

모델링 : Inception V3 딥러닝 모델 학습

↓

데이터 적재 : DB에 데이터 저장

↓

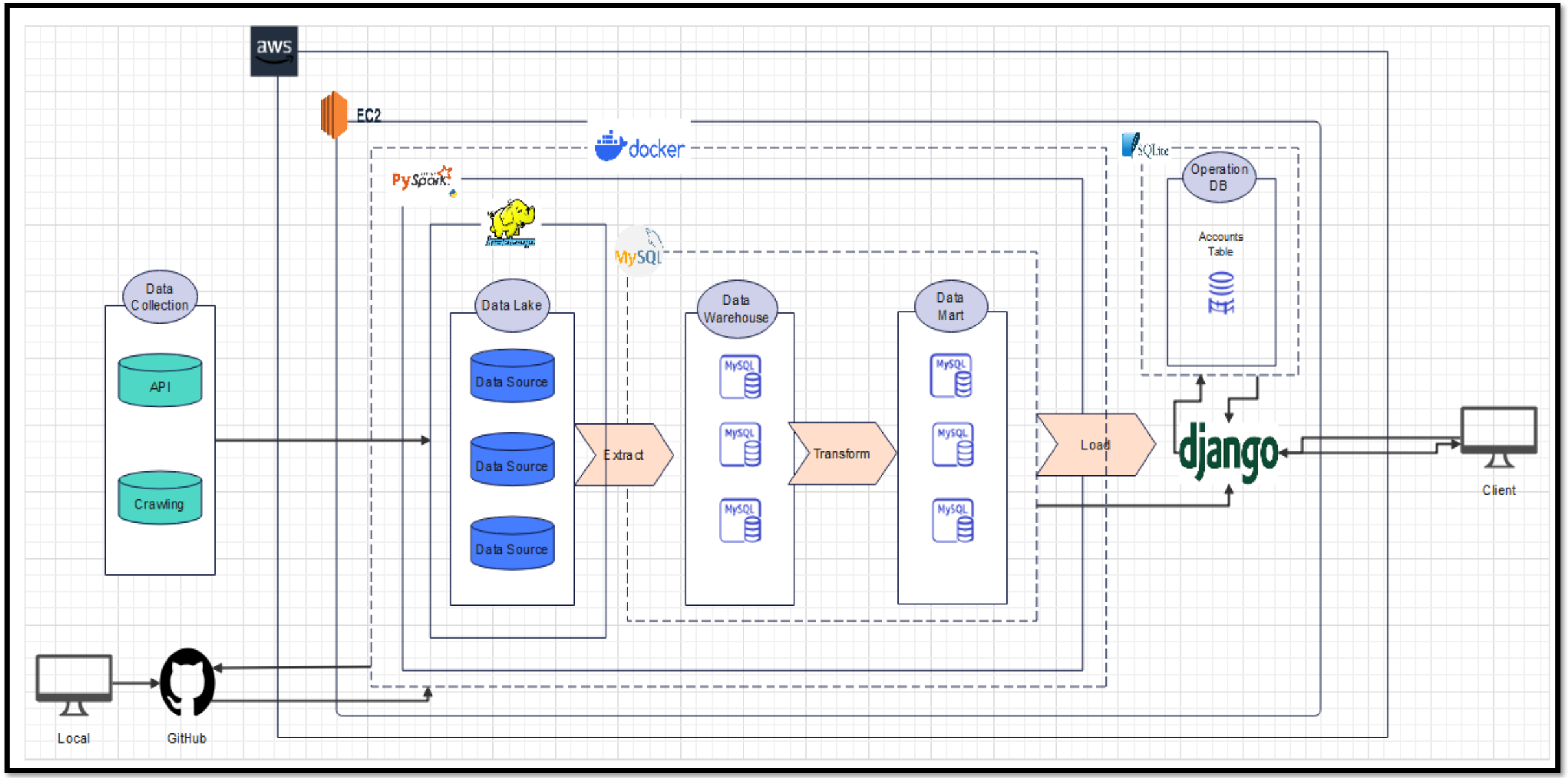
서비스 구현 : Django를 활용하여 웹서비스 구축

Part 2 **프로젝트 프로세스**

현황 분석 → 데이터 수집 → 데이터 모델링 → 데이터 적재

ETL Process → DB 모델링 → 웹 구현 → 서비스 배포

Part 2 **시스템 아키텍처 및 데이터 파이프라인**



프로젝트 Work-Flow

구분	기간	활동	활용 도구
사전 계획	23.12.26 ~ 24.01.05	주제선정 및 수행계획 수립	**Google Drive**
전처리 및 모델링	24.01.06 ~ 24.01.20	데이터 수집 및 전처리, 데이터 모델링	**Selenium, Pandas, Bs4**
데이터 파이프라인 구축	24.01.21 ~ 24.01.28	ETL, 데이터베이스 설계 및 테이블 간 관계 설정	**Hadoop, Pyspark, Mysql, Sqlite3**
웹 서비스 구현	24.01.29 ~ 24.02.05	DB 연동 및 웹 디자인, 웹 기능 개발	**Django**
포트폴리오 작성	24.02.05 ~ 24.02.07	PPT 작성	**PowerPoint, vllo(영상편 집 툴)**
프로젝트 발표	2024.02.08	파이널 프로젝트 발표	**PowerPoint**

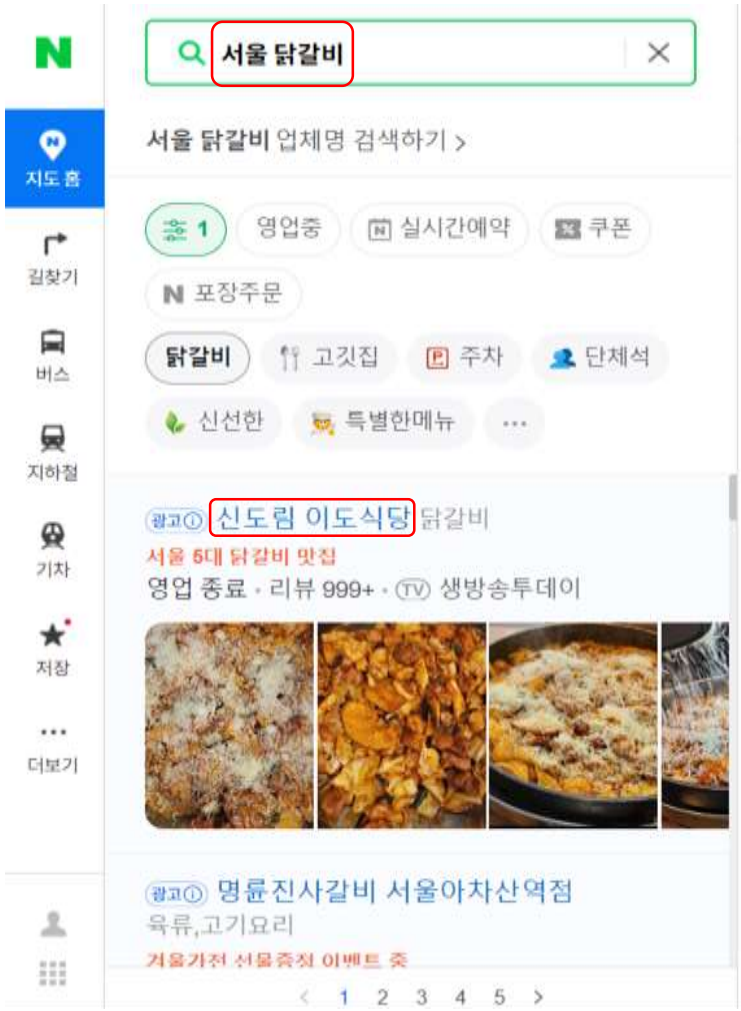
3

수행절차 및 방법

Part 3-1

리뷰 데이터 수집

수행절차 및 방법



검색창에 음식 입력 후,
주소, 전화번호
방문자 리뷰 상위 5개
가게 리뷰 1개 추출

Part 3-1 리뷰 데이터 수집

수행절차 및 방법

```
def create_driver():
    service = Service()
    options = webdriver.ChromeOptions()
    driver = webdriver.Chrome(service=service, options=options)
    return driver
```

- 웹 페이지를 크롬으로 설정

```
def open_browser(url):
    driver = create_driver()
    driver.get(url)
    driver.implicitly_wait(2)
    driver.maximize_window()
    return driver
```

- 2초안에 웹페이지를 load 하면 바로 넘어가거나, 2초를 기다림
- 페이지 열고 화면 비율을 맥심으로

```
def perform_page_down(scroll_element, num_pages=50, delay=1):
    for _ in range(num_pages):
        scroll_element.send_keys(Keys.PAGE_DOWN)
        time.sleep(delay)
```

- 스크롤 50회 수행

```
LOC = '서울'
food_lst = ['간장게장', '갈비찜', '갈비탕', '감자전', '감자탕', '곱창구이', '김밥', '김치전',
'김치찌개', '닭갈비', '닭볶음탕', '도토리묵', '된장찌개', '떡볶이', '막국수', '물냉면', '물회',
'미역국', '배추김치', '불고기', '비빔냉면', '비빔밥', '삼겹살', '삼계탕', '설렁탕', '순대',
'순두부찌개', '양념게장', '양념치킨', '육회', '잡채', '제육볶음', '족발', '주꾸미볶음', '짜장면',
'칼국수', '파전', '해물찜', '황태구이', '후라이드치킨']
crawl_lst = [f'{LOC} {food}' for food in food_lst]
```

- 외국인이 좋아하거나 먹고 싶은 음식 40선 선정

Part 3-1 리뷰 데이터 수집

수행절차 및 방법

```
for crawl in crawl_list:
    search_food = crawl
    driver = open_browser('https://map.naver.com/')
    search_element = (By.CLASS_NAME, 'input_search')
    search_box = WebDriverWait(driver, 5).until(
        EC.presence_of_element_located(search_element))
```

- 네이버 지도 브라우저 열기

```
    # 키 입력 및 엔터
    search_box.click()
    time.sleep(1)
    search_box.clear()
    search_box.send_keys(search_food)
    search_box.send_keys(Keys.ENTER)
    time.sleep(2)

    # searchIframe 진입
    searchIF = driver.find_element(By.CSS_SELECTOR, '#searchIframe')
    driver.switch_to.frame(searchIF)
```

- 검색하고자 하는 음식 대입 후 ENTER
- searchIframe 진입

```
    # searchIframe 스크롤
    searchIF_scroll_locator = (By.CSS_SELECTOR, 'body')
    searchIF_scroll = WebDriverWait(driver, 5).until(
        EC.element_to_be_clickable(searchIF_scroll_locator))

    searchIF_scroll.click()
    time.sleep(1)

    perform_page_down(searchIF_scroll, 35)

    li_elements = driver.find_element(By.CSS_SELECTOR,   '#_pcmap_list_scroll_container').find
_elements(By.TAG_NAME, "li")
    time.sleep(1)
```

- searchIframe 내 스크롤 페이지 끝까지 내리기
- 동적으로 생성되는 모든 음식점 **li태그** 가져오기

Part 3-1 리뷰 데이터 수집

수행절차 및 방법

```
for i in range(len(li_elements)):
    if i == 0:
        element1 = li_elements[i+1].find_element(By.CLASS_NAME, 'TYaxT')
        driver.execute_script("arguments[0].click();", element1)
        time.sleep(1)
```

- 첫번째 음식점 클릭 시, 원치 않는 부분 클릭되는 경우가 발생해 첫번째 li태그일 경우 두번째 음식점 태그 먼저 클릭하기

```
    element0 = li_elements[i].find_element(By.CLASS_NAME, 'TYaxT')
    driver.execute_script("arguments[0].click();", element0)
    time.sleep(1)

    driver.switch_to.default_content()
    time.sleep(2)
```

- 두번째 음식점 클릭 후 다시 첫번째 음식점 클릭
- **driver_switch_to default_content()**
  : 음식점 세부정보가 담긴 페이지의 iframe으로 이동하기 위해 driver 위치 초기화

```
    ## entryIframe
    WebDriverWait(driver, 10).until(
        EC.frame_to_be_available_and_switch_to_it((By.ID, 'entryIframe')))
    # driver.switch_to.frame(entryIF_element)

    # entryIframe 클릭
    entry_click_element = (By.CSS_SELECTOR, '#app-root > div > div > div > div.place_fixed_maintab > div > div > div > div > a:nth-child(1) > span')
    entry_click = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable(entry_click_element))
     entry_click.click()
     time.sleep(1)
```

- 음식점 세부정보가 담긴 **'entryIframe'**으로 진입
- 진입 후 페이지 스크롤 위해 요소 클릭

Part 3-1 리뷰 데이터 수집

수행절차 및 방법

```
entryIF_scroll_locator = (By.CSS_SELECTOR, 'body')
entryIF_scroll = WebDriverWait(driver, 5).until(
    EC.presence_of_element_located(entryIF_scroll_locator))

 perform_page_down(entryIF_scroll, 6)
```

- 'entryIframe' 내 스크롤 끝까지 내리기

```
# data crawling
food.append(search_food.split(' ')[1])

name_element = (By.CSS_SELECTOR, '#_title > div > span.Fc1rA')
name = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located(name_element)).text
store.append(name)
```

- 검색하는 음식 이름과 음식점 이름이 담긴 element 찾아 리스트에 추가

```
reviews = []  # reviews 변수 초기화
try:
    reviews_element = (By.CLASS_NAME, 'nbD78')
    reviews = WebDriverWait(driver, 5).until(
        EC.presence_of_all_elements_located(reviews_element)
    )
except TimeoutException:
    for i in range(1, 6):
        # 기존에 생성되어 있는 변수에 append
        globals()[f'rank{i}'].append('미등록')

for index, review in enumerate(reviews):
    # 리뷰의 텍스트 가져오기
    review_text = review.text.replace('₩n', '').replace('이 키워드를 선택한 인원', '').replace('"',
'')
    modified_string = re.sub(r'(₩D)(₩d+)(₩D|$)', r'₩1 ₩2₩3', review_text)
```

- 방문자 리뷰 키워드에 대한 element를 활용해 방문자 리뷰 키워드 추출
- 키워드 미등록시 ' 미등록'으로 저장
- 정규식을 활용해 불필요한 텍스트 삭제 후 상위 5개의 방문자 리뷰 키워드 추출

Part 3-1 리뷰 데이터 수집

수행절차 및 방법

```
        if index == 0:
            rank1.append(modified_string)
        elif index == 1:
            rank2.append(modified_string)
        elif index == 2:
            rank3.append(modified_string)
        elif index == 3:
            rank4.append(modified_string)
        elif index == 4:
            rank5.append(modified_string)
```

- 상위 5개의 방문자 리뷰를 rank1~rank5의 리스트로 나눠서 저장하기 위해 index별로 따로 append 하기

```
        # 방문자 리뷰 더보기 클릭
        try:
            more_text_element = (By.CSS_SELECTOR, 'div.RGkHL > a > span.rvCSr > span')
            more_text = WebDriverWait(driver, 5).until(
                EC.presence_of_element_located(more_text_element))
            driver.execute_script("arguments[0].click();", more_text)
        except TimeoutException:
            pass
```

- 등록된 방문자 리뷰 중 상위 1개의 방문자 리뷰만 추출
- 리뷰 전체 내용을 추출하기 위해 해당 리뷰에 대한 더보기 버튼 click, 없으면 pass하기

```
        # 방문자 리뷰 저장
        try:
            vreview_element = (By.CSS_SELECTOR, 'div.TraH1 > ul > li:nth-child(1) > div.IEbo1 > div.
GP2eR > div.RGkHL > a > span')
            vreview = WebDriverWait(driver, 5).until(
                EC.presence_of_element_located(vreview_element)).text.replace('₩n', '')
        except TimeoutException:
            vreview = '미등록'

        visitor_review.append(vreview)
```

- 리뷰 내용 저장
- 내용이 없을 시 '미등록 ' 으로 저장하는 예외처리 추가

Part 3-1 리뷰 데이터 수집

수행절차 및 방법

```
        # 주소 저장
        try:
            addr_element = (By.CLASS_NAME, 'LDgIH')
            addr_lo = WebDriverWait(driver, 5).until(
                EC.presence_of_element_located(addr_element)).text
        except TimeoutException:
            addr_lo = '미등록'

        addr.append(addr_lo)
```

- 주소에 대한 element 찾은 후 주소 데이터 저장
- 주소 부재 시 '미등록' 저장

```
        # 전화번호 저장
        try:
            tel_element = (By.CLASS_NAME, 'xlx7Q')
            tel_lo = WebDriverWait(driver, 5).until(
                EC.presence_of_element_located(tel_element)).text

        except TimeoutException:
            tel_lo = '미등록'
        tel.append(tel_lo)

        driver.switch_to.default_content()
        time.sleep(1)
        driver.switch_to.frame(searchIF)

driver.close()
```

- 전화번호에 대한 element 찾은 후 전화번호 데이터 저장
- 전화번호 부재 시 '미등록' 저장

Part 3-1

리뷰 데이터 수집

수행절차 및 방법

40종의 음식 리뷰를 크롤링하여 2081개의 데이터 수집

	food	store_name	addr	tel	re_rank1	re_rank2	re_rank3	re_rank4	re_rank5	re_visitor
0	간장게장	춘남식당	서울 마포구 성미산로 97 춘남식당	0507-1384-0693	음식이 맛있어요 35	재료가 신선해요 28	양이 많아요 24	친절해요 23	특별한 메뉴가 있어요 12	꽃게탕 소자에 칼국수사리, 볶음밥을 추가해서 먹었어요! 연평도산 꽃게라 그런지 야들...
1	간장게장	장안동 먹깨비 간장게장 해물아구찜	서울 동대문구 장한로 92 1층	0507-1393-2158	음식이 맛있어요 456	친절해요 225	재료가 신선해요 202	양이 많아요 164	매장이 넓어요 115	남친 부모님이 매운걸 못드셔서 안맵게로 부탁존나 맵게해야 더 맛있을듯 이모님 사장...
2	간장게장	예담밥상	서울 광진구 동일로22길 13	0507-1443-0073	음식이 맛있어요 96	재료가 신선해요 52	친절해요 42	비싼 만큼 가치 있어요 33	차분한 분위기에요 30	간장게장과 갈비가 같이 나오는 특선을 추천해주셔서 먹었습니다~ ^^ 확실히 찬들...
3	간장게장	명현만간장게장무한리필	서울 마포구 양화로 45 메세나폴리스 123호~129호	0507-1302-3837	음식이 맛있어요 677	재료가 신선해요 385	친절해요 370	양이 많아요 328	가성비가 좋아요 315	명현만의 간장게장 합정점! 평소 명현만 선수의 팬이었다면 더욱 좋을 것 같아요! 가...
4	간장게장	진미식당	서울 마포구 마포대로 186-6	02-3211-4468	음식이 맛있어요 364	재료가 신선해요 180	특별한 메뉴가 있어요 105	친절해요 73	특별한 날 가기 좋아요 41	맛있어요~예약이 힘들었는데 다행히 성공했어요ㅋ 들어가서 앉고 거의바로 음식이나왔어요...
...	...	...	...	...	...	...	...	...	...	...
2076	후라이드치킨	뮌헨호프	서울 중구 을지로14길 30 1층	02-2273-2288	음식이 맛있어요 264	가성비가 좋아요 231	기본 안주가 좋아요 174	단체모임 하기 좋아요 151	친절해요 118	팝콘맛집.오후에 팝콘을 미리 만드는데, 고소하니 팝콘이 젤 따끈하니 맛있음.국물떡볶...
2077	후라이드치킨	달가득치킨	서울 은평구 연서로28길 10	0507-1337-9390	음식이 맛있어요 347	친절해요 132	양이 많아요 108	재료가 신선해요 91	가성비가 좋아요 84	역시 맥주는 치킨이답이죠!! 시원한 생맥에 바삭한 치킨 매콤달콤한 떡볶이까지 환상의...
2078	후라이드치킨	어거스트치킨 양재역점	서울 서초구 강남대로 224 한신휴플러스 101호	0507-1477-8294	음식이 맛있어요 196	친절해요 73	매장이 넓어요 64	단체모임 하기 좋아요 49	양이 많아요 42	저녁에는 기다려야 먹을수 있는 통닭맛집으로 알고있는데 이른시간에 가니 여유롭게 먹어...

Part 3-1 리뷰 데이터 수집

수행절차 및 방법

```
# json에 포함시킬 meta data 구성
res = {
        'meta':{
            'desc':'음식점 리뷰 크롤링 데이터',
            'cols':{
                'food':'food'
                ,'store_name':'store_name'
                ,'addr':'addr'
                ,'tel' : 'tel'
                ,'re_rank1' : 're_rank1'
                ,'re_rank2' : 're_rank2'
                ,'re_rank3' : 're_rank3'
                ,'re_rank4' : 're_rank4'
                ,'re_rank5' : 're_rank5'
                ,'re_visitor' : 're_visitor'
                  },
        },
        'data':data
}
```

JSON 형식으로 변환


```
{'meta': {'desc': '음식점 리뷰 크롤링 데이터',
  'cols': {'food': 'food',
   'store_name': 'store_name',
   'addr': 'addr',
   'tel': 'tel',
   're_rank1': 're_rank1',
   're_rank2': 're_rank2',
   're_rank3': 're_rank3',
   're_rank4': 're_rank4',
   're_rank5': 're_rank5',
   're_visitor': 're_visitor'}},
 'data': [{'food': '간장게장',
   'store_name': '춘남식당',
   'addr': '서울 마포구 성미산로 97 춘남식당',
   'tel': '0507-1384-0693',
   're_rank1': '음식이 맛있어요 35',
   're_rank2': '재료가 신선해요 28',
   're_rank3': '양이 많아요 24',
   're_rank4': '친절해요 23',
   're_rank5': '특별한 메뉴가 있어요 12',
   're_visitor': '꽃게탕 소자에 칼국수사리, 볶음밥을 추가해서 먹었어요! 연평도산 꽃게라 그런지 야들야들하고 부드
  {'food': '간장게장',
   'store_name': '장안동 먹깨비 간장게장 해물아구찜',
   'addr': '서울 동대문구 장한로 92 1층',
   'tel': '0507-1393-2158',
```

Part 3-2 DB에 저장/전달 하는 함수

수행절차 및 방법

```
conf_dw = {
      'url':'jdbc:mysql://localhost:3306/finalPrj_DW?characterEncoding=utf8&serverTimezone=Asia/Seoul'
      ,'props':{
      'user':'bigMysql',
      'password':'bigMysql1234@'
      }
}

conf_dm = {
      'url':'jdbc:mysql://localhost:3306/finalPrj_DM?characterEncoding=utf8&serverTimezone=Asia/Seoul'
      ,'props':{
      'user':'bigDM',
      'password':'bigDM1234@'
      }
}

def find_data(config, table_name) :
      return spark.read.jdbc(url= config['url'], table=table_name, properties=config['props'])

def save_data(config, df, table_name) :
      return df.write.jdbc(url= config['url'], table=table_name, mode='append' , properties=config['props'])
```

Part 3-1 리뷰 데이터 DW에 적재

수행절차 및 방법

```
read_restaurant_info = read_restaurant_info.filter(col("addr").startswith("서울"))
```

- 크롤링한 데이터에서 서울이 아닌 다른지역 음식점 제거

```
restaurant_drop_duplicate = restaurant_info .dropDuplicates(["food", "store_name", 'addr'])
restaurant_drop_duplicate.show(truncate=False)
```

- 중복 제거
- 총 2054개의 데이터

```
+-----------------------------------------------------------------+-------+-----------------+-----------------+-----------------+-----
|addr                                                             |food   |re_rank1         |re_rank2         |re_rank3         |re_ran
+-----------------------------------------------------------------+-------+-----------------+-----------------+-----------------+-----
|서울 강동구 상일로6길 39 지하1층                                |간장게장|음식이 맛있어요 404|매장이 넓어요 170|양이 많아요 141  |단체모임 하
|서울 강남구 선릉로131길 17 1층                                  |간장게장|음식이 맛있어요 230|재료가 신선해요 164|매장이 청결해요 159|친절해요 14
|서울 강남구 강남대로84길 33 대우디오빌플러스 지하1층 107호     |간장게장|음식이 맛있어요 117|친절해요 80      |재료가 신선해요 67 |특별한 메뉴
|서울 강서구 공항대로 269-15 힐스테이트에코 마곡 2층 220-1호 (A동 방향)|간장게장|음식이 맛있어요 1182|친절해요 697   |재료가 신선해요 672|매장이 청결하
|서울 서대문구 연희로26길 28                                     |간장게장|음식이 맛있어요 413|친절해요 157     |재료가 신선해요 155|가성비가 좋
|서울 동대문구 장한로17길 13-10 1층 101호                        |간장게장|음식이 맛있어요 238|가성비가 좋아요 150|친절해요 121   |재료가 신선
|서울 마포구 어울마당로 136-3 2층 꽃게나라                       |간장게장|음식이 맛있어요 136|재료가 신선해요 94|친절해요 88      |양이 많아요
|서울 종로구 율곡로1길 31 지하1층                                |간장게장|음식이 맛있어요 452|재료가 신선해요 250|양이 많아요 195|가성비가 좋
|서울 강남구 봉은사로 610                                        |간장게장|음식이 맛있어요 130|재료가 신선해요 50|친절해요 45      |특별한 메뉴
|서울 도봉구 시루봉로 139-6                                      |간장게장|음식이 맛있어요 586|친절해요 184     |단체모임 하기 좋아요 181|재료가 신선
|서울 영등포구 국회대로 800 여의도파라곤 지하1층                 |간장게장|음식이 맛있어요 53|친절해요 25       |매장이 넓어요 19  |차분한 분위
|서울 영등포구 선유로17길 24 신일아르디세 1층                    |간장게장|음식이 맛있어요 315|재료가 신선해요 129|친절해요 99    |매장이 넓어
|서울 강북구 덕릉로41길 42 1층 레알 간장게장 무한리필 본점      |간장게장|음식이 맛있어요 313|가성비가 좋아요 194|재료가 신선해요 191|양이 많아요
|서울 강서구 공항대로 261 발산파크프라자 2층 208호               |간장게장|음식이 맛있어요 201|재료가 신선해요 116|친절해요 113   |매장이 청결하
|서울 마포구 양화로 45 메세나폴리스 123호~129호                  |간장게장|음식이 맛있어요 677|재료가 신선해요 385|친절해요 370   |양이 많아요
|서울 강서구 마곡동로 161 서울식물원 4층                         |간장게장|음식이 맛있어요 97|매장이 넓어요 53|재료가 신선해요 51|뷰가 좋아요
|서울 송파구 백제고분로 420 1층                                  |간장게장|음식이 맛있어요 754|재료가 신선해요 431|친절해요 347   |매장이 넓어
|서울 송파구 위례성대로 6 현대토픽스 2층                         |간장게장|음식이 맛있어요 1029|매장이 넓어요 365|단체모임 하기 좋아요 352|재료가 신선하
|서울 서초구 반포대로 287 고속터미널역 G1 출구                   |간장게장|음식이 맛있어요 840|매장이 넓어요 286|단체모임 하기 좋아요 280|가성비가 좋
|서울 동대문구 장한로24길 7 1층                                  |간장게장|음식이 맛있어요 180|재료가 신선해요 79|매장이 넓어요 53  |친절해요 46
+-----------------------------------------------------------------+-------+-----------------+-----------------+-----------------+-----
only showing top 20 rows
```

Part 3-2 음식 정보 데이터 수집

수행절차 및 방법

```
def food_info(name):
    url = f"https://www.10000recipe.com/recipe/list.html?q={name}"
    response = requests.get(url)
    if response.status_code == 200:
        html = response.text
        soup = BeautifulSoup(html, 'html.parser')
    else :
        print("HTTP response error :", response.status_code)
        return

    food_list = soup.find_all(attrs={'class':'common_sp_link'})
    food_id = food_list[0]['href'].split('/')[-1]
    new_url = f'https://www.10000recipe.com/recipe/{food_id}'
    new_response = requests.get(new_url)
    if new_response.status_code == 200:
        html = new_response.text
        soup = BeautifulSoup(html, 'html.parser')
    else :
        print("HTTP response error :", response.status_code)
        return
```

- 만개의 레시피에서 음식의 재료 크롤링


```
    food_info = soup.find(attrs={'type':'application/ld+json'})
    result = json.loads(food_info.text)
    ingredient = ','.join(result['recipeIngredient'])
    wiki_ko = wikipediaapi.Wikipedia('KFR ()', 'ko')
    page_py_ko = wiki_ko.page(name)

    res = {
        '음식': name,
        '정보': page_py_ko.summary,
        '재료': ingredient
    }

    return res
```

- 위키디피아 API를 활용하여 음식 정보 가져오기

Part 3-2 음식 정보 데이터 수집

수행절차 및 방법

```
rows = []
for name in food_names :
    a = food_info(name)['음식']
    b = food_info(name)['정보']
    c = food_info(name)['재료']
    row = [a, b, c]
    rows.append(row)
food_info = pd.DataFrame(rows, columns
'ingredient'])
```

```
cols = ['food', 'info', 'ingredient']
data = []
for idx in food_info.index :
    tuple_t = []
    tmp = food_info.loc[idx]
    tuple_t.append(str(tmp.food))
    tuple_t.append(str(tmp.info))
    tuple_t.append(str(tmp.ingredient))
    data.append(dict(zip(cols,tuple_t)))
```

```
# json에 포함시킬 meta data 구성
res = {
        'meta':{
            'desc':'음식 재료, 정보',
            'cols':{
                '음식':'food'
                ,'정보':'info'
                ,'재료':'ingredient'
                  },
        },
        'data':data
}
```

food	info	ingredient
0 간장게장	게장(-醬)은 한국의 게 요리이다. 게를 간장물에 담가 숙성해 만든 젓갈의 하나로,..	꽃게 5마리,간장 3머그컵,물 3.5머그컵,청양고추 4개,홍고추 2개,사과 1개,생..
1 갈비찜	갈비찜은 갈비에 양념과 간을 하여 국물을 붓고 쪄서 만든 찜류의 음식이다. 과거엔 ..	돼지갈비 2kg,감자,양파,대파,당근,버섯,밤,청양고추,진간장 2컵,설탕 1컵,물 ..
		갈비탕용 소고기 1kg,무 1토막,대파 흰부분 5대,양파 1/2개,당면 1줌,물 1..
		개,물 감자가잠길만큼,소금 1꼬집,식용유 넉넉히,간장 3큰술,식초 1/2큰술..
		감자탕용 돼지등뼈 1.5kg,감자 작은거 6개,대친 열갈이 배추 200g,깻잎 10..

```
+--------------------+
| Tables_in_etlmysql |
+--------------------+
| food_info          |
| restaurant_list    |
+--------------------+
2 rows in set (0.00 sec)
```

ON 형식으로 변환 후, DW에 저장

Part 3-2 데이터 마트 적재 (restaurant_info)

수행절차 및 방법

```
def json_request(url):
    headers = {'Authorization': 'KakaoAK {}'.format(KAKAO_API_KEY)}
    res = requests.get(url, headers=headers)
    return res.text
```

- 카카오서버에 요청 후 응답받은 데이터를 반환하는 함수

```
def addr_lat_lon(addr):
    url =
'https://dapi.kakao.com/v2/local/search/address.json?query={address}'.format(address=add
r)
    try :
        res_json = json_request(url)
        res = json.loads(res_json)
        match_adr = res['documents'][0]['address']
    except :
        return 'NaN','NaN'

    return float(match_adr['x']), float(match_adr['y'])
```

- 주소를 통하여 좌표값 추출
- 찾지 못할 경우 NAN, NAN로 반환

```
def cg_addr(df, store_name, new_address):
    df.loc[df['store_name'] == store_name, 'addr'] = new_address
    return df
```

```
udf_addr_lat_lon = udf(addr_lat_lon, StructType([
    StructField("x", FloatType(), True),
    StructField("y", FloatType(), True)
]))
```

- 좌표 값을 찾지 못한 주소를 확인하여 바꿈
- 'addr_lat_lon' 함수를 PySpark UDF로 등록 좌표를 구하는 함수를 각 row마다 실행시키기 위해

Part 3-2 데이터 마트 적재 (restaurant_info)

수행절차 및 방법

```
res = df_spark
res_point = res.withColumn("좌표", udf_addr_lat_lon("addr"))₩
              .persist()
```

- 좌표컬럼을 추가, 튜플로 좌표가 들어감
- '좌표' 컬럼을 가지고 있는 df = res_point

```
def extract_coordinates(coord):
    if coord:
        return coord.x, coord.y
    else:
        return None, None

udf_extract_coordinates = udf(extract_coordinates, StructType([
    StructField("x", FloatType(), True),
    StructField("y", FloatType(), True)
]))
```

- x, y좌표를 나누는 함수 작성
- 'extract_coordinates' 함수를 PySpark UDF로 등록

```
restaurant_df = res_point.withColumn("x좌표",
udf_extract_coordinates("좌표").getField("x"))₩
                  .withColumn("y좌표", udf_extract_coordinates("좌표").getField("y")).persist()
```

- restaurant_df에 최종 x,y좌표를 나눠서 입력

```
columns_to_drop = ['re_rank1', 're_rank2', 're_rank3', 're_rank4', 're_rank5', 're_visitor', '좌표']
restaurant_df = restaurant_df.drop(*columns_to_drop)
```

- 불필요한 컬럼 제거

Part 3-2 # 데이터 마트 적재 (restaurant_info)

카카오 api를 활용해 x,y 좌표를 추가한 데이터 적재

food	store_name	addr	tel	x좌표	y좌표	id
감자전	04판	서울 종로구 대명1길 21	0507-1318-1711	127.00089	37.582714	0
김치전	04판	서울 종로구 대명1길 21	0507-1318-1711	127.00089	37.582714	1
갈비찜	24시 산더미불고기	서울 마포구 어울마당로 47	0507-1305-7626	126.9206	37.549465	2
감자탕	24시 용산원조감자탕	서울 용산구 청파로 393	02-797-1900	126.968796	37.55474	3
김치찌개	24시서울밥집	서울 관악구 관천로 36-1	미등록	126.927574	37.483192	4
순두부찌개	24시서울밥집	서울 관악구 관천로 36-1	미등록	126.927574	37.483192	5
제육볶음	24시서울밥집	서울 관악구 관천로 36-1	미등록	126.927574	37.483192	6
삼계탕	3대삼계장인	서울 서초구 반포대로28길 56...	02-522-2270	127.01159	37.491302	7
김밥	5412	서울 용산구 이태원로54가길 1...	02-790-7990	127.00167	37.53726	8
감자전	7.8 을지로	서울 중구 창경궁로8길 22-7	02-7182-2060	126.99943	37.56847	9
닭볶음탕	77년생 곱도리 전골식당	서울 광진구 동일로24길 70 ...	0507-1360-6998	127.06879	37.54283	10
파전	79번지국수집	서울 동대문구 회기로13길 25 1층	0507-1344-9494	127.05097	37.592678	11
순두부찌개	853	서울 종로구 인사동12길 16 ...	02-733-0853	126.98511	37.574734	12
물회	Gongi	서울 용산구 이태원로45길 4	0507-1404-7753	126.999344	37.536198	13
후라이드치킨	JH텍사스바	서울 중구 무교로 32 효령빌딩	02-774-0804	126.97956	37.568768	14
물회	WOW.물회와황태탕	서울 송파구 석촌호수로12길 3...	0507-1434-4889	127.08295	37.508537	15
주꾸미볶음	가가솥밥 강남점	서울 강남구 역삼로5길 19 1층	0507-1480-7903	127.03186	37.495026	16
감자탕	가나안뼈해장탕	서울 중구 다산로 131 약수장여관	02-2234-5200	127.010826	37.555702	17
잡채	가담	서울 강남구 언주로167길 35	02-545-5163	127.03026	37.526367	18
해물찜	가락골마산아구찜	서울 송파구 송이로19길 3 1층	0507-1370-6666	127.12172	37.49746	19

Part 3-2 # 데이터 마트 적재 (food_info_with_taboo)

수행절차 및 방법

Egg, Pork, Beef, Offal이 들어간 음식들을
새로운 컬럼을 추가하여 표기

```
pork = ['돼지', '족발', '햄']
beef = ['소고기','소불고기']
egg = ['계란','달걀']
offal = ['곱창','순대']
def categorize_ingredient(ingredient):
    for e in pork:
        if e in ingredient:
            return 'Pork'
    for e in beef:
        if e in ingredient:
            return 'Beef'
    for e in egg:
        if e in ingredient:
            return 'Egg'
    for e in offal:
        if e in ingredient:
            return 'Offal'
    else:
        return 'None'
```

food	info	ingredient
비빔냉면	냉면(冷麵, 문화어: 랭면)은 ...	냉면사리 2인분,시판냉면육수 1...
비빔밥	비빔밥은 한국의 밥 요리이다. ...	돼지고기 100g,애호박 1/5...
삼겹살	삼겹살의 다른 뜻은 다음과 같다...	돼지고기 600g,깻잎 10장,...
삼계탕	삼계탕(蔘鷄湯, 문화어: 인삼닭...	닭 2마리,삼계탕용약재,찹쌀,마...
설렁탕	설렁탕(영어: Seolleong...	무 1개,쪽파 150g,설탕 4...
순대	순대는 한국의 음식으로, 돼지의...	순대500g,양배추3컵,깻잎 1...
순두부찌개	순두부찌개는 순두부를 주재료로 ...	순두부,대파 1/4개,양파 1/...
양념게장	양념게장(---醬)은 한국의 게...	냉동 꽃게 4마리,간장(소주컵)...
양념치킨	양념치킨은 한국식 닭튀김 요리로...	물엿 4큰술,다진 마늘 2큰술,...
육회	육회는 대한민국의 요리에서 회의...	소고기(앞다리/우둔/엉덩이) 1...

⇩

food	info	ingredient	taboo_ingredient
비빔냉면	냉면(冷麵, 문화어: 랭면)은 ...	냉면사리 2인분,시판냉면육수 1...	Egg
비빔밥	비빔밥은 한국의 밥 요리이다. ...	돼지고기 100g,애호박 1/5...	Pork
삼겹살	삼겹살의 다른 뜻은 다음과 같다...	돼지고기 600g,깻잎 10장,...	Pork
삼계탕	삼계탕(蔘鷄湯, 문화어: 인삼닭...	닭 2마리,삼계탕용약재,찹쌀,마...	None
설렁탕	설렁탕(영어: Seolleong...	무 1개,쪽파 150g,설탕 4...	None
순대	순대는 한국의 음식으로, 돼지의...	순대500g,양배추3컵,깻잎 1...	Offal
순두부찌개	순두부찌개는 순두부를 주재료로 ...	순두부,대파 1/4개,양파 1/...	Egg
양념게장	양념게장(---醬)은 한국의 게...	냉동 꽃게 4마리,간장(소주컵)...	None
양념치킨	양념치킨은 한국식 닭튀김 요리로...	물엿 4큰술,다진 마늘 2큰술,...	None
육회	육회는 대한민국의 요리에서 회의...	소고기(앞다리/우둔/엉덩이) 1...	Beef

Part 3-2 데이터 마트 적재(visitor_review)

수행절차 및 방법

```
selected_columns = ['id', 're_visitor']
visitor_review = restaurant.select(*selected_columns)

visitor_review = visitor_review.withColumnRenamed("id", "res_id")

visitor_review = visitor_review.withColumn('id', monotonically_increasing_id())
visitor_review.show()
```

방문자 리뷰에 id값 추가한 후, DM에 적재

res_id	re_visitor
1088	게장이 짜지도않고너무 맛있어요부...
1092	비리지 않고 정갈하게 차려진 게...
1105	간장게장을 좋아하는 우리 딸이 ...
1081	점심시간때 갈더니 가격도 할인되...
1078	고미정에 주말 점심시간에 방문해...
1098	정말 오랜만에 먹어본 게장이었어...
1110	홍대간장게장! 간장게장을 처음먹...
1107	작년 여름쯤이었던가.부모님과 경...
1101	삼성동에 유명한 간장게장 집입니...
1083	아빠 칠순이라 저녁식사 했는데 ...
1082	엄마랑 한정식 먹으러 방문했어요...
1077	영등포 최고의 게장집!양념게장 ...
1066	수유 간장게장 맛집이에요!~ 암...
1086	가까운 게장 집을 찾아서 맛있게...
1060	명현만의 간장게장 합정점! 평소...
1063	퇴근후 찾은 서울식물원 저녁메뉴...
1062	방이동 간장게장 최고 맛집이에요...
1071	보는 반찬 아닌 먹는 반찬으로 ...
1085	가성비는 좋아요직원 분은 바빠서...
1097	명불허전 밥도둑 간장게장.. 너...

res_id	re_visitor	id
1088	게장이 짜지도않고너무 맛있어요부...	0
1092	비리지 않고 정갈하게 차려진 게...	1
1105	간장게장을 좋아하는 우리 딸이 ...	2
1081	점심시간때 갈더니 가격도 할인되...	3
1078	고미정에 주말 점심시간에 방문해...	4
1098	정말 오랜만에 먹어본 게장이었어...	5
1110	홍대간장게장! 간장게장을 처음먹...	6
1107	작년 여름쯤이었던가.부모님과 경...	7
1101	삼성동에 유명한 간장게장 집입니...	8
1083	아빠 칠순이라 저녁식사 했는데 ...	9
1082	엄마랑 한정식 먹으러 방문했어요...	10
1077	영등포 최고의 게장집!양념게장 ...	11
1066	수유 간장게장 맛집이에요!~ 암...	12
1086	가까운 게장 집을 찾아서 맛있게...	13
1060	명현만의 간장게장 합정점! 평소...	14
1063	퇴근후 찾은 서울식물원 저녁메뉴...	15
1062	방이동 간장게장 최고 맛집이에요...	16
1071	보는 반찬 아닌 먹는 반찬으로 ...	17
1085	가성비는 좋아요직원 분은 바빠서...	18
1097	명불허전 밥도둑 간장게장.. 너...	19

데이터 마트 적재(rank_review)

필요한 컬럼 선택하여 새로운 데이터프레임 생성

```
selected_columns = ['id', 're_rank1', 're_rank2', 're_rank3', 're_rank4', 're_rank5']
rank_review = restaurant.select(*selected_columns)
```

id	re_rank1	re_rank2	re_rank3	re_rank4	re_rank5
1088	음식이 맛있어요 404	매장이 넓어요 170	양이 많아요 141	단체모임 하기 좋아요 131	가성비가 좋아요 125
1092	음식이 맛있어요 230	재료가 신선해요 164	매장이 청결해요 159	친절해요 147	특별한 메뉴가 있어요 80
1105	음식이 맛있어요 117	친절해요 80	재료가 신선해요 67	특별한 메뉴가 있어요 48	매장이 청결해요 38
1081	음식이 맛있어요 1182	친절해요 697	재료가 신선해요 672	매장이 청결해요 601	가성비가 좋아요 482
1078	음식이 맛있어요 413	친절해요 157	재료가 신선해요 155	가성비가 좋아요 118	단체모임 하기 좋아요 96
1098	음식이 맛있어요 238	가성비가 좋아요 150	친절해요 121	재료가 신선해요 116	양이 많아요 109
1110	음식이 맛있어요 136	재료가 신선해요 94	친절해요 88	양이 많아요 71	가성비가 좋아요 49
1107	음식이 맛있어요 452	재료가 신선해요 250	양이 많아요 195	가성비가 좋아요 151	매장이 넓어요 144
1101	음식이 맛있어요 130	재료가 신선해요 50	친절해요 45	특별한 메뉴가 있어요 32	매장이 넓어요 31
1083	음식이 맛있어요 586	친절해요 184	단체모임 하기 좋아요 181	재료가 신선해요 178	특별한 날 가기 좋아요 121
1082	음식이 맛있어요 53	친절해요 25	매장이 넓어요 19	차분한 분위기에요 16	재료가 신선해요 14
1077	음식이 맛있어요 315	재료가 신선해요 129	친절해요 99	매장이 넓어요 82	매장이 청결해요 74
1066	음식이 맛있어요 313	가성비가 좋아요 194	재료가 신선해요 191	양이 많아요 170	친절해요 162
1086	음식이 맛있어요 201	재료가 신선해요 116	친절해요 113	매장이 청결해요 75	양이 많아요 68
1060	음식이 맛있어요 677	재료가 신선해요 385	친절해요 370	양이 많아요 328	가성비가 좋아요 315
1063	음식이 맛있어요 97	매장이 넓어요 53	재료가 신선해요 51	뷰가 좋아요 36	친절해요 35
1062	음식이 맛있어요 754	재료가 신선해요 431	친절해요 347	매장이 넓어요 230	양이 많아요 206
1071	음식이 맛있어요 1029	매장이 넓어요 365	단체모임 하기 좋아요 352	재료가 신선해요 343	친절해요 296
1085	음식이 맛있어요 840	매장이 넓어요 286	단체모임 하기 좋아요 280	가성비가 좋아요 254	재료가 신선해요 244
1097	음식이 맛있어요 180	재료가 신선해요 79	매장이 넓어요 53	친절해요 46	특별한 메뉴가 있어요 26

Part 3-2 데이터 마트 적재(rank_review)

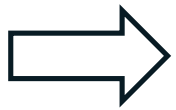
수행절차 및 방법

```
rank_df_1 = rank_review.selectExpr("id", "stack(5, 're_rank1', re_rank1, 're_rank2', re_rank2, 're_rank3', re_rank3, 're_rank4', re_rank4, 're_rank5', re_rank5) as (rank, value)")
rank_df_1.show(truncate=False)
```

```
rank_df_1 = rank_df_1.drop("rank")
rank_df_1 = rank_df_1.withColumn("text", regexp_replace(rank_df_1["value"], r'₩d+', ''))
rank_df_1 = rank_df_1.withColumn("cnt", regexp_extract(rank_df_1["value"], r'(₩d+)', 1).cast("int"))
```

- 'value' 컬럼을 기반으로 'text'와 'cnt' 컬럼 생성

id	rank	value
1088	re_rank1	음식이 맛있어요 404
1088	re_rank2	매장이 넓어요 170
1088	re_rank3	양이 많아요 141
1088	re_rank4	단체모임 하기 좋아요 131
1088	re_rank5	가성비가 좋아요 125
1092	re_rank1	음식이 맛있어요 230
1092	re_rank2	재료가 신선해요 164
1092	re_rank3	매장이 청결해요 159
1092	re_rank4	친절해요 147
1092	re_rank5	특별한 메뉴가 있어요 80
1105	re_rank1	음식이 맛있어요 117
1105	re_rank2	친절해요 80
1105	re_rank3	재료가 신선해요 67
1105	re_rank4	특별한 메뉴가 있어요 48
1105	re_rank5	매장이 청결해요 38
1081	re_rank1	음식이 맛있어요 1182
1081	re_rank2	친절해요 697
1081	re_rank3	재료가 신선해요 672
1081	re_rank4	매장이 청결해요 601
1081	re_rank5	가성비가 좋아요 482



id	text	cnt
1088	음식이 맛있어요	404
1088	매장이 넓어요	170
1088	양이 많아요	141
1088	단체모임 하기 좋아요	131
1088	가성비가 좋아요	125
1092	음식이 맛있어요	230
1092	재료가 신선해요	164
1092	매장이 청결해요	159
1092	친절해요	147
1092	특별한 메뉴가 있어요	80
1105	음식이 맛있어요	117
1105	친절해요	80
1105	재료가 신선해요	67
1105	특별한 메뉴가 있어요	48
1105	매장이 청결해요	38
1081	음식이 맛있어요	1182
1081	친절해요	697
1081	재료가 신선해요	672
1081	매장이 청결해요	601
1081	가성비가 좋아요	482

Rank 컬럼은 삭제
리뷰와 리뷰수를 나누기 위해
Value를 text와 cnt로 나눔

Part 3-2 # 데이터 마트 적재(codezip)

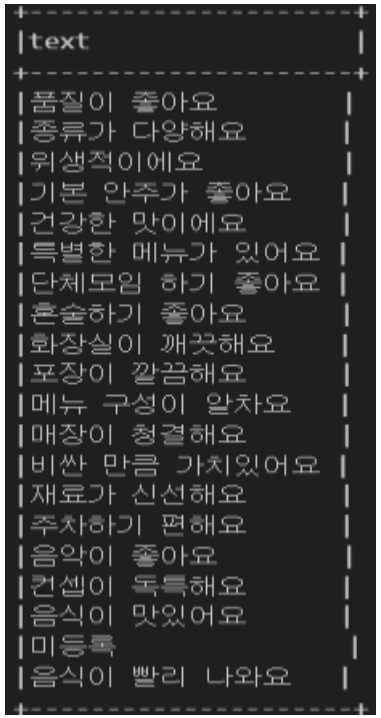
수행절차 및 방법

```
unique_texts = rank_df_1.select("
distinct_texts = rank_df_1.select("
distinct_texts = distinct_texts.with
codezip = distinct_texts.select("c
```

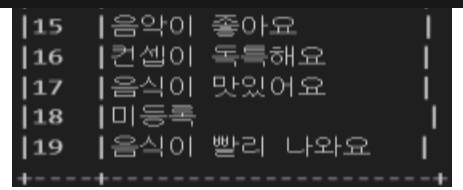
creasing id 함수를 사용하여 각 행에 고유한 ID 부여
용하여 codezip 테이블 생성

```
+----------------------+
| Tables_in_etlmysqlDM |
+----------------------+
| codezip              |
| food_info_with_taboo |
| rank_review          |
| restaurant_info      |
| visitor_review       |
+----------------------+
5 rows in set (0.00 sec)
```

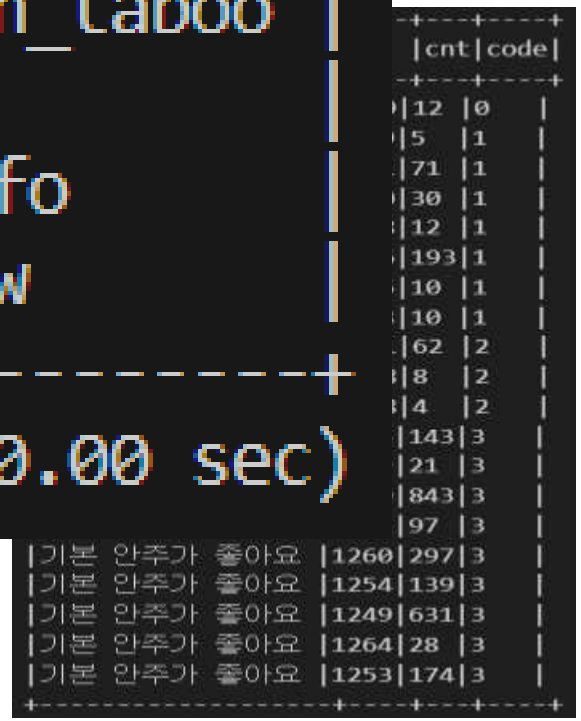
Rank_review



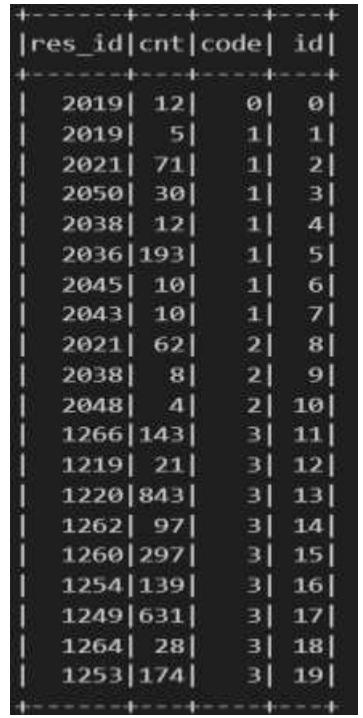
text
품질이 좋아요
종류가 다양해요
위생적이에요
기본 안주가 좋아요
건강한 맛이에요
특별한 메뉴가 있어요
단체모임 하기 좋아요
혼술하기 좋아요
화장실이 깨끗해요
포장이 깔끔해요
메뉴 구성이 알차요
매장이 청결해요
비싼 만큼 가치있어요
재료가 신선해요
주차하기 편해요
음악이 좋아요
컨셉이 독특해요
음식이 맛있어요
미등록
음식이 빨리 나와요



15	음악이 좋아요
16	컨셉이 독특해요
17	음식이 맛있어요
18	미등록
19	음식이 빨리 나와요



		cnt	code
		12	0
		5	1
		71	1
		30	1
		12	1
		193	1
		10	1
		10	1
		62	2
		8	2
		4	2
		143	3
		21	3
		843	3
		97	3
기본 안주가 좋아요	1260	297	3
기본 안주가 좋아요	1254	139	3
기본 안주가 좋아요	1249	631	3
기본 안주가 좋아요	1264	28	3
기본 안주가 좋아요	1253	174	3



res_id	cnt	code	id
2019	12	0	0
2019	5	1	1
2021	71	1	2
2050	30	1	3
2038	12	1	4
2036	193	1	5
2045	10	1	6
2043	10	1	7
2021	62	2	8
2038	8	2	9
2048	4	2	10
1266	143	3	11
1219	21	3	12
1220	843	3	13
1262	97	3	14
1260	297	3	15
1254	139	3	16
1249	631	3	17
1264	28	3	18
1253	174	3	19

Part 3-2 데이터 마트 - ERD

데이터 마트에 적재된 테이블 간 관계 설정 위한 ERD 설계

Part 3-3

딥러닝 학습(데이터 수집)

수행절차 및 방법

40종의 한식 선정 후, 각 1000장씩 이미지 데이터를 AI-Hub을 통해서 수집

딥러닝 학습(CNN)

수행절차 및 방법

```
all_images = os.listdir(class_dir)
```

- 해당 클래스의 모든 이미지 파일을 리스트로 가져옴

```
train_images, test_images = train_test_split(all_images,
                                test_size=0.2, random_state=42)
test_images, validation_images = train_test_split(test_images,
                                    test_size=0.5, random_state=42)
```

- 데이터셋을 훈련, 테스트, 검증 데이터로 니눔
- Train, Test, validation 비율은 0.8 : 0.1 : 0.1

```
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 15,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    shear_range = 0.2,
    zoom_range = 0.3,
    horizontal_flip = True,
    fill_mode = 'nearest'
)
```

- 학습 데이터 이미지 증강
- **1/255**로 스케일링하여 **0-1** 범위로 변환
- 이미지 회전 범위 **(15)**
- 그림을 수평 또는 수직으로 랜덤하게 평행 이동시키는 범위
- 임의 전단 변환 **(0.2)**
- 임의 확대/축소 범위 **(0.3)**
- **True**로 설정할 경우, **50%** 확률로 이미지를 수평으로 뒤집음
- 이미지를 회전, 이동하거나 축소할 때 생기는 공간을 채우는 방식

```
val_datagen = ImageDataGenerator(rescale = 1./255)
test_datagen = ImageDataGenerator(rescale= 1./255)
```

- 검증, 테스트 데이터는 증강 사용하지 않음

수행절차 및 방법

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size = (224, 224),
    batch_size = 16,
    seed = 42,
    class_mode = 'categorical')
```

- Train_generator 설정
- Target_size는 (224, 224)
- Batch_size는 16
- Categorical로 지정하여 2D one-hot 부호화된 라벨이 반환
- Test, validation도 이와 같이 설정

```
model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(32, (3,3), activation='relu',
                               input_shape=(224, 224, 3)),
        tf.keras.layers.MaxPooling2D((2, 2)),
```

- 첫 번째 합성곱 층
- 입력 이미지 크기 224 X 224
- 필터 수 32
- 풀 크기 2 X 2

```
        tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D((2, 2)),
```

- 두 번째 합성곱 층
- 필터 수 32, 풀 크기 2 X 2

```
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D((2, 2)),
```

- 세 번째 합성곱 층
- 필터 수 64

```
        tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.MaxPooling2D((2, 2)),
```

- 네 번째 합성곱 층
- 필터 수 128, 과대 적합 방지 드롭아웃 레이어 추가

```
        tf.keras.layers.Flatten(),
```

- 밀집 층에 전달하기 위해 펼침

```
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dense(len(class_names), activation='softmax')
])
```

- 512개 뉴런을 가진 은닉층
- 다중 분류이므로 activatio은 softmax

```
model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
              metrics=['accuracy'])
```

- 분류 이므로 손실 함수는 categorical_crossentropy
- RMS를 이용하여 최적화
- 평가지표는 정확도

```
MODEL_DIR = './model/'
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)

modelpath="./model/{epoch:02d}-{val_loss:.4f}.hdf5"
```

- 모델 저장 : model 디렉터리 생성
- model 안에 파일로 저장

```
checkpointer = ModelCheckpoint(filepath = modelpath,
                               monitor = 'val_loss',
                               verbose = 1,
                               save_best_only = True),
early_stopping_clbk = EarlyStopping(monitor = 'val_loss', patience = 50)

history = model.fit(train_generator,
                    validation_data = validation_generator,
                    epochs = 100
                    ,callbacks = [early_stopping_clbk, checkpointer])
```

- 기준은 val_loss으로 설정
- 모델 성능이 좋은 것만 저장
- 에폭을 100으로 설정한 후, 50번이 지나도 성능이 나아지지 않으면 종료

딥러닝 학습(CNN)

CNN은 입력 계층, 출력 계층, 그리고 그 사이의 여러 은닉 계층으로 구성
해당 데이터의 고유한 특징을 학습한다는 의도로 데이터를 변경시키는 연산을 수행

CNN 딥러닝 모델의 경우 정확도가 낮아, 다른 모델 탐색

```
Epoch 31: val_loss did not improve from 2.08335
2000/2000 [==============================] - 177s 89ms/step - loss: 1.2302 - accuracy: 0.7944 - val_loss: 14.1981 - val_accuracy: 0.3943
Epoch 32/100
2000/2000 [==============================] - ETA: 0s - loss: 1.2969 - accuracy: 0.7909
Epoch 32: val_loss did not improve from 2.08335
2000/2000 [==============================] - 186s 93ms/step - loss: 1.2969 - accuracy: 0.7909 - val_loss: 9.8451 - val_accuracy: 0.4035
Epoch 33/100
2000/2000 [==============================] - ETA: 0s - loss: 1.3216 - accuracy: 0.7972
Epoch 33: val_loss did not improve from 2.08335
2000/2000 [==============================] - 197s 99ms/step - loss: 1.3216 - accuracy: 0.7972 - val_loss: 15.3711 - val_accuracy: 0.3988
Epoch 34/100
2000/2000 [==============================] - ETA: 0s - loss: 1.2942 - accuracy: 0.8025
Epoch 34: val_loss did not improve from 2.08335
2000/2000 [==============================] - 202s 101ms/step - loss: 1.2942 - accuracy: 0.8025 - val_loss: 9.6474 - val_accuracy: 0.3747
Epoch 35/100
2000/2000 [==============================] - ETA: 0s - loss: 1.3506 - accuracy: 0.7974
Epoch 35: val_loss did not improve from 2.08335
2000/2000 [==============================] - 204s 102ms/step - loss: 1.3506 - accuracy: 0.7974 - val_loss: 21.3390 - val_accuracy: 0.4003
```

Part 3-3 딥러닝 학습(InceptionV3)

수행절차 및 방법

```
img_size = (299, 299)
batch_size = 32
```

- 이미지 (299, 299)
- Batch_size는 32로 설정

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)
```

- 학습 데이터 이미지 증강
- 1/255로 스케일링하여 0-1 범위로 변환
- 임의 전단 변환 (0.2)
- 임의 확대/축소 범위 (0.2)
- True로 설정할 경우, 50% 확률로 이미지를 수평으로 뒤집음

```
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=img_size,
                                                    batch_size=batch_size,
                                                    class_mode='categorical')
```

- Train_generator 설정
- Categorical로 지정하여 2D one-hot 부호화된 라벨이 반환
- Test, validation도 이와 같이 설정

```
base_model = InceptionV3(input_shape=(299, 299, 3),
                         include_top=False,
                         weights='imagenet')
```

- InceptionV3 모델 불러오기
- Fully Connected Layer를 포함하지 않음
- 이미지넷으로 트레이닝 된 모델을 불러옴

```
x = base_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(1024, activation='relu')(x)
predictions = layers.Dense(len(class_names), activation='softmax')(x)
```

- 새로운 Fully Connected Layer 추가

```
model = Model(inputs=base_model.input, outputs=predictions)

model.compile(optimizer=Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

- 새로운 모델 생성

딥러닝 학습(InceptionV3)


```
checkpoint_filepath = './model/inceptionv3_checkpoint.h5'
model_checkpoint = ModelCheckpoint(checkpoint_filepath,
                                   monitor='val_accuracy',
                                   save_best_only=True,
                                   mode='max',
                                   verbose=1)
```

- 모델 체크포인트 설정
- Monitor가 val_accuracy라 mode는 max로 설정

```
if os.path.exists(checkpoint_filepath):
    model.load_weights(checkpoint_filepath)
    print("Checkpoint loaded successfully!")
```

- 저장된 체크포인트 불러오기

```
history = model.fit(train_generator,
                    epochs=100,
                    initial_epoch=0,
                    validation_data=validation_generator,
                    callbacks=[model_checkpoint])

# 모델 평가
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test Accuracy: {test_acc}')

# 모델 저장
model.save('./model/inceptionv3_model_final.h5')
```

- 모델 훈련
- 에폭은 100으로 설정
- 중간에 멈췄다면 마지막 에폭부터 다시 시작

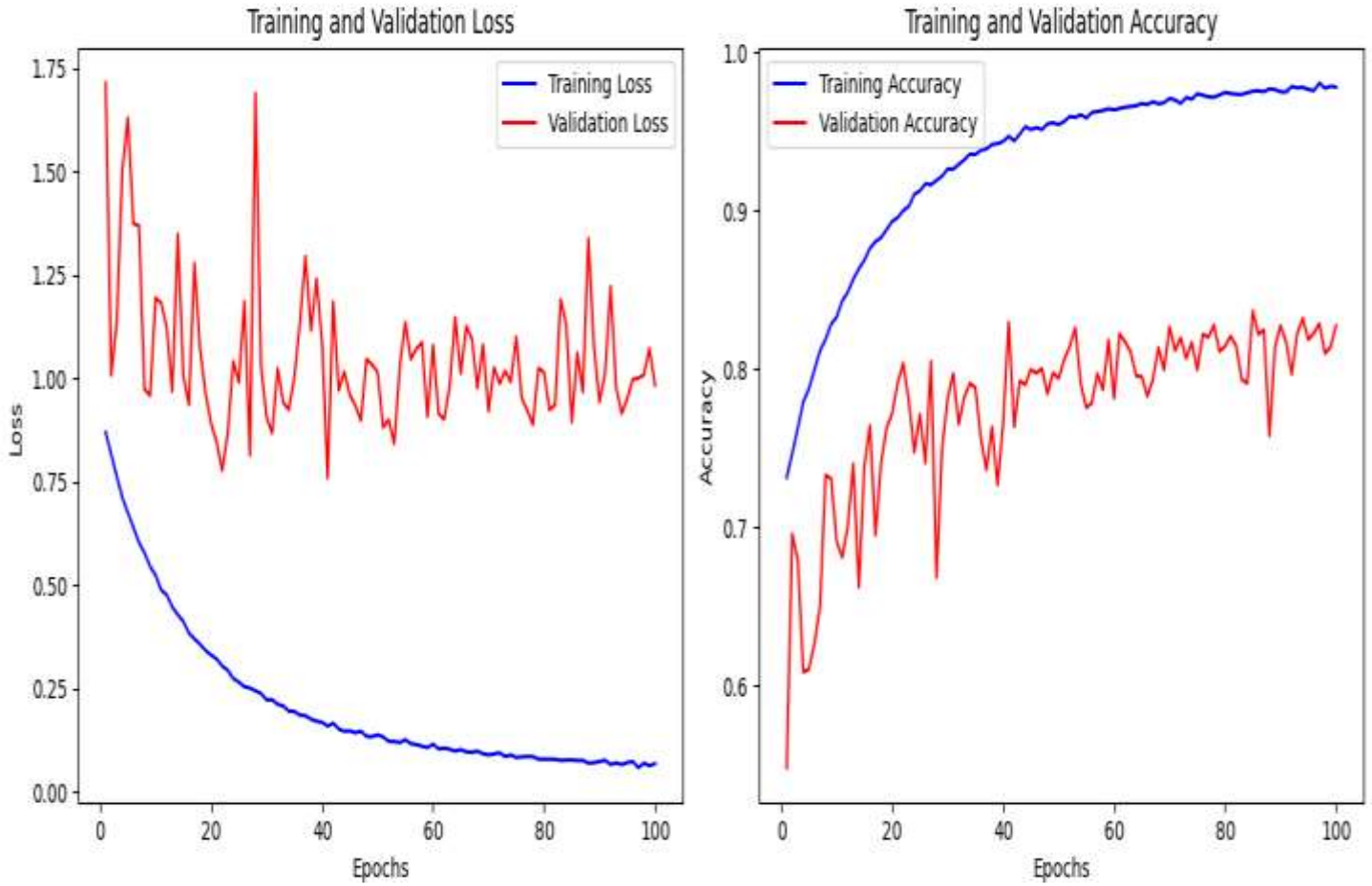
딥러닝 학습(InceptionV3)

Epoch 100으로 설정 후 모델 학습 결과 -
85번 째에서 0.83으로 가장 높은 정확도를 보여줌

```
Epoch 00082: val_accuracy did not improve from 0.82956
Epoch 83/100
1000/1000 [==============================] - 583s 583ms/step - loss: 0.0754 - accuracy: 0.9735 - val_loss: 1.1908 - val_accuracy: 0.7935

Epoch 00083: val_accuracy did not improve from 0.82956
Epoch 84/100
1000/1000 [==============================] - 584s 583ms/step - loss: 0.0764 - accuracy: 0.9745 - val_loss: 1.1319 - val_accuracy: 0.7910

Epoch 00084: val_accuracy did not improve from 0.82956
Epoch 85/100
1000/1000 [==============================] - 583s 583ms/step - loss: 0.0767 - accuracy: 0.9755 - val_loss: 0.8935 - val_accuracy: 0.8368

Epoch 00085: val_accuracy improved from 0.82956 to 0.83678, saving model to ./model\inceptionv3_checkpoint.h5
Epoch 86/100
1000/1000 [==============================] - 584s 583ms/step - loss: 0.0754 - accuracy: 0.9758 - val_loss: 1.0616 - val_accuracy: 0.8218

Epoch 00086: val_accuracy did not improve from 0.83678
Epoch 87/100
1000/1000 [==============================] - 583s 583ms/step - loss: 0.0755 - accuracy: 0.9752 - val_loss: 0.9669 - val_accuracy: 0.8248

Epoch 00087: val_accuracy did not improve from 0.83678
Epoch 88/100
1000/1000 [==============================] - 583s 583ms/step - loss: 0.0690 - accuracy: 0.9770 - val_loss: 1.3388 - val_accuracy: 0.7579
```

딥러닝 학습(InceptionV3)


```
# 훈련 중에 기록된 손실값과 정확도 가져오기
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

# 에폭 수 가져오기
epochs = range(1, len(train_loss) + 1)

# 손실값 그래프
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, train_loss, 'b-', label='Training Loss')
plt.plot(epochs, val_loss, 'r-', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# 정확도 그래프
plt.subplot(1, 2, 2)
plt.plot(epochs, train_acc, 'b-', label='Training Accuracy')
plt.plot(epochs, val_acc, 'r-', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```



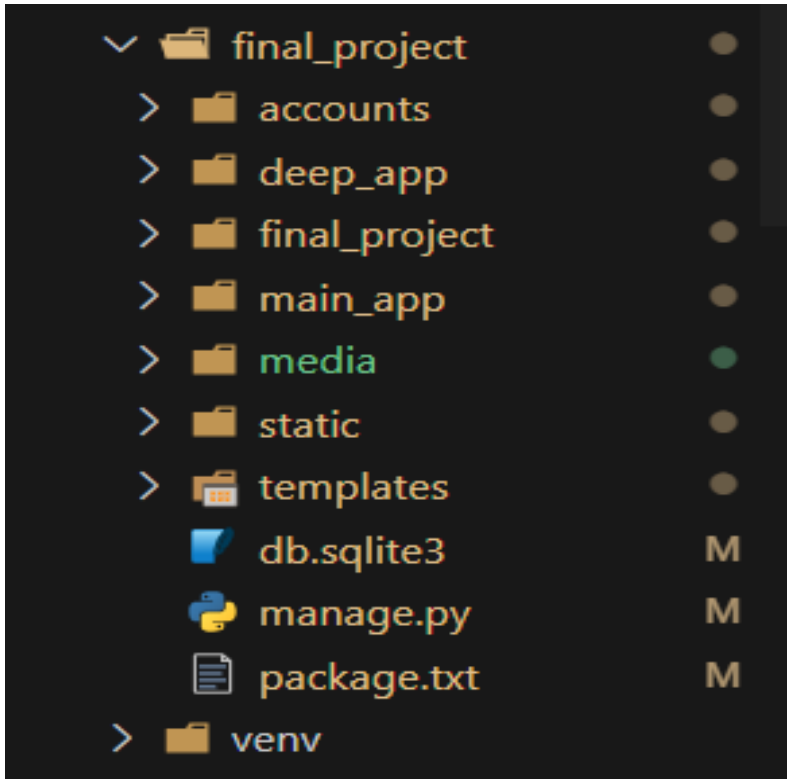
4

웹 구현

Part 4

웹 구현(App)

Django service



Final_project라는 Django 프로젝트 생성
Accounts : 로그인/아웃, 회원가입 관리하는 app
Deep_app : 이미지를 호출 받으면 분석해주는 주요 app
Main_app : 메인 페이지 기초

웹 구현(App)

Deep_app > deepRouter.py

```python
class DeepRouter:
    route_app_labels = {'deep_app'}
    db_name = 'datamart'

    def db_for_read(self, model, **hints):
        if model._meta.app_label in self.route_app_labels:
            return self.db_name
        return None

    def db_for_write(self, model, **hints):
        if model._meta.app_label in self.route_app_labels:
            return self.db_name
        return None

    def allow_relation(self, obj1, obj2, **hints):
        if (
            obj1._meta.app_label in self.route_app_labels or
            obj2._meta.app_label in self.route_app_labels
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if app_label in self.route_app_labels:
            return self.db_name
        return None
```

- DeepRouter: 데이터베이스 라우팅을 수행하는 클래스
- db_for_read: 읽기 작업(read)에 대한 데이터베이스를 결정
- db_for_write: 쓰기 작업(write)에 대한 데이터베이스를 결정
- allow_relation : 두 모델 간의 관계를 허용할지 여부를 결정
- allow_migrate : 마이그레이션을 허용할지 여부를 결정

웹 구현(App)

Deep_app > models.py

```python
class Codezip(models.Model):
    code = models.IntegerField(primary_key=True)
    text = models.TextField(blank=True, null=True)

    class Meta:
        managed = False
        app_label = 'deep_app'
        db_table = 'codezip'
```

•Codezip: 리뷰코드(int)와 해당 코드에 대한 설명 (ex. 맛있어요, 가성비있어요)

```python
class FoodInfoWithTaboo(models.Model):
    food = models.CharField(primary_key=True, max_length=255)
    info = models.TextField(blank=True, null=True)
    ingredient = models.TextField(blank=True, null=True)
    taboo_ingredient = models.CharField(max_length=255, blank=True, null=True)

    class Meta:
        managed = False
        app_label = 'deep_app'
        db_table = 'food_info_with_taboo'
```

•FoodInfoWithTaboo: 40가지 음식에 대한 정보.
• 음식종류, 음식정보, 음식에 들어간 재료,
• 음식에 포함된 주의해야할 재료

```python
class RankReview(models.Model):
    res = models.ForeignKey('RestaurantInfo', models.DO_NOTHING, blank=True, null=True)
    cnt = models.IntegerField(blank=True, null=True)
    code = models.ForeignKey(Codezip, models.DO_NOTHING, db_column='code', blank=True, null=True)
    id = models.BigIntegerField(primary_key=True)

    class Meta:
        managed = False
        app_label = 'deep_app'
        db_table = 'rank_review'
```

•RankReview: 리뷰 정보. 음식점 이름,
•리뷰 숫자, 리뷰코드, id

Part 4

웹 구현(App)

Django service

Deep_app > models.py

```
class RestaurantInfo(models.Model):
    addr = models.TextField(blank=True, null=True)
    food = models.ForeignKey(FoodInfoWithTaboo, models.DO_NOTHING, db_column='food', blank=True, null=True)
    store_name = models.TextField(blank=True, null=True)
    tel = models.CharField(max_length=255, blank=True, null=True)
    id = models.IntegerField(primary_key=True)
    x = models.FloatField(blank=True, null=True)
    y = models.FloatField(blank=True, null=True)

    class Meta:
        managed = False
        app_label = 'deep_app'
        db_table = 'restaurant_info'
```

•RestaurantInfo: 음식점의 정보. 주소,
•음식종류, 가게이름, 전화번호,
• x좌표, y좌표 로 구성

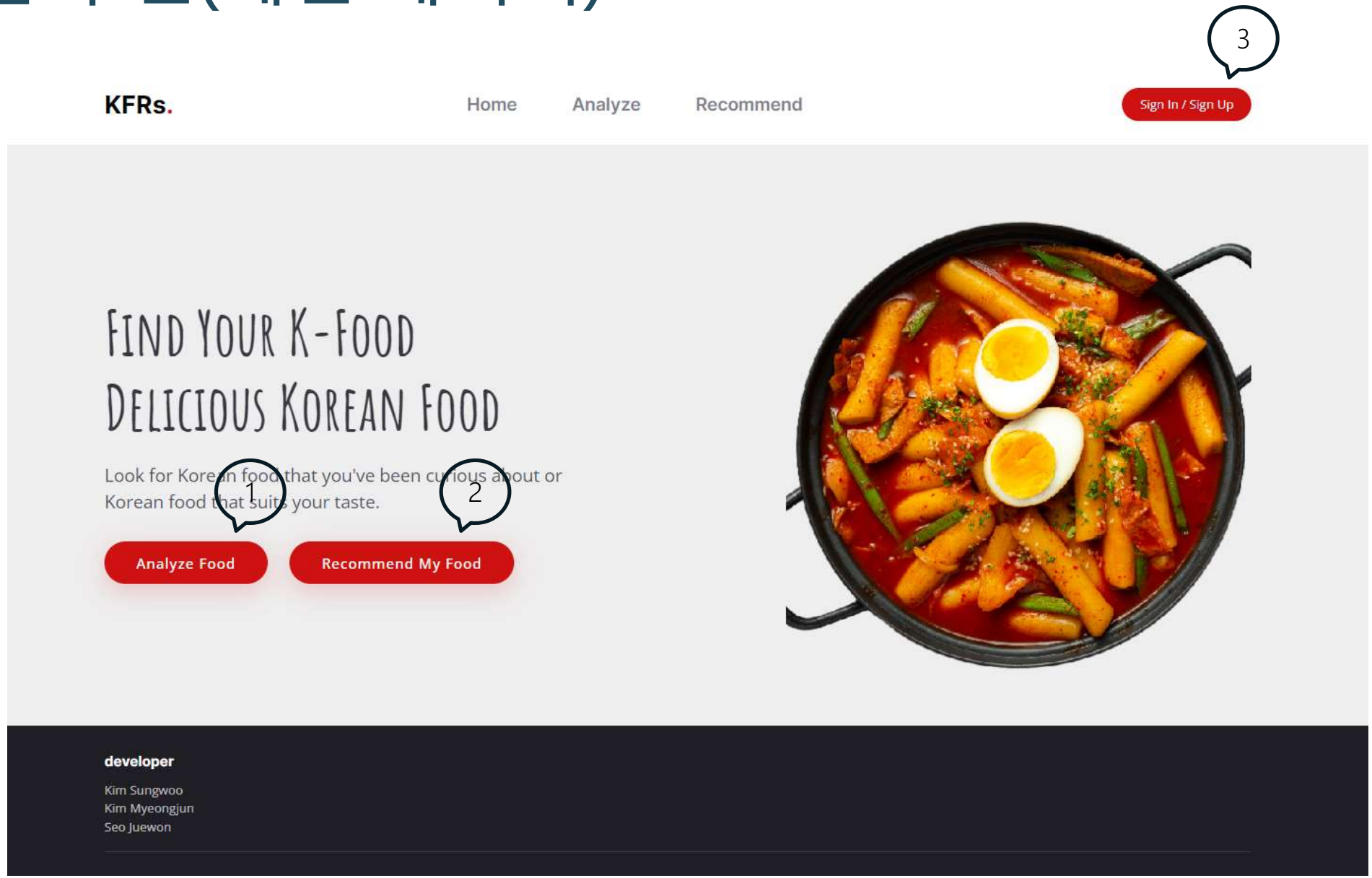
```
class VisitorReview(models.Model):
    id = models.OneToOneField(RestaurantInfo, models.DO_NOTHING, db_column='id', primary_key=True)
    re_visitor = models.TextField(blank=True, null=True)

    class Meta:
        managed = False
        app_label = 'deep_app'
        db_table = 'visitor_review'
```

- VisitorReview: 실제 방문자리뷰.
- ID와 리뷰 내용으로 구성

Part 4

웹 구현(메인 페이지)

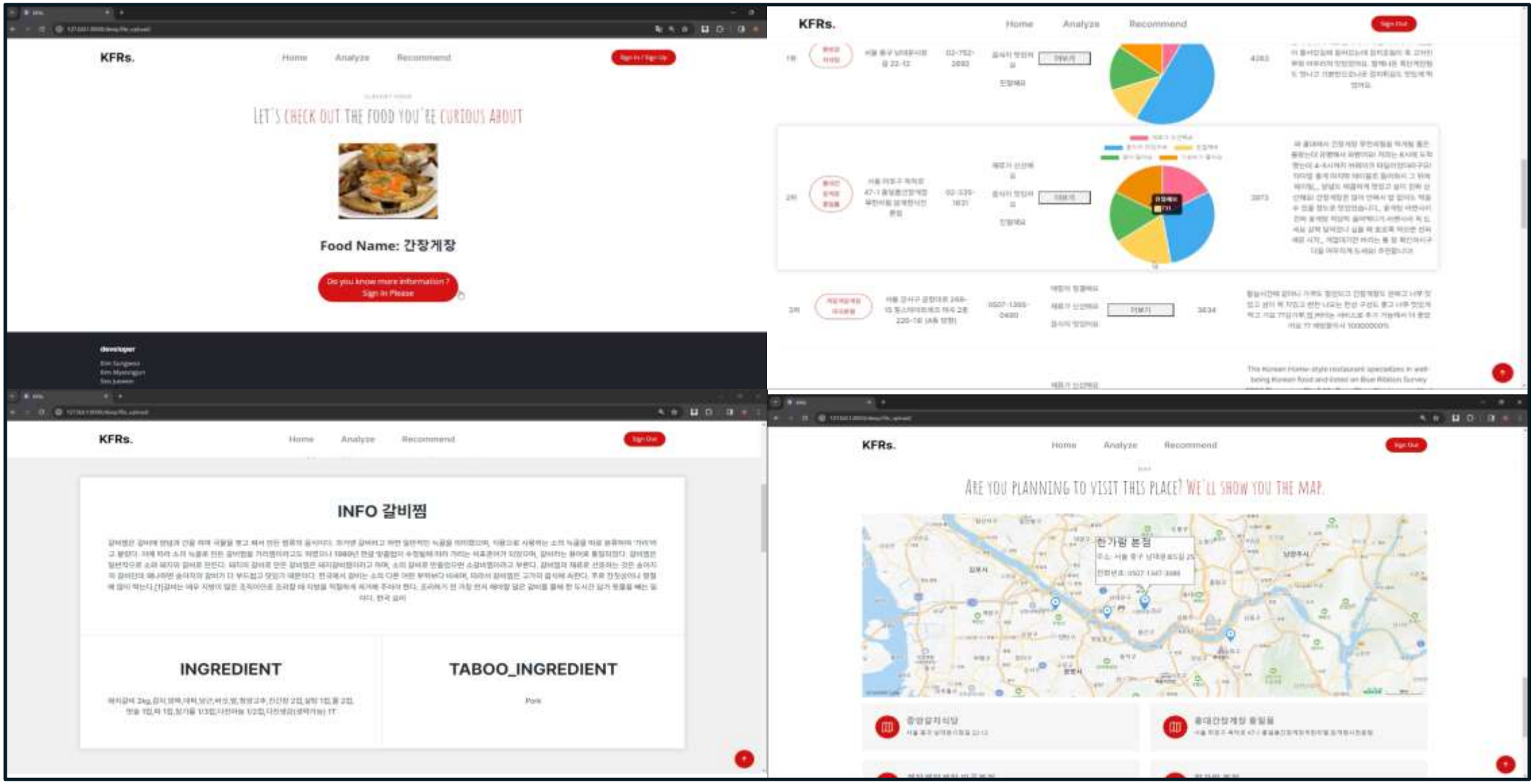
Django service



Part 4

웹 구현(Analyze Food)

Django service



Part 4

웹 구현(Analyze Food)

Django service

Deep_app > views.py

```
def input_image(request):
    return render(request, 'deep_app/input_image.html')
```

```
def file_upload(request) :
    if request.method == "POST" :
        file = request.FILES.get('imgFile')
        fs = FileSystemStorage()
        fs.save(file, file)
        # 모든 분석과 관련된 모듈은 deep_app>modules>data_ana.py
        class_name = image_classify(file)
        img_name = file.name
        print(f"저장한 사진: {img_name}")

        food_info = FoodInfoWithTaboo.objects.get(food=class_name)
        food = food_info.food
        info = food_info.info
        ingredient = food_info.ingredient
        taboo_ingredient = food_info.taboo_ingredient
```

- Request.FILES.get() 클라이언트 전송한 파일 추출하는 코드
- FileSystemStorage() : 파일 저장 객체 생성
- Fs.save() : 전송된 파일을 저장

- 분류예측 모듈 호출 결과 반환, 이미지 분류 카테고리명을 반환
- 클라이언트 전송한 이미지 이름

- 음식이름, 정보, 재료, taboo 가져오기

```
restaurant_info_list = RestaurantInfo.objects.filter(food=food_info)
rank_review_list = RankReview.objects.filter(res__in=restaurant_info_list)

count_sum_by_res = {}

for rank_review_instance in rank_review_list:
    res_value = rank_review_instance.res
    cnt_value = rank_review_instance.cnt
```

- 추출된 음식의 각 음식점별로 추천 수 더하기

웹 구현(Analyze Food)

Django service

Deep_app > views.py

```
    if res_value is not None and cnt_value is not None:
        # res_value가 이미 딕셔너리에 있으면 기존 값에 더하기, 없으면 추가
        if res_value in count_sum_by_res:
            count_sum_by_res[res_value] += cnt_value
        else:
            count_sum_by_res[res_value] = cnt_value

    # 추천수 많은 상위 5개 음식점 추출
sorted_res_by_count = sorted(count_sum_by_res.items(), key=itemgetter(1), reverse=True)
top_5_res = sorted_res_by_count[:5]
```

- None이 아닌 경우에만 연산
- res_value가 이미 딕셔너리에 있으면 기존 값에 더하기, 없으면 추가

- 추천수 많은 상위 5개 음식점 추출

```
    # 상위 5개 음식점의 정보 추출
    rank5_info_list = []
    for rank, (res_value, count_sum) in enumerate(top_5_res, start=1):
        restaurant_info_instance = res_value
        visitor_review_instance = VisitorReview.objects.get(id=restaurant_info_instance.id)

        rank_review_instances = RankReview.objects.filter(res=restaurant_info_instance)

        code_texts = []
        cnt_values = []

        for rank_review_instance in rank_review_instances:
            code_text =      Codezip.objects.get(code=rank_review_instance.code.code).text
            code_texts.append(code_text)
            cnt_values.append(rank_review_instance.cnt)
```

- 상위 5개 음식점의 정보 추출

Part 4

웹 구현(Analyze Food)

Django service

Deep_app > views.py

```
        rank5_info_list.append({
            'rank': rank,  # 순위 추가
            'addr': restaurant_info_instance.addr,
            'store_name': restaurant_info_instance.store_name,
            'tel': restaurant_info_instance.tel,
            're_visitor': visitor_review_instance.re_visitor,
            'x': restaurant_info_instance.x,
            'y': restaurant_info_instance.y,
            'code_texts': code_texts,
            'cnt_values': cnt_values,
            'cnt_tot': count_sum})

    context = { # 클라이언트 페이지에 전달된 dic(이미지 분류명, 이미지 파일명)
        'class_name' : class_name,
        'img_name' : img_name,
        'food' : food,
        'info' : info,
        'ingredient' : ingredient,
        'taboo_ingredient' : taboo_ingredient,
        'rank5_info_list' : rank5_info_list,}
return render(request, 'deep_app/result.html', context)
```

•리뷰 상위 5개에 해당하는 음식점 정보를 rank5_info_list에 넣고 처리한 모든 데이터를 context에 담아 html로 전달.

Part 4

웹 구현(Analyze Food)

Django service

Deep_app > data_anal.py

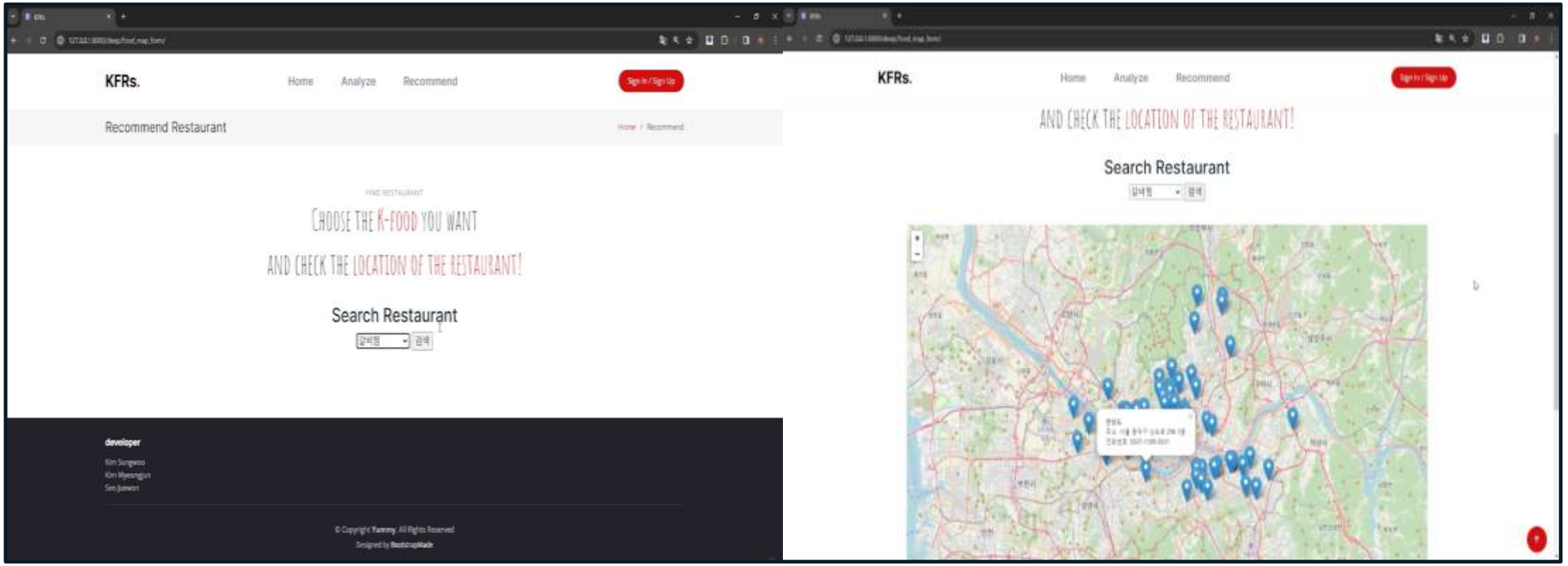
```
# 한 개의 이미지에 대해 분류 예측 후 분류 카테고리를 반환하는 함수
def image_classify(file):
    file_path = settings.BASE_DIR / 'deep_app' / 'modules'
    loaded_model = load_model(file_path / 'inceptionv3_model_final.h5') # 사용모델 로딩, 모델이 저장되어 있는 경로에 한글이 있으면 에러발생

    # 이미지 전처리
    imgs = []

    image_size = 299  # 학습할때 사용한 이미지 사이즈와 동일하게
    img = Image.open(file)  # 이미지 정보 추출
    print(f"Image opened successfully: {img.size}, Mode: {img.mode}")
    img = img.convert("RGB")  # RGB로 색상값 배치 수정
    img = img.resize((image_size, image_size))  # 이미지 사이즈 학습이미지와 동일하게 수정
    data = np.asarray(img) # array 로 변환
    data = data.astype('float32') / 255  # 픽셀 표준화
    imgs.append(data)  # 이미지 데이터 차원으로 변경 : 학습 차원과 동일한 형태로 구성
    imgs = np.array(imgs)  # 리스트를 array로 변환

    pred_prob = loaded_model.predict(imgs)
    # predict_value = np.argmax(pred_prob[0])
    predict_value = pred_prob.argmax()  # 확률이 가장 높은 인덱스 추출
    categories = ['간장게장', '갈비찜', '갈비탕', '감자전', '감자탕', '곱창구이', '김밥', '김치전', '김치찌개',₩
              '닭갈비', '닭볶음탕', '도토리묵', '된장찌개', '떡볶이', '막국수', '물냉면', '물회', '미역국',₩
              '배추김치', '불고기', '비빔냉면', '비빔밥', '삼겹살', '삼계탕', '설렁탕', '순대', '순두부찌개',₩
              '양념게장', '양념치킨', '육회', '잡채', '제육볶음', '족발', '주꾸미볶음', '짜장면', '칼국수',₩
              '파전', '해물찜', '황태구이', '후라이드치킨']

    return categories[predict_value]  # 카테고리 내의 추출된 인덱스에 맞는 결과값 추출
```

image classify: 이미지를 분류하는 함수.
학습시킨 inception3모델을 불러와 이미지를 전처리하여 이미지 예측 결과를 반환.

Part 4

웹 구현(Recommend My Food)

Django service



웹 구현(Recommend My Food)

Django service

Deep_app > data_anal.py

```
def get_res_map(food_name): # 호출시 전달된 서울시 자치구에 대한 지도 시각화 하는 함수
    food_info = FoodInfoWithTaboo.objects.get(food=food_name)
    restaurant_instances = RestaurantInfo.objects.filter(food=food_info)
```

•get_res_map: 전달된 서울시 지도를 시각화 하는 함수
파라미터로 전달받은 음식이름에 해당하는 정보를 가져옴

```
    restaurnat_info_list = []
    for restaurant_instance in restaurant_instances:
        restaurant_info_dict = {
            'store_name' : restaurant_instance.store_name,
            'addr' : restaurant_instance.addr,
            'tel' : restaurant_instance.tel,
            'x' : restaurant_instance.x,
            'y' : restaurant_instance.y
        }
        restaurnat_info_list.append(restaurant_info_dict)

    smap = folium.Map(location=[37.5502, 126.982], zoom_start=11) # 기준 지도 생성
```

- 가게이름, 주소, 전화번호, 좌푯값을 리스트에 담음

```
    for restaurant_info in restaurnat_info_list:
        latitude = restaurant_info['y']  # 레스토랑의 위도
        longitude = restaurant_info['x']  # 레스토랑의 경도
        store_name = restaurant_info['store_name']  # 레스토랑 이름
        addr = restaurant_info['addr']  # 레스토랑 주소
        tel = restaurant_info['tel']  # 레스토랑 전화번호
        popup_html = f"<b>{store_name}</b><br>주소: {addr}<br>전화번호: {tel}"

        folium.Marker(# 마커 추가
            location=[latitude, longitude],
            popup=folium.Popup(popup_html, max_width=300),
            tooltip=store_name
            ).add_to(smap)

    return smap._repr_html_() # html코드를 추출 후 반환, folium 객체가 아닌 html 코드가 반환됨
```

- 리스트에 담아놓은 음식점 지도정보를 반환

Part 4

웹 구현(Recommend My Food)

Django service

Deep_app > views.py

```
def food_map_form(request) :
    food_name_list = []
    food_info_instance = FoodInfoWithTaboo.objects.all()

    for food_info in food_info_instance:
        food = food_info.food
        food_name_list.append(food)

    food_name_list = sorted(food_name_list)
    return render(request, 'deep_app/food_map_form.html', {'food_name_list' : food_name_list})
```

- food_map_form: FoodInfoWithTaboo에 있는 모든 정보를 가져와 정렬 후 food_name_list을 템플릿에 반환

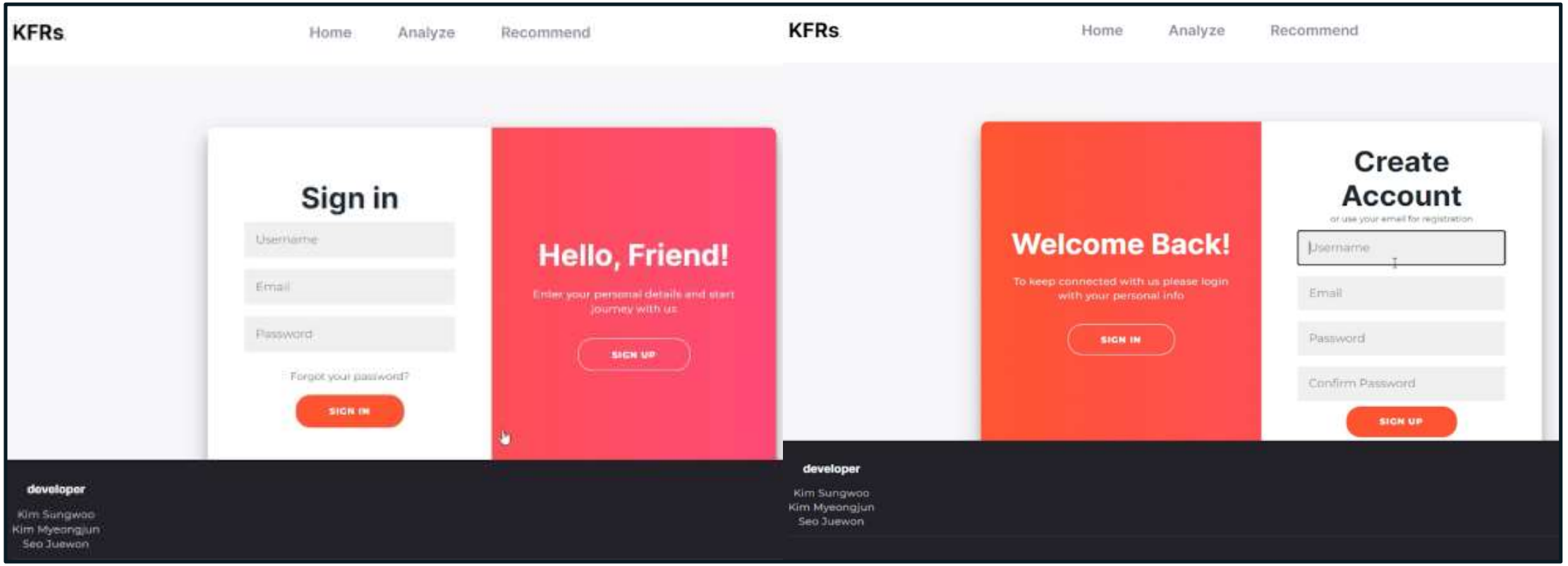
```
def show_res_map(request) :
    if request.method == 'POST' :
        food_name = request.POST['food_name']
        smap = get_res_map(food_name)
    return render(request, 'deep_app/res_map_result.html', {'smap' : smap})
```

- show_res_map: get_res_map을 호출 해 지도정보를 가져오고 화면에 보여줌

Part 4

웹 구현(sign in/sign up)

Django service



Part 4

웹 구현(sign in/sign up)

Django service

Accounts > views.py

```
@require_http_methods(['GET', 'POST'])
def signup_signin(request):
    if request.user.is_authenticated:
        return redirect('main_app:index')
```

•signup_signin: 사용자의 회원가입, 로그인을 처리하는 함수.
만약 사용자가 이미 인증되어 있다면 메인 페이지로 리다이렉트

```
    signup_form = CustomUserCreationForm(request.POST)
    signin_form = AuthenticationForm(request, data=request.POST)
```

•회원가입 폼(CustomUserCreationForm)과
로그인 폼(AuthenticationForm)을 생성

```
    if request.method == 'POST':
        if 'signup' in request.POST and signup_form.is_valid():
            user = signup_form.save()
            auth_login(request, user)
            return redirect('main_app:index')
```

•회원가입의 경우, 새로운 사용자를 생성하고 로그인
상태로 변경한 후 메인 페이지로 리다이렉트

```
        elif 'signin' in request.POST and signin_form.is_valid():
            user = signin_form.get_user()
            auth_login(request, user)
            return redirect('main_app:index')

    return render(request, 'accounts/signup_signin.html', {
        'signup_form': signup_form,
        'signin_form': signin_form,
    })
```

•로그인의 경우, 해당 사용자를 로그인 상태로
변경한 후 메인 페이지로 리다이렉트

```
def signout(request):
    auth_logout(request)
    return redirect('main_app:index')
```

•sigout: 사용자의 로그아웃을 처리하는 함수.

5

결론 및 향후과제

Part 5

결론 및 향후과제

결론

- 이미지 분류 모델의 성능을 비교해 보았을때, CNN보다 **Inception V3** 모델의 성능이 훨씬 더 **향상된것을** 확인
  - 40여개의 음식 class에 대한 분류 정확히 수행함
- 유저 계정 DB와 데이터 DB를 분리시켜 멀티DB를 운용함으로써 **보안성 확장**
- 사용자가 원하는 음식 이미지에 대한 수집된 모든 정보가 웹 서비스에서 문제없이 전달됨을 확인
  - 구축한 데이터 파이프라인에 대한 정상 작동 확인

향후 과제

- 음식 class가 늘어날 경우를 대비해, 보다 성능이 좋은 이미지 분류 모델 모색 방안 필요
- 좀 더 다양한 데이터를 DB에 추가해 통계적인 기능도 추가해야할 필요성을 느낌

Part 5

프로젝트 회고

이름	좋았던 점	아쉬웠던 점
김성우	데이터 수집부터 시작해서 DB에 적재하고 최종적으로 웹을 통해 서비스를 구현하는 큰 과정을 처음 접해보면서 데이터가 중요하다는 것을 느꼈고, 딥러닝 이미지 모델 말고도 동영상도 분석해보고 싶다.	서비스 구현에 있어서 좀 더 세부적인 서비스를 제공하고 싶었지만 웹에 관련하여 지식이 부족한 점이 있어서 추가적으로 해야 할 부분을 못해서 아쉬웠다.
서주원	웹 디자인 및 기능 구현에 대한 이해를 향상시키는 계기가 되었고, 데이터 파이프라인을 실제로 구축해보면서 엔지니어링에 대한 스킬들을 향상시키는 좋은 기회였다.	상대적으로 적은 인력과 능숙하지 못한 엔지니어링 및 웹 구현 스킬로 인해 처음에 계획했었던 기능들을 프로젝트 기간 내에 다 구현하지 못한것이 아쉬웠다.
김명준	딥러닝 모델을 활용하면 많은것을 할 수 있다고 느꼈다. 앞으로 서비스를 개발할 때 적극적으로 AI를 활용할 것 같다. 또한, 프로젝트 전 과정에 참여하다 보니 업무 포지션마다의 어려움점에 대해 잘 알게 되었다.	프론트엔드 실력이 부족하여 준비한것을 모두 보여주지 못한점이 아쉬웠다. 그러다보니 서비스쪽에 시간투자를 많이 하게 되었고, 더 많은 데이터를 다루지 못해서 아쉬웠다.

감사합니다.