**Pusan National University**

**Computer Science and Engineering**

**Technical Report 2022-09**

# 맛집 빅데이터 분석 웹서비스


201724599 최경섭

201724561 장희승

지도교수: 홍봉희 교수님

# 목 차

# 1. 서론

## 1.1. 연구 배경

'빅데이터 시대'는 엄청난 양의 다양한 데이터들을 수집, 가공하여 정치, 사회, 경제, 문화, 과학 등 전 영역에서 새로운 가치를 창출하는 현시대를 뜻한다. 데이터를 사용하여 사회 문제 해결 및 서비스 제공 사례가 증가하고 있다. 이러한 빅데이터 기술을 맛집 추천 시스템에 적용하고자 한다.

최근 평점 시스템의 활성화로 맛집 빅데이터 제공 시스템이 늘어나고 있다. 많은 사람들이 해당 시스템을 이용하고 있지만, 광고 및 악의적 평점 테러의 문제가 시스템의 신뢰성에 문제를 일으키고 있다. 특히 관광지에서 그 현상이 두드러지며 많은 사람들이 시스템에 대한 불신을 가지고 있는 상황이다.

**기계학습을 활용한 허위 리뷰 필터링 작업**을 통해 신뢰할 수 있는 리뷰 및 평점을 제공하고자 한다.

## 1.2. 기존 문제점

현재 맛집 빅데이터 제공 시스템은 모든 사용자의 평점 및 리뷰를 취합하여 제공하는 형태로 구축되어 있다. 많은 정보를 통해 객관적인 평가가 이루어질 수 있지만, 광고 및 악의적 평점 테러와 같은 문제점이 존재한다.

### 1) 평점 필터링 시스템의 부재

광고 및 악의적 평점 테러의 문제는 시스템의 신뢰성에 문제를 일으키고 있다. 지역주민을 주 고객층으로 상대하는 상권보다 타 지역 사람을 상대하는 관광지 주변 상권에서 이러한 문제가 많이 나타난다. 이는 맛집 소개 시스템에 대한 불신으로 이어지고 있다. 따라서 평점 시스템의 신뢰성을 높이기 위해 광고와 평점 테러를 필터링하는 시스템이 필요하다. 하지만 현재 제공되고 있는 시스템에는 해당 기능이 제대로 작동하지 않고 있다. 사용자 개인이 각자의 경험을 통해 직접 필터링하고 있으며, 이는 사용자의 이용 편의성에 악영향을 끼치고 있다.

#### 2) 데이터 분류의 부족함

기존 시스템의 경우 평점, 영업시간을 중심으로 데이터 분류를 진행하고 있다. 하지만 음식점을 이용하는 사람들의 목적은 다양하다. 사용자가 음식점 방문 목적에 맞는 가게를 찾는 것에 많은 시간이 소요되고 이는 결국 사용자의 이용 편의성 저하로 이어지게 된다.

### 1.3. 연구 목표

부산광역시 내 맛집 빅데이터 구축 및 웹 서비스 제공을 목표로 한다. 리뷰 및 평점 필터링 작업을 거쳐 신뢰성 높은 정보를 제공한다. 또한 데이터 분류의 다양화를 목표로 삼는다. 카테고리별 순위, 지역별 순위, 전체 순위를 따로 제공하여 사용자의 편의성을 높인다.

## 2. 연구 배경

### 2.1. 데이터 수집

데이터 수집은 '구글 맵'을 통해 수집했다. '네이버지도'와 '카카오 맵'의 데이터를 함께 수집하면 내용 중복 및 과적합 문제가 발생할 수 있을 것이라 판단하여 데이터 수집을 '구글 맵'으로 한정했다.

### 2.2. 라이브러리

#### 2.2.1. 크롤링 작업

##### 1) Selenium

크롤링 작업 시 웹 구동 원격화 과정에서 동적 동작을 활용하기 위해 Selenium 라이브러리를 활용한다. Selenium은 웹사이트 테스트를 위한 도구로, 브라우저 동작을 자동화할 수 있다. Web driver 프로그램과 연동하여 '동적 크롤링'을 수행한다. 브라우저를 직접 동작 시켜 비동기 적인 컨텐츠를 가져올 수 있으며, 렌더링이 완료된 후의 DOM결과물에 접근이 가능하다는 장점이 있다.

### 2) PyAutoGUI

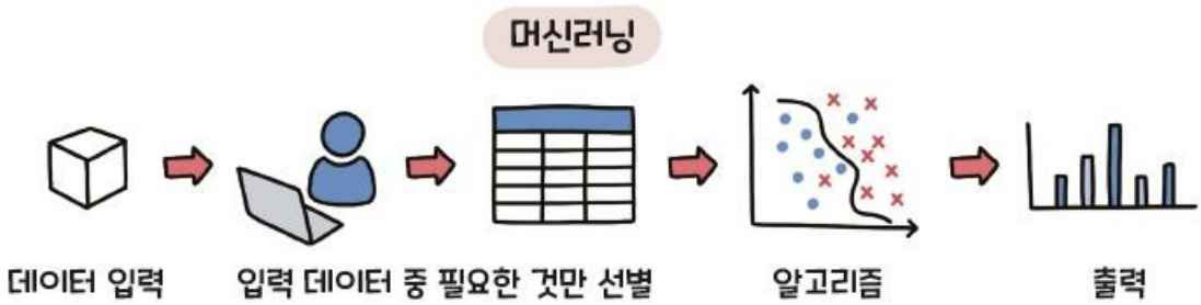
PyAutoGUI 라이브러리는 매크로 프로그램 같은 단순 반복작업의 자동화를 구현하도록 돕는 메소드를 제공한다. 라이브러리를 통해 마우스와 키보드 제어가 가능하며, 응용 프로그램에 마우스 및 키보드 입력을 보내는 기능이 제공된다. 스크롤을 내렸을 때 등장하는 가게들을 크롤링할 수 있도록 지원한다.

### 3) 기타 라이브러리

Pandas 라이브러리는 추후 진행될 감정분석 및 리뷰 필터링 작업에서 활용할 데이터들을 Data Frame형식으로 저장하기 위해 사용된다.

time 라이브러리와 random 라이브러리는 크롤링 과정에서 지연될 시간에 대한 대비를 한다. 의도적인 딜레이를 통해 예상치 못한 오류를 방지한다.

## 2.2.2. 리뷰 필터링




[그림 3] 머신 러닝의 단계별 절차

### 1) Sklearn

Scikit-learn은 머신 러닝 라이브러리다. 텍스트 클렌징과 '가중치 부여를 통한 중요도 분석'을 진행한다.

TF-IDF는 텍스트 마이닝에서 이용하는 가중치로, 중요도를 나타내는 통계적 수치이다. 문서의 핵심어를 추출하는데 주로 이용된다.

TF(증가 빈도): 문서의 길이가 길 경우, 단어 빈도 값을 조절하기 위해 사용

$$tf\ (t,d) = 0.5 + \frac{0.5 \times f(t,d)}{\max\{f(w,d): w \in d\}}$$

IDF(역문서 빈도): 한 단어가 문서 집합 전체에서 공통적으로 나타나는 정도

$$idf\ (t,D) = \log\frac{|D|}{|\{d \in D: t \in d\}|}\quad (|D|: \textit{전체 문서의 수})$$

TF-IDF값은 특정 문서 내에서 단어 빈도가 높고, 전체 문서 중에서 그 단어를 포함하는 문서가 적을수록 높게 형성된다.

$$tfidf\ \ (t,d,D) = tf\ (t,d) \times idf\ (t,D)$$

즉, 특정 단어를 포함하는 문서들이 많을수록 IDF값과 TF-IDF값은 0에 가까워진다.

**2) KoNLPy**

KoNLPy는 한국어 텍스트를 분석할 때 가장 기본적으로 사용되는 형태소 분석기들을 하나로 모은 라이브러리이다. 'Hannanum', 'Kkma', 'Komoran', 'Mecab', 'Okt' 총 5가지의 형태소 분석 방법을 제공한다. 형태소 분석을 통해 일부 형태소의 '리뷰 내 비중'을 파악하고, 이를 리뷰 필터링 작업에 활용한다.

**3) Seaborn**

Seaborn은 데이터 시각화를 수행하는 라이브러리이다. 이번 프로젝트에서는 감정평가 과정에서 heatmap을 활용해 모델 학습 결과의 편향성을 시각적으로 확인한다. 감정 평가 특성상 긍정과 부정, 2개의 카테고리 값에 대한 값 변화를 파악해야 하므로 이를 이용하기 용이하다.

# 3. 연구 내용

## 3.1. 데이터 크롤링

1. 크롤링을 위한 초기 세팅을 한다. 스크롤을 위한 PyAutoGUI, 크롤링을 위한 Selenium 및 Chromedriver 사용을 위해 import하였다.

```
import os
import time
import pandas as pd
import pyautogui
import random as r
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver import ActionChains
```

2. 사용자의 컴퓨터 성능과 인터넷 환경에 따라 로딩 하는 시간이 다를 수 있으므로 "driver.implicitly_wait(10)" 을 통해 10초 내로 로딩이 될 시 바로 넘어가거나 10초를 기다린다.

```
def wait(type_, time_):
    if type_==0:
        driver.implicitly_wait(10)
    time.sleep(time_)
```

3. Get_data함수를 통해 가게 정보를 받아온다. First_content의 value값을 리뷰의 별점과 작성 일자가 적힌 id="DU9Pgb"로 설정한다. 설정한 값을 마우스로 클릭하기 위해 action 변수를 생성한다. 이후 로딩시간을 위해 wait 함수를 사용한다. 로딩이 끝난 후 PyAutoGUI를 이용해 스크롤을 내리는 역할을 한다. Div_reviews 변수를 생성해 리뷰 전체를 선택하는 id="jftiEf"로 설정한다. 리뷰로 이동 후 예외처리를 사용해여 리뷰의 개수를 받아온다. 리뷰의 작성자의 총 리뷰 개수가 3개 미만인 경우 리뷰는 필터링을 한다. 나머지 리뷰들의 별점 및 리뷰 작성일을 가게 데이터에 추가한다. 만약 필터링 후 리뷰가 40개 미만인 경우 가게를 목록에서 제외하였다.

```python
def get_data(driver, temp, current_index, titl, cate, addr):
    # 스크롤 다운
    first_content = driver.find_elements(by=By.CLASS_NAME, value="DU9Pgb")[-1]
    action = ActionChains(driver)
    action.move_to_element(first_content).click().perform()
    wait(1, 1)
    pyautogui.press("pagedown", presses=3, interval=0.2)

    # 자세히 클릭
    try:
        for _ in range(3):
            btn_mores = [i for i in driver.find_elements(by=By.TAG_NAME, value="button") if i.text=="자세히"]
            for btn_more in btn_mores:
                action = ActionChains(driver)
                action.move_to_element(btn_more).click().perform()
                wait(1, 0.2)
            pyautogui.press("pagedown", presses=3, interval=0.2)


    except:
        pass

    div_reviews_ori = driver.find_elements(by=By.CLASS_NAME, value="jftiEf")
    div_reviews = div_reviews_ori[current_index:]

    for div in div_reviews:
        # 리뷰 개수
        try:
            temp_text = div.text
            review_count = int(temp_text[temp_text.index("리뷰")+3:temp_text.index("개")])
        except:
            review_count = 0

        # 댓글
        content = div.find_element(by=By.CLASS_NAME, value="MyEned").text

        # 필터링
        if review_count < 3 or len(content)<10:
            continue

        # 별점, 작성일
        spans = div.find_element(by=By.CLASS_NAME, value="DU9Pgb").find_elements(by=By.TAG_NAME, value="span")
        try:
            score = spans[1].get_attribute("aria-label")
        except:
            score = ""
        try:
            date = spans[2].text
        except:
            date = ""

        # 데이터 추가
        temp.append([titl, cate, addr, score, date, review_count, content])

    # 40개 가져오기
    if len(temp)>=40 or current_index==len(div_reviews_ori): # 리뷰가 20개가 넘거나 더 댓글이 없는 경우
        return temp
    else: # 40개가 안 될 경우
        return get_data(driver, temp, len(div_reviews_ori), titl, cate, addr)
```

4. 구글 맵을 켜기 위한 web driver는 chromedriver를 사용하였다. Chrome창을 최대로 설정 후 구글 맵에 접속해 검색창에 "부산 맛집" 입력 후 검색 버튼을 눌러주었다. 이 때 컴퓨터와 인터넷 상태를 고려해 wait함수를 추가하였다. 가게의 Id값 "hfpxzc" 통해 alist 변수를 생성하였고, 스크롤을 내리며 밑에 있는 가게들의 총 개수를 확인한다. 스크롤을 내릴 때 일정시간 로딩이 지연되면 스크롤을 다시 올린 후, 내려가도록 설계한다. 스크롤을 내리며 마지막까지 진행이 완료된 경우 가게의 전체 개수를 출력 및 저장한다.

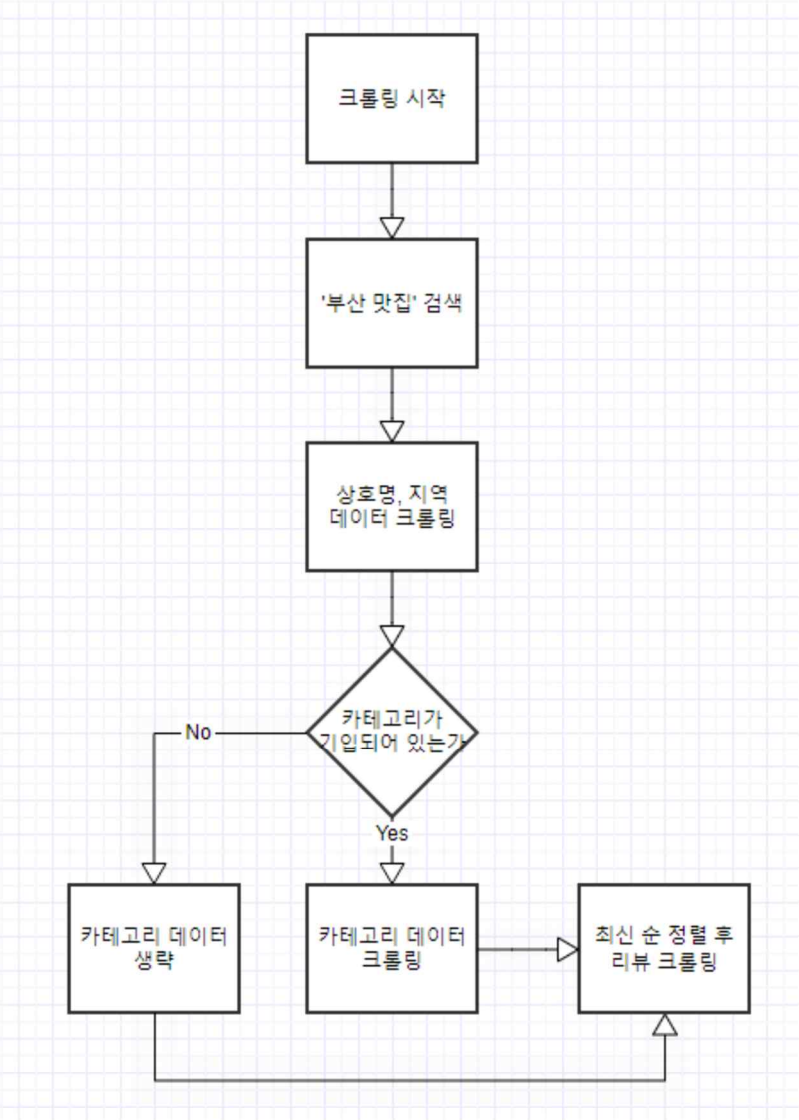
```python
if __name__ == "__main__":
    # set data
    reviews = []

    # go to url
    driver = webdriver.Chrome("./chromedriver.exe")
    driver.maximize_window()
    url = "https://www.google.co.kr/maps/?hl=ko"
    driver.get(url)

    # 부산 맛집 검색
    driver.find_element(by=By.ID, value="searchboxinput").send_keys("부산 맛집")
    wait(1, 1)
    driver.find_element(by=By.ID, value="searchbox-searchbutton").click()
    wait(0, 2)

    # get a tag
    alist = driver.find_elements(by=By.CLASS_NAME, value="hfpxzc")

    # 스크롤 다운
    alist[0].click()
    wait(1, 2)
    alist[0].click()
    wait(1, 1)
    prior = 0
    while True:
        for _ in range(5):
            pyautogui.press("pagedown")
            wait(1, 1+r.random())
        pyautogui.press("pageup")
        wait(0, 2)
        alist = driver.find_elements(by=By.CLASS_NAME, value="hfpxzc")
        if len(alist)==prior:
            break
        else:
            prior = len(alist)
    print("전체 개수: ", len(alist))

    # main loop
    i = 0
    alen = len(alist)
    while True:...
```

5. 가게의 상세정보에 접속하여 리뷰를 클릭한 후, 정렬을 최신순으로 정렬한다. 정렬을 마치면 스크롤을 내리며 리뷰 데이터를 수집 및 추가한다. 데이터 수집을 위해 정보에 맞는 id값을 각각 입력해준다.

```
while True:
    try:
        # 맛집 클릭
        alist = driver.find_elements(by=By.CLASS_NAME, value="hfpxzc")
        alist[i].click()
        wait(1, 2)

        if i!=0 and i%20==0:
            pyautogui.press("pagedown")

        # 상호명, 카테고리, 지역
        title = driver.find_element(by=By.CLASS_NAME, value="fontHeadlineLarge").text
        try:
            category = driver.find_element(by=By.CLASS_NAME, value="uóijk").text
        except:
            category = ''
        address = driver.find_element(by=By.CLASS_NAME, value="IooYTe").text

        # 정렬 클릭
        div = driver.find_element(by=By.CLASS_NAME, value="Hu9e2e")
        div = div.find_element(by=By.CLASS_NAME, value="móQErb")
        btns = div.find_elements(by=By.CLASS_NAME, value="S9kvJb")
        btn_sort = [i for i in btns if i.get_attribute("data-value")=="정렬"][0]
        action = ActionChains(driver)
        action.move_to_element(btn_sort).click().perform()
        wait(1, 1.5)

        # 최신순 클릭
        ul_list = driver.find_elements(by=By.TAG_NAME, value="ul")
        ul = [i for i in ul_list if i.get_attribute("role")=="menu"][0]
        ul.find_elements(by=By.TAG_NAME, value="li")[1].click()
        wait(1, 2)

        # 한 달이내 리뷰 확인
        recent_text = driver.find_element(by=By.CLASS_NAME, value="rsqaWe").text
        if "달" in recent_text and "년" in recent_text:
            i +=1
            continue

        # 스크롤 다운
        first_content = driver.find_element(by=By.CLASS_NAME, value="DU9Pgb")
        action = ActionChains(driver)
        action.move_to_element(first_content).click().perform()

        # 데이터 수집 및 추가
        temp = get_data(driver, [], 0, title, category, address)
        reviews += temp
        #print(len(reviews))
    except Exception as e:
        print(e)
        pyautogui.press("pagedown")
        i -= 1

    i +=1
    if i==alen:
        break
```

6. 전체 가게의 데이터를 수집 완료하였으면 가게의 이름, 카테고리, 주소, 별점, 리뷰 및 리뷰작성 날짜를 받아 .xlsx 형식으로 저장하였다.

```
# data save
df = pd.DataFrame(reviews)
df.columns = ["name", "category", "address", "score", "date", "reviewers_reivew_count", "content"]
df.to_excel("data_result.xlsx")
```




[그림 1] Google_map_crawl.py의 크롤링 Data Flow Chart

| | name | category | address | score | date | ers_reivew | content |
|---|---|---|---|---|---|---|---|
| 0 | 합천일류되 | 돼지국밥 | 부산광역시 | 별표 5개 | 5시간 전 | 31 | 항상 즐겨찾는곳맛있어요 |
| 1 | 합천일류되 | 돼지국밥 | 부산광역시 | 별표 4개 | 1일 전 | 275 | 2022.09맛있네요. 특별함은 없고요.빠르 |
| 2 | 합천일류되 | 돼지국밥 | 부산광역시 | 별표 5개 | 1일 전 | 13 | 깔끔한국물이 최곱니다 |
| 3 | 합천일류되 | 돼지국밥 | 부산광역시 | 별표 5개 | 2일 전 | 60 | 점심시간 맞춰오면 줄이기니 11시 정도 |
| 4 | 합천일류되 | 돼지국밥 | 부산광역시 | 별표 1개 | 3일 전 | 23 | 맛이 변했슴 밥 양이 너무적고 국물도 |
| 5 | 합천일류되 | 돼지국밥 | 부산광역시 | 별표 4개 | 1주 전 | 21 | 좋아하는 국밥집입니다.김치가 맛있어서 |

[표 1] 데이터 수집 항목 및 예제 데이터

### 3.2. 감성분석 및 리뷰 필터링

1. 감성분석을 위한 초기 세팅을 한다. 크롤링을 통해 필터링한 엑셀파일을 Data Frame으로 불러와 작업을 위한 pandas, 머신 러닝을 위한 scikit-learn(sklearn), 형태소 분석을 위한 konlpy, 데이터 시각화를 위한 seaborn 및 matplotlib을 import하였다.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from konlpy.tag import Okt
from collections import Counter
import re
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Content_filter 함수를 통해 기준에 부합하는 리뷰를 제외한다. 먼저 한글 정규 표현식 처리 후에 형태소를 추출한다. 추출한 형태소의 명사, 형용사 비율이 동사, 부사의 비율보다 낮은 경우와 내용이 없는 리뷰 또는 총 길이 10자 미만의 리뷰들을 필터링 하여 제외한다.

```
def content_filter(text):
    hangul = re.compile('[^ ㄱ-ㅣ가-힣]')  # 정규 표현식 처리
    result = hangul.sub('', text)
    okt = Okt()  # 형태소 추출
    pos = [i[1] for i in okt.pos(result)]

    if len(pos)!=0:
        n_na = (pos.count('Noun')+pos.count('Adjective'))/len(pos)
        n_va = (pos.count('Verb')+pos.count('Adverb'))/len(pos)
    else:
        n_na = 0
        n_va = 0

    # 명사, 형용사 비율이 낮고, 동사와 부사의 비율이 높은 경우
    # 내용 없는 리뷰 또는 총 길이 10자 미만 리뷰
    if len(result)<10 or n_na < n_va:
        return True # 필터링 기준에 부합 -> 제외
    return False
```

3. 별점을 숫자로 바꾸기 위한 score_to_int 함수, 별점을 긍정과 부정으로 나누기 위한 rating_to_label 함수 그리고 한 글자 키워드와 불용어를 제거하기 위해 text_cleaning 함수를 만들었다.

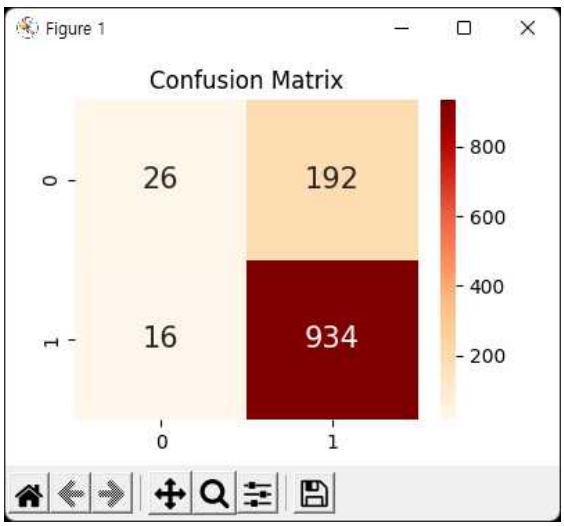
```python
def score_to_int(text):
    return int(text[4:-2])

def text_cleaning(text):
    hangul = re.compile('[^ ㄱ-ㅣ 가-힣]')  # 정규 표현식 처리
    result = hangul.sub('', text)
    okt = Okt()  # 형태소 추출
    nouns = okt.nouns(result)
    nouns = [x for x in nouns if len(x) > 1]  # 한글자 키워드 제거
    nouns = [x for x in nouns if x not in stopwords]  # 불용어 제거
    return nouns

def rating_to_label(rating):
    if rating > 3: # 4, 5점: 긍정
        return 1
    else:
        return 0 # 1~3: 부정
```

4. 머신 러닝을 위한 전처리 작업을 진행한다. 기본적으로 데이터를 df로 불러온다. 불러온 데이터를 content_filter 함수를 이용해 기본 필터링을 진행한다. 기본 필터링 적용 후 불용어 제거 및 불필요 열 제거, 별점에 따라 긍정부정 판단을 위한 별점 숫자로 변환 등 필터링을 적용하였다. TF-IDF 기법을 사용하여 가중치를 부여해 중요한 단어를 잡아내도록 한다.

```python
# 데이터 불러오기
df = pd.read_excel("final_result.xlsx")

# 기본 필터링
df['filtering'] = df['content'].apply(content_filter)
df = df[df['filtering']==False]
df.reset_index(drop=True, inplace=True)

# 별 개수를 숫자로 변환
df['score'] = df['score'].apply(score_to_int)

# 감성 분석을 위해 필요한 열만 남김
df = df[['score', 'content']]
df.columns = ['rating', 'text']# 열 이름 변경

# 불용어 사전
stopwords = pd.read_csv("https://raw.githubusercontent.com/yoonkt200/FastCampusDataset/master/korean_stopwords.txt").values.tolist()
my_stopwords = [] #나의 불용어 추가: 예시 - 노맛

# 텍스트 클렌징
vect = CountVectorizer(tokenizer = lambda x: text_cleaning(x))
bow_vect = vect.fit_transform(df['text'].tolist())
word_list = vect.get_feature_names()
count_list = bow_vect.toarray().sum(axis=0)

# tf-idf
tfidf_vectorizer = TfidfTransformer()
tf_idf_vect = tfidf_vectorizer.fit_transform(bow_vect)

# 별점에 따라 긍정부정 -> 4, 5(긍정) // 1, 2, 3(부정)
df['y'] = df['rating'].apply(lambda x: rating_to_label(x))
```

5. 1차 모델 학습을 진행한다. X축에 CounterVectorizer을 통해 벡터화 한 값과, Y축에 전처리를 통해 저장해 두었던 값을 입력한다. Test_size는 0.3으로, random_state는 1로 지정하여 기본적인 모델 학습을 진행한다. 이 때 과적합을 방지하기 위해 train과 validation으로 데이터를 구분하여 학습을 진행한다. 결과의 정확도, 정밀도, 재현율, F점수를 출력한다. 혼돈행렬을 얻기 위해 변수에 저장 후 히트맵으로 출력한다.

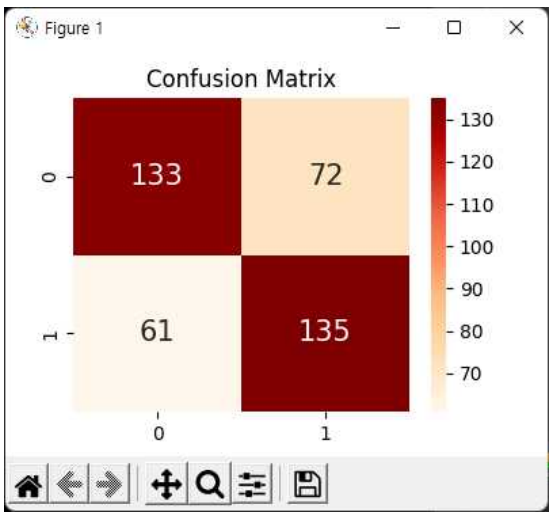
```
### 모델 학습1 ###
x = tf_idf_vect
y = df['y']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=1)

# fit in training set
lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)

# predict in test set
y_pred = lr.predict(x_test)

# classification result for test set
print('accuracy: %.2f' % accuracy_score(y_test, y_pred))
print('precision: %.2f' % precision_score(y_test, y_pred))
print('recall: %.2f' % recall_score(y_test, y_pred))
print('F1: %.2f' % f1_score(y_test, y_pred))

# confusion matrix
confu = confusion_matrix(y_true = y_test, y_pred = y_pred)
plt.figure(figsize=(4, 3))
sns.heatmap(confu, annot=True, annot_kws={'size':15}, cmap='OrRd', fmt='.10g')
plt.title('Confusion Matrix')
plt.show()
```




[그림 2] 1차 모델 학습 결과 히트 맵

6. 2차 모델 학습을 진행한다. 2차에서는 부정응답의 개수만큼 학습을 진행한다. Random idx 긍정, 부정 값에 random_state 값을 12로 부여 후 train_test를 진행한다. Test_size는 0.25로, random_state는 1로 지정하여 학습한다. 1차와 마찬가지로 결과값과 히트맵을 출력한다.

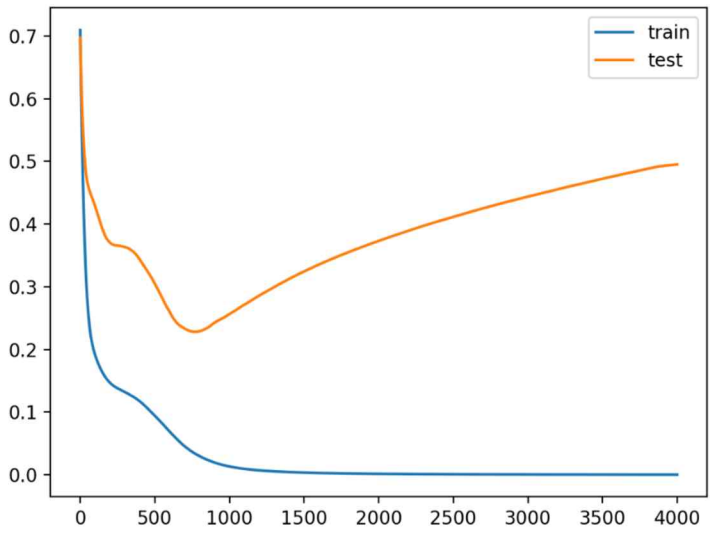
```
### 모델 학습2 ###
print(df['y'].value_counts()) # 부정 응답의 개수만큼 학습
r_value = df['y'].value_counts()[0]
positive_random_idx = df[df['y']==1].sample(r_value, random_state=12).index.tolist()
negative_random_idx = df[df['y']==0].sample(r_value, random_state=12).index.tolist()

random_idx = positive_random_idx + negative_random_idx
x = tf_idf_vect[random_idx]
y = df['y'][random_idx]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=1)

lr2 = LogisticRegression(random_state = 0)
lr2.fit(x_train, y_train)
y_pred = lr2.predict(x_test)

# classification result for test set
print('accuracy: %.2f' % accuracy_score(y_test, y_pred))
print('precision: %.2f' % precision_score(y_test, y_pred))
print('recall: %.2f' % recall_score(y_test, y_pred))
print('F1: %.2f' % f1_score(y_test, y_pred))

confu = confusion_matrix(y_true = y_test, y_pred = y_pred)
plt.figure(figsize=(4, 3))
sns.heatmap(confu, annot=True, annot_kws={'size':15}, cmap='OrRd', fmt='.10g')
plt.title('Confusion Matrix')
plt.show()
```



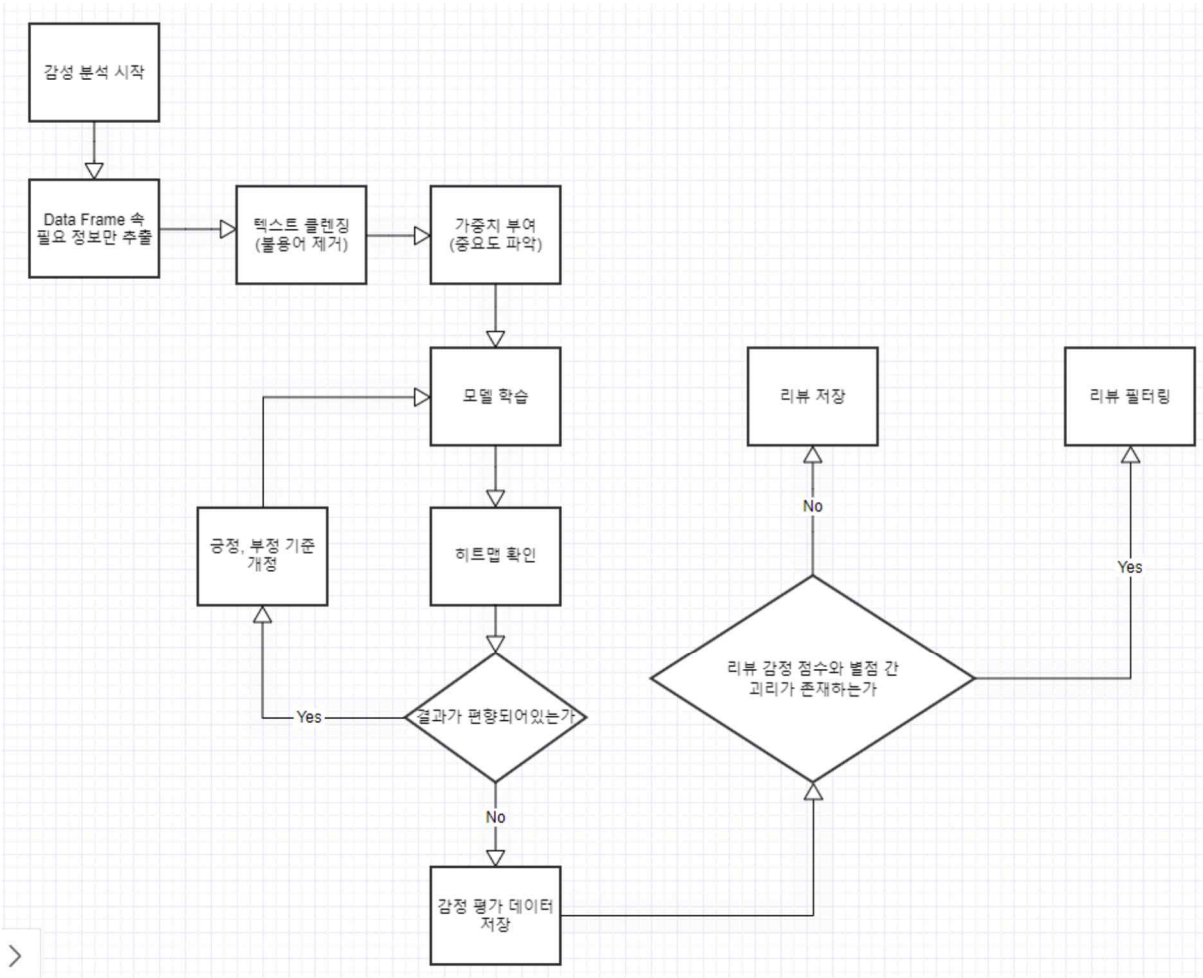

[그림 3] 2차 모델 학습 결과 히트 맵

7. 2차 학습을 마친 값을 파이썬의 내장함수인 enumerate()함수를 사용해 루프를 돌려 긍정 값과 부정값을 각각 변수에 저장한다. 저장한 긍정 값과 부정값을 상위 20개 출력한다. 긍정 키워드와 부정 키워드를 엑셀파일로 저장한다.

```
### 최종 결과 ###
print(sorted(((value, index) for index, value in enumerate(lr2.coef_[0])), reverse = True)[:5])
print(sorted(((value, index) for index, value in enumerate(lr2.coef_[0])), reverse = True)[-5:])

coef_pos_index = sorted(((value, index) for index, value in enumerate(lr2.coef_[0])), reverse = True)
coef_neg_index = sorted(((value, index) for index, value in enumerate(lr2.coef_[0])), reverse = False)

invert_index_vectorizer = {v: k for k, v in vect.vocabulary_.items()}

# 긍정 top20
print()
print("### 긍정 top20 ###")
for coef in coef_pos_index[:20]:
    print(invert_index_vectorizer[coef[1]], coef[0])

# 부정 top20
print()
print("### 부정 top20 ###")
for coef in coef_neg_index[:20]:
    print(invert_index_vectorizer[coef[1]], coef[0])

# 키워드 저장
pos = [[invert_index_vectorizer[coef[1]], coef[0]] for coef in coef_pos_index]
neg = [[invert_index_vectorizer[coef[1]], coef[0]] for coef in coef_neg_index]

df_pos = pd.DataFrame(pos, columns=['word', 'coef'])
df_neg = pd.DataFrame(neg, columns=['word', 'coef'])
df_pos.to_excel("result_pos_word.xlsx")
df_neg.to_excel("result_neg_word.xlsx")
```




[그림 4] iteration 크기에 따른 train과 test의 오차율




[그림 5] 감성 분석 Data Flow Chart

## 3.3. 출력 시스템

1. Pandas 패키지를 사용한다. 일정 크기 이상에서 자료가 생각되는 문제를 해결하기 위해 set_option을 사용했다. display.max_colwidth를 200으로 설정하여 리뷰 모두를 출력할 수 있도록 설정했다.

```
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
pd.set_option('display.width', 500)
pd.options.display.max_colwidth = 200
```

2. 첫번째로 실행할 main함수를 설정한다. Pandas 패키지를 활용하여 .xlsx 파일을 read한 뒤 Data Frame형태로 df 변수에 받아온다. 이때 행 순서가 자동으로 첫번째 열의 value로 들어간다. 이는 필요 없는 값이므로 df.set_index("name")을 통해 name 열로 첫번째 열을 대체한다. score_to_int함수를 사용하여 별점을 숫자로 변환한다. 개별적으로 작성 되어있던 별점을 하나로 통합하여 평점을 낸 뒤, 평점 순으로 정렬한다. 이후에 information함수를 제외한 나머지 함수에서는 리뷰 상세 내용이 필요 없으므로, 'name', 'address', 'category', 'score_avg'값만 Data Frame df에 남기고 first_task함수를 호출한다.

```
def main():
    # 데이터 로드
    df = pd.read_excel("final_result.xlsx", index_col=0)
    df.set_index("name")

    # 별점 -> 숫자
    df['score'] = df['score'].apply(score_to_int)

    # 별점 평균, category 분류 단순화
    avg_list = []
    for i in range(len(df)):
        n = len(df[df['name'] == df['name'][i]])
        s = sum(df.loc[df['name'] == df['name'][i], "score"].values)
        avg_list.append(round(s/n, 2))
    df['score_avg'] = avg_list
    df.drop_duplicates(['name'], inplace=True) # 이름 중복 제거
    df = df.sort_values(by=['score_avg'], ascending=False) # 별 평균으로 나열
    df = df[['name', 'address', 'category', 'score_avg']]
    first_task(df)
```

3. 선택지에 따라 동작을 결정하는 first_task 함수를 만든다. select함수를 호출하여 return값을 choice 변수에 저장하여 case별 시스템 실행을 위한 함수를 호출한다.

```
def first_task(df):     #동작 task 결정
    choice = select()
    if choice == "지역":
        local_select(df)
    elif choice == "카테고리":
        category_select(df)
    else:
        total_select(df)
```

4. 선택지와 그 결과를 출력하는 select 함수를 만든다. choice_list를 설정하여 입력 값이 choice_list 내에 존재하지 않으면 재입력 요청을 한다. 입력 받은 값을 return한다.

```
def select():   # 응답 받기
    choice_list = ["지역", "카테고리", "전체"]
    while True:
        result = input("{}중에서 한 가지를 선택해 주세요: ".format(choice_list))
        if result not in choice_list:
            print("다시 선택해주세요.")
        else:
            break
    return result
```

5. first_task함수에서 choice 변수 값이 지역일 때, local_select함수가 호출된다. 부산 내 지역을 busan list 요소로 저장한 뒤, 사용자가 원하는 지역을 입력 받는다. 이후 temp2에 주소에서 사용자 입력 값이 존재하는 음식점만 저장한다. concat함수를 사용하여 temp와 temp2를 병합한다. 이때 axis값을 0으로 부여하여 세로방향으로 병합이 진행된다. 이름 중복을 제거한 뒤, 별점 순으로 정렬한다. df 내에 사용자 입력 지역의 음식점이 없다면 재 입력 요청을 한다. 음식점이 존재한다면 store_list_maker함수를 호출하여 store_list를 생성한 뒤, 음식점 정보들을 출력하고 information함수를 호출한다. 마지막으로 last_task함수를 호출한다.

```
def local_select(df):
    temp = pd.DataFrame()
    busan = ['중구', '서구', '동구', '영도구', '부산진구', '동래구', '남구', '북구',
             '해운대구', '사하구', '금정구', '강서구', '연제구', '수영구', '사상구', '기장군']

    while True:
        local = input("부산광역시 내 지역 중에서 원하시는 곳을 입력해 주세요(ex. 금정구): ".format(busan))
        if local not in busan:
            print("다시 선택해주세요.")
        else:
            break

    temp2 = df[df['address'].str.contains(local) == True]
    temp = pd.concat([temp, temp2], axis=0)
    temp.drop_duplicates(['name'], inplace=True)  # 이름 중복 제거
    temp.sort_values(by=['score_avg'], ascending=False, inplace=True)  # 정렬
    temp.reset_index(drop=True, inplace=True)
    if len(temp) == 0:
        print('죄송합니다. 해당 지역의 음식점은 없습니다. 다른지역을 선택해주세요.')
        local_select(df)
    else:
        k = len(temp.index)
        store_list = store_list_maker(temp, k)
        print(temp)
        information(df, store_list)
        last_task(df, 1)
```

6. first_task함수에서 choice 변수 값이 카테고리일 때, category_select함수가 호출된다. Category 단순화를 위해 각 카테고리별 키워드 단어가 들어가 있는 리스트들을 선언한다. Category는 일식, 한식, 중식, 일식으로 단순화한다. 이때 한식의 키워드 단어는 다른 카테고리 키워드를 모두 합친 것으로 입력되어 있는데, 이는 차 집합 연산을 통해 한식 category 음식점을 찾기 위한 설정이다. 사용자로부터 카테고리 입력을 받으면, 해당 카테고리 범주에 해당하는 키워드 리스트를 사용하여 카테고리 별 음식점을 출력한다. 이때 키워드가 df의 category 열에 포함되어 있으면 해당 음식점을 카테고리 범주에 넣는 방식으로 진행된다. 한식의 경우, 나머지 카테고리 음식점들 모두를 합집합 한 뒤, 차집합을 통해 구해진다. 이후 과정은 local_select함수와 동일하다.

```
def category_select(df):
    category_list = ["일식", "한식", "중식", "양식"]
    japan = ['일본', '돈까스', '초밥', '텐동', '라멘']
    china = ['중국', '마라탕', '훠궈', '탕수육', '깐풍기']
    western = ['이탈리아', '프랑스', '피자', '양식', '브라질', '유럽']
    korea = japan + china + western
    while True:
        cate = input("{} 중에서 원하시는 카테고리를 입력해 주세요: ".format(category_list))
        if cate not in category_list:
            print("다시 선택해주세요.")
        else:
            break
    temp = pd.DataFrame()
    if cate == "일식":
        for c in japan:
            temp2 = df[df['category'].str.contains(c) == True]
            temp = pd.concat([temp, temp2], axis=0)
    elif cate == "중식":
        for c in china:
            temp2 = df[df['category'].str.contains(c) == True]
            temp = pd.concat([temp, temp2], axis=0)
    elif cate == "양식":
        for c in western:
            temp2 = df[df['category'].str.contains(c) == True]
            temp = pd.concat([temp, temp2], axis=0)
    else:
        temp3 = pd.DataFrame()
        for c in korea:
            temp2 = df[df['category'].str.contains(c) == True]
            temp3 = pd.concat([temp3, temp2], axis=0) #한식이 아닌 음식점 concat
        temp3.drop_duplicates(['name'], inplace=True)  # 이름 중복 제거
        temp = pd.concat([df, temp3, temp3]).drop_duplicates(keep=False)  #차집합

    temp.drop_duplicates(['name'], inplace=True)  # 이름 중복 제거
    temp.sort_values(by=['score_avg'], ascending=False, inplace=True)  # 정렬
    temp.reset_index(drop=True, inplace=True)
    k = len(temp.index)
    store_list = store_list_maker(temp, k)
    print(temp)
    information(df, store_list)
    last_task(df, 2)
```

7. first_task함수에서 choice 변수 값이 전체일 때, total_select함수가 호출된다. 전체 가게를 대상으로 출력을 진행하므로, 출력 음식점 개수에 대한 입력을 우선 받는다. 입력 받은 출력 개수만큼 출력한다. 이후 과정은 local_select함수와 동일하다.

```
def total_select(df):
    answer = ['10', '20', '50', '전체']
    while True:
        k = input("전체 가게 중 상위 몇개의 가게가 출력되길 원하십니까? (10, 20, 50, 전체 중 택 1): ")
        if k not in answer:
            print("다시 선택해주세요.")
        else:
            break
    if k == '전체':
        print(df)
        last_task()
    else:
        k = int(k)
        store_list = store_list_maker(df, k)
        print(df.iloc[:k, :])
        information(df, store_list)
        last_task(df, 3)
```

8. store_list_maker함수는 출력된 음식점들의 가게 이름 list를 return해주는 함수이다. df.iat[i, 0]를 통해 df의 [i, 0] 위치 값을 읽어와서 list에 append해준다.

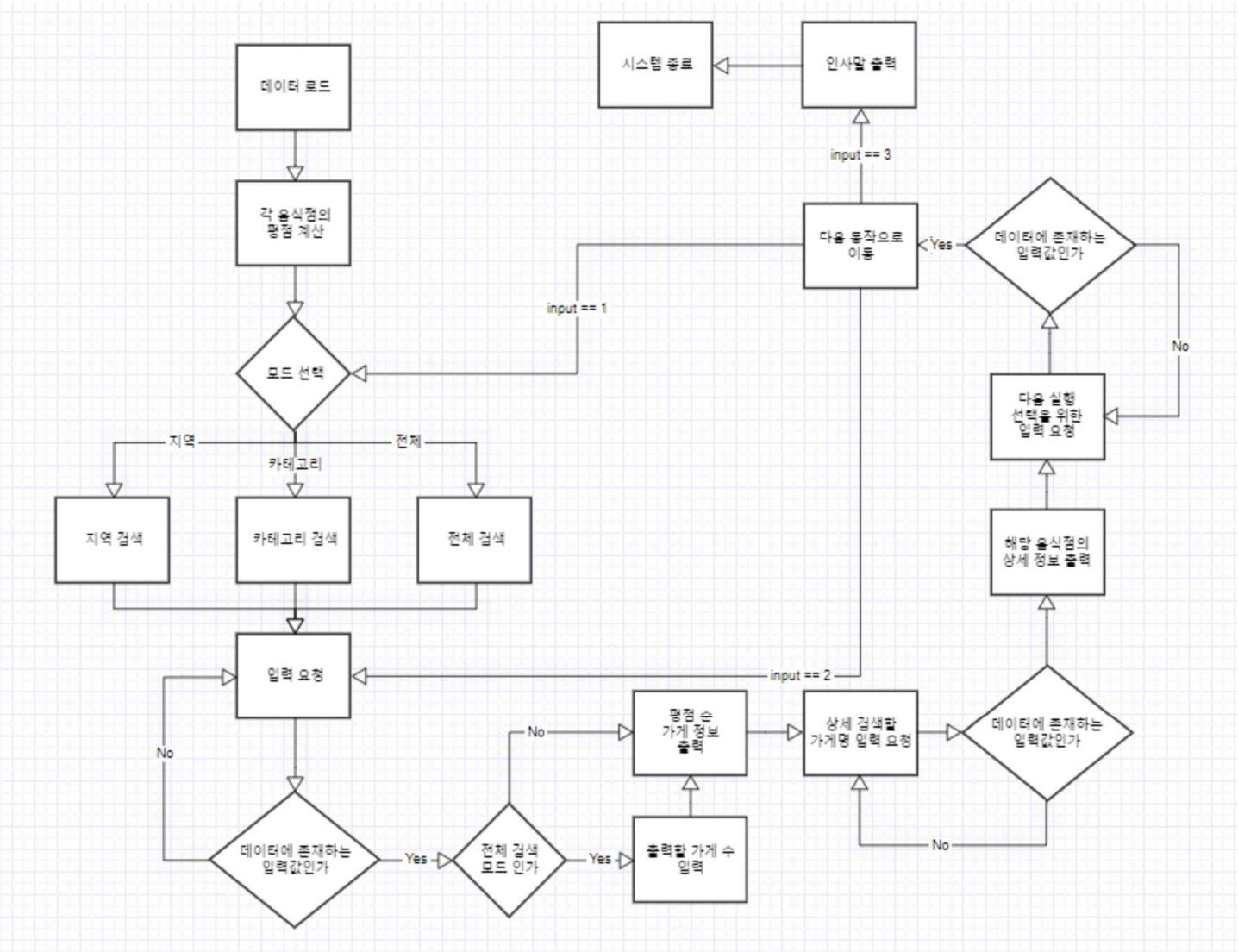
```
def store_list_maker(df, k):
    store_list = []
    for i in range(k):
        store_list.append(df.iat[i, 0])
    return store_list
```

9. information 함수는 가게의 상세정보를 출력하는 함수이다. df는 가게당 1행의 정보를 가지며 리뷰와 별점이 생략되어 있고, 전체 평점이 포함 되어있다. 상세정보 출력을 위해 .xlsx 파일을 pandas 패키지를 사용하여 다시 불러와 이를 df2변수에 Data Frame형태로 저장한다. 상세 열람을 희망하는 가게 입력을 받으며, parameter 값으로 받은 store_list에 존재하지 않는 가게 입력 시 재입력을 요청한다. df, df2의 name 열에서 가게 이름이 포함 되어있는 행을 따로 뽑아온다. 이를 활용하여 가게의 이름, 상세 주소, 가게 카테고리, 평점을 출력한 뒤, 각 리뷰에 대한 별점과 리뷰 내용을 출력한다. 이때 리뷰 part는 df2를 활용하여 출력하는 것이고, 나머지 part는 df를 활용하여 출력하는 것이다.

```python
def information(df, store_list):
    df2 = pd.read_excel("final_result.xlsx", index_col=0)
    df2.set_index("name")
    temp = pd.DataFrame()
    temp_df = pd.DataFrame()
    while True:
        store_name = input("상세정보 열람을 희망하는 가게의 이름을 입력해 주세요: ")
        if store_name not in store_list:
            print("잘못된 정보입니다. 다시 입력해주세요")
        else:
            break
    temp_df2 = df[df['name'].str.contains(store_name) == True]
    temp_df = pd.concat([temp_df, temp_df2], axis=0)
    temp_df.drop_duplicates(['name'], inplace=True)  # 이름 중복 제거

    temp2 = df2[df2['name'].str.contains(store_name) == True]
    temp = pd.concat([temp, temp2], axis=0)

    print("음식점 이름 : {}".format(store_name))
    print("상세 주소 : {}".format(','.join(list(temp_df['address']))))
    print("가게 카테고리 : {}".format(','.join(list(temp_df['category']))))
    print("가게 평점 : {} 점".format(','.join(map(str, list(temp_df['score_avg'])))))
    print("-------------------------------------------------")
    print(temp[['score', 'date', 'content']])
```

10. last_task 함수는 다음 동작을 결정한다. 1번 선택 시 홈, 2번 선택 시 현재 모드, 3번 선택 시 종료를 한다. 2번을 선택할 경우 parameter값인 mode_num의 값에 따라 해당하는 함수를 호출한다.

```python
def last_task(df, mode_num):    #다음 동작 결정
    while True:
        mode_select = int(input("\n1. 홈\n2. 현재 모드\n3. 종료\n다음 작업의 번호를 선택해 주세요: "))
        if mode_select not in [1, 2, 3]:
            print("잘못된 선택입니다. 다시 선택해주세요.")
        else:
            break
    if mode_select == 1:
        first_task(df)
    elif mode_select == 2:
        if mode_num == 1:
            local_select(df)
        elif mode_num == 2:
            category_select(df)
        else:
            total_select(df)
    else:
        print("감사합니다. 다음에 다시 이용해 주세요.")
```




[그림 6] 출력 시스템 Data Flow Chart

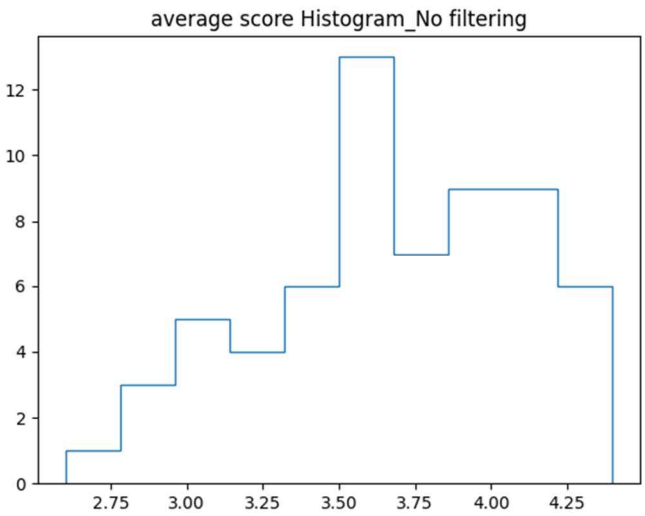
# 4. 연구 결과 분석 및 평가

## 4.1. 크롤링

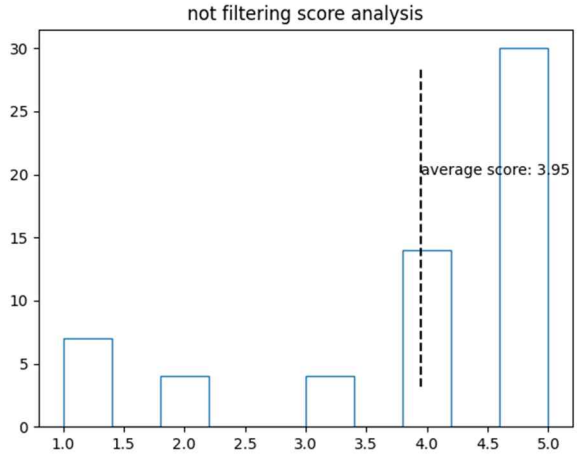
| | | | |
|---|---|---|---|
| 123 | 부산족발 족발/보쌈:부산광역시 별표 5개 10달 전 | 329 족발은 부산이지~ 차가운 족발이 쫀득하니 맛남 | |
| 124 | 부산족발 족발/보쌈:부산광역시 별표 4개 10달 전 | 291 밑반찬 구성 좋고 얼음컵 센스 굿냉채족발은 맛은 있으나 조금 겨자맛이 과한것 같음 | |
| 125 | 부산족발 족발/보쌈:부산광역시 별표 2개 11달 전 | 48 입맛이 엄서 부산 족발 한번씩 들리는데 오늘은영 족발이 질기고 살 이 말라 맛이 별로 다옛전에 그래도 손님도 만구 알아 줫는데코로나 인해 손님이 엄서 그런지 신경쓸 | |
| 126 | 부산족발 족발/보쌈:부산광역시 별표 4개 11달 전 | 58 족발을 좋아하지는 않지만 동행자들이 원해서 왔어요. 강강 괜찮아요 | |
| 127 | 부산족발 족발/보쌈:부산광역시 별표 5개 11달 전 | 34 양이 많지는 않지만 확실히 부드럽고 깔끔함. 냉채수육이 제일 유명해서 그것을 주문했는데 사이드로 일반족발이 몇개 나와서 그것도 맛볼수있음.냉채양념맛이 와사비맛이 강하 | |
| 128 | 부산족발 족발/보쌈:부산광역시 별표 2개 11달 전 | 142 그냥 쏘쏘..관광객 여러분들!!!!!!!!!!!!배달앱 켜고 아무집이나시켜도 다들 이정도는 합니다여기 너무 비싸요 ㅠㅠ | |
| 129 | 부산족발 족발/보쌈:부산광역시 별표 1개 1년 전 | 5 첨에는 모르고갔지만 두번은 안갈듯 | |
| 130 | 부산족발 족발/보쌈:부산광역시 별표 5개 1년 전 | 21 정말 궁금했음 어떤맛일지 오길 잘했음 왜 진작 안왔지!!!! 홀도 크고 방역수칙 좋았음 테이블마다 칸막이 냉채족발 소자치곤 양 많음 서비스로 사이다랑 발가락 2개 주시고 반찬 | |
| 131 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 3일 전 | 74 신선한 해물과 깔끔한 국물 깔끔한국물 맛 !애매한 해물탕집 아님 맛집 인정 | |
| 132 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 6일 전 | 85 낙새곰 점심 잘먹었어요. | |
| 133 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 1주 전 | 245 세상 맑은 해물탕 예약필수. | |
| 134 | 연산낙지ㅎ낙지전문점부산광역시 별표 3개 1달 전 | 17 가격대비 가성비가 낮아요 | |
| 135 | 연산낙지ㅎ낙지전문점부산광역시 별표 1개 1달 전 | 45 맛이 없어요 너무너무 국물도 밍밍하구요 돈이 아깝네요 정말 | |
| 136 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 1달 전 | 20 재료 신선~^^ 이 집은 볶음밥 스페셜 맛집이더군요~~~^^ | |
| 137 | 연산낙지ㅎ낙지전문점부산광역시 별표 2개 2달 전 | 92 오랜만에 갔는데 가격이 많이 올났고 내용물은 부실했어요 재방문은 안할것 같네요 | |
| 138 | 연산낙지ㅎ낙지전문점부산광역시 별표 4개 2달 전 | 137 예약하고 갔다왔는데 일요일은 안하는 것 같아요맛은 해물탕 고유의 맛이며 억수로 맛있다는 아님중짜해물탕~~~, 볶음밥 3 | |
| 139 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 2달 전 | 13 맛있게 잘~먹고왔어요신선한 해물 재료가 맛을 첨가시켜서 국물도 해장국처럼 시원 하여서 좋았고 볶음밥이 맛있어요 | |
| 140 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 4달 전 | 85 예약이 힘들기는 하지만 가성비와 맛이 다좋아요. 해물탕은 중자 이상을 추천합니다. 잘먹는 성인 3명이면 대자가 좋아요. 볶은밥은 따로 볶아서 나오니 미리 시켜도 좋아요. 주차 | |
| 141 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 4달 전 | 60 여기는 맛도 좋고 싱싱한데 너무 사람이 많습니다사장님만 손질이 가능합니다예약안하고 가면 못먹습니다볶음밥은 한번만시킬수있으니 참고하세요! | |
| 142 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 4달 전 | 199 맛있어요볶음밥은 진짜 최공한번밖에 못시키니깐 넉넉하게 시기세요 | |
| 143 | 연산낙지ㅎ낙지전문점부산광역시 별표 5개 4달 전 | 117 항상 신선한 재료만 사용하고 당일 재료 소진 시 영업이 종료된다. 맛이 깔끔하고 개운하다.예약필수 | |

[그림 7] 크롤링 결과 .xlsx 파일의 일부

크롤링 결과 가게들의 평점 분포 및 평균, 분산, 표준편차를 히스토그램을 통해 확인하였다. 그 결과 평점이 고르게 분포되어 분산 값이 높은 것으로 확인되었다. 이는 아래 그래프를 통해 알 수 있다. 실제 리뷰 작성자들 중 상당수가 우호적인 글을 남기기 위해 리뷰를 남긴다. 그러나 일부 허위리뷰나 악성 리뷰의 경우 좋지 않은 별점을 남기고, 광고 및 대가성 리뷰들은 높은 별점을 남기게 된다. 따라서 높은 별점의 분포 또한 높게 나타난다. 이러한 허위 및 악성 리뷰들의 필터링이 필요하다고 생각된다.




크롤링 결과 평점 분포 히스토그램




```
평균: 3.95
분산: 1.91
표준편차: 1.38

Process finished with exit code 0
```

[예시] 국제밀면 필터링 이전 별점 통계

## 4.2. 감정 분석 및 리뷰 필터링

```
좋아하는 국밥집입니다. 김치가 맛있어서 가는 곳인데 요즘들어 김치가 점점 맛이 떨어지고 있습니다..아쉽..
[('좋아하는', 'Adjective'), ('국밥집', 'Noun'), ('입니다', 'Adjective'), ('.', 'Punctuation'), ('김치', 'Noun'), ('가', 'Josa')]
[('맛있어서', 'Adjective'), ('가는', 'Verb'), ('곳', 'Noun'), ('인데', 'Josa'), ('요즘', 'Noun'), ('들어', 'Verb')]
[('김치', 'Noun'), ('가', 'Josa'), ('점점', 'Noun'), ('맛', 'Noun'), ('이', 'Josa'), ('떨어지고', 'Verb')]
[('있습니다', 'Adjective'), ('..', 'Punctuation'), ('아쉽', 'Adjective'), ('..', 'Punctuation')]
명사+형용사 비율: 0.55
동사+부사 비율: 0.14
```

[예시] 형태소 분석 및 비율 계산 결과

로지스틱 회귀분석을 통한 키워드 감정 분석 결과 중 일부이다. 2차례에 걸쳐 학습을 진행하여 긍정, 부정 양 측에 고른 분포를 띄고 있다. coef값의 절대값도

긍정, 부정 동일 순위 키워드에서 비슷한 점수를 띄고 있다.

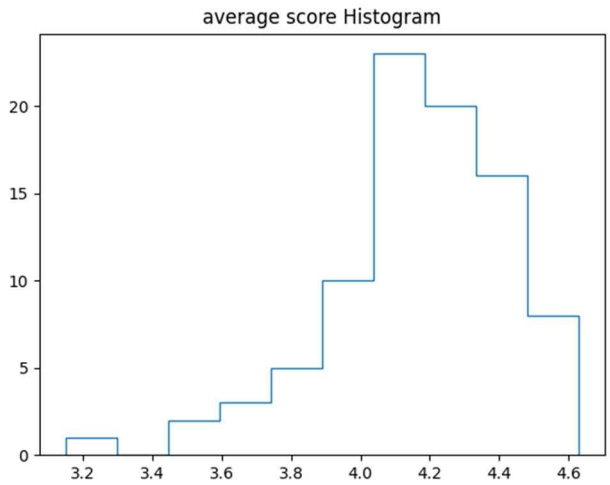
| | A | B | C |
|---|---|---|---|
| 1 | | word | coef |
| 2 | 0 | 최고 | 2.216937 |
| 3 | 1 | 웨이팅 | 1.202797 |
| 4 | 2 | 야주 | 1.189982 |
| 5 | 3 | 추천 | 1.135988 |
| 6 | 4 | 음식 | 1.080469 |
| 7 | 5 | 완전 | 1.068155 |
| 8 | 6 | 분위기 | 1.062387 |
| 9 | 7 | 삼겹살 | 1.020161 |
| 10 | 8 | 가족 | 0.989597 |
| 11 | 9 | 오픈 | 0.987419 |
| 12 | 10 | 만두 | 0.930751 |
| 13 | 11 | 부산 | 0.915412 |
| 14 | 12 | 회전 | 0.904969 |
| 15 | 13 | 강추 | 0.890663 |
| 16 | 14 | 맛집 | 0.866736 |
| 17 | 15 | 광안 | 0.837766 |
| 18 | 16 | 제일 | 0.836823 |
| 19 | 17 | 새우 | 0.831481 |
| 20 | 18 | 노포 | 0.7912 |
| 21 | 19 | 여기 | 0.788593 |
| 22 | 20 | 곱창 | 0.786483 |
| 23 | 21 | 무료 | 0.780168 |
| 24 | 22 | 낙곱새 | 0.778958 |
| 25 | 23 | 정말 | 0.766302 |
| 26 | 24 | 소스 | 0.764986 |
| 27 | 25 | 위치 | 0.758599 |

[그림 8] 긍정 키워드 값

| | A | B | C |
|---|---|---|---|
| 1 | | word | coef |
| 2 | 0 | 별로 | -2.05416 |
| 3 | 1 | 그냥 | -2.04471 |
| 4 | 2 | 예전 | -2.028 |
| 5 | 3 | 대비 | -1.68234 |
| 6 | 4 | 실망 | -1.67301 |
| 7 | 5 | 정도 | -1.46828 |
| 8 | 6 | 입맛 | -1.42241 |
| 9 | 7 | 살짝 | -1.24048 |
| 10 | 8 | 수준 | -1.1344 |
| 11 | 9 | 다른 | -1.13273 |
| 12 | 10 | 생각 | -1.12068 |
| 13 | 11 | 최악 | -1.09952 |
| 14 | 12 | 명성 | -1.08956 |
| 15 | 13 | 밍밍 | -1.07694 |
| 16 | 14 | 추억 | -1.07193 |
| 17 | 15 | 보통 | -1.05729 |
| 18 | 16 | 서서 | -1.01942 |
| 19 | 17 | 굳이 | -1.00263 |
| 20 | 18 | 손님 | -0.95767 |
| 21 | 19 | 느낌 | -0.95182 |
| 22 | 20 | 조미료 | -0.93901 |
| 23 | 21 | 만원 | -0.93276 |
| 24 | 22 | 불친절 | -0.9197 |
| 25 | 23 | 가격 | -0.91424 |
| 26 | 24 | 거의 | -0.91032 |
| 27 | 25 | 장사 | -0.88741 |

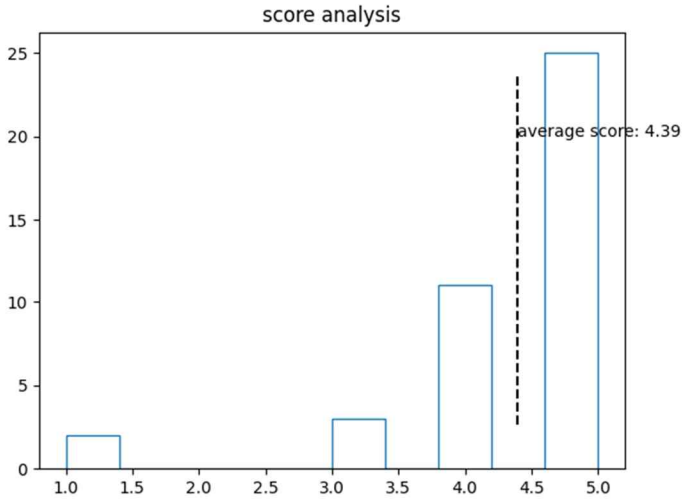
[그림 9] 부정 키워드 값

긍정 및 부정 키워드를 통해 리뷰 필터링을 진행하였다. 리뷰 필터링 이전의 평점 분포의 경우 평점이 고르게 분포되어 분산 값이 높았던 반면, 리뷰 필터링 이후 분산의 크기가 줄어들었다. 이를 필터링 후 평점 분포 히스토그램으로 확인하였다.




리뷰 필터링 후 평점 분포 히스토그램

분산의 크기가 줄어든 것은 양 극단의 별점이 필터링 되어 신뢰도 높은 평점이 되었음을 의미한다. 추가적으로 평점이 증가한 추세를 띄고 있다. 이는 리뷰를 가져오는 시기에 따라 추세성이 달라질 것이다. 해당 시스템 특성에 영향을 받는 것은 아니라 판단된다.

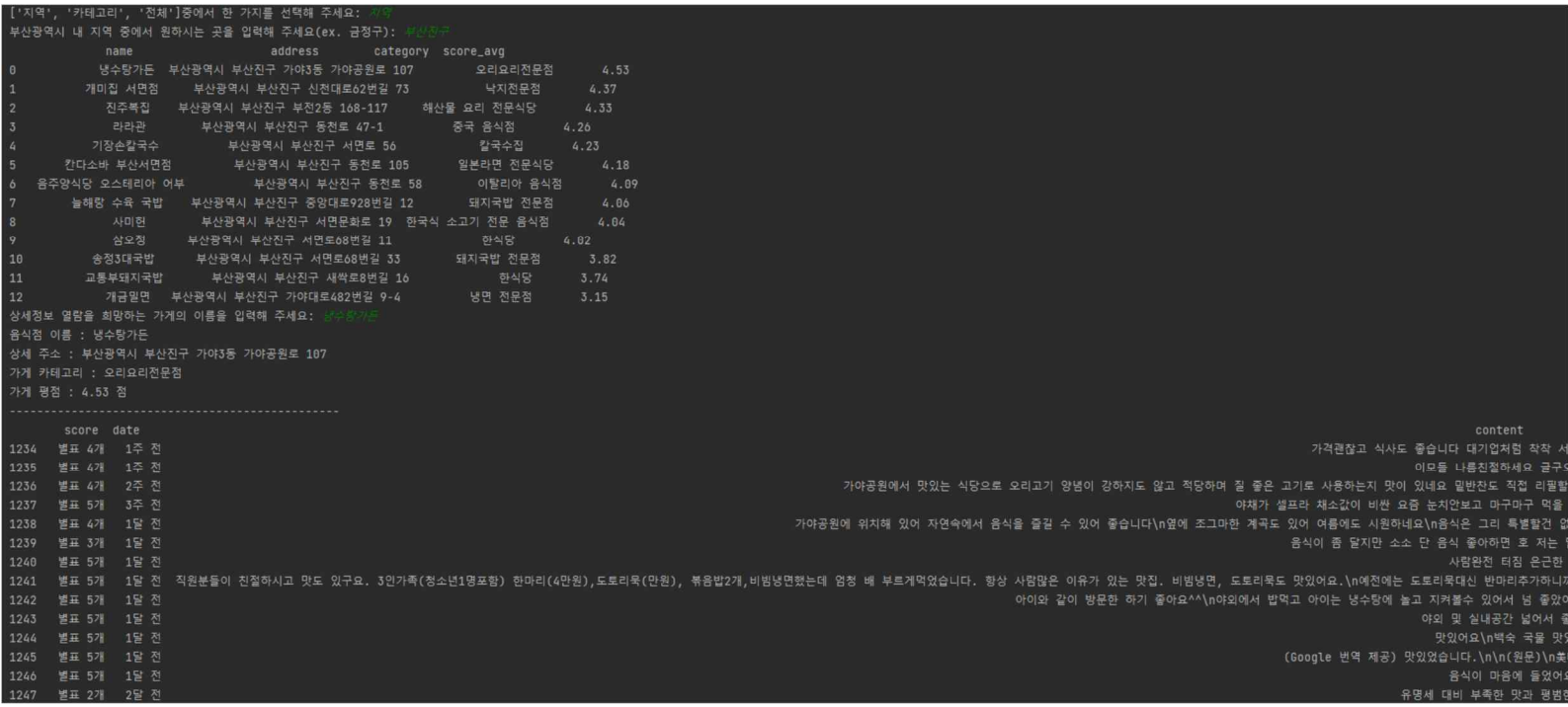



```
평균: 4.39
분산: 0.97
표준편차: 0.98

Process finished with exit code 0
```

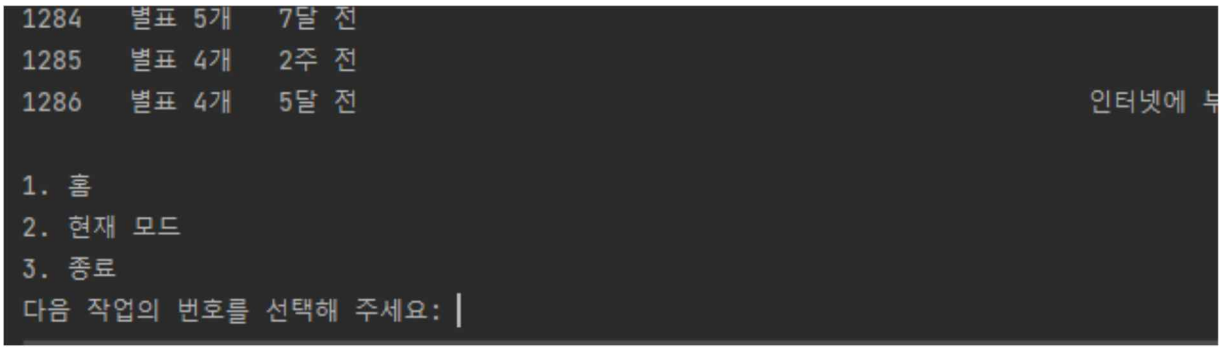
[예시]국제밀면 필터링 이후 별점 통계

필터링을 통해 리뷰의 신뢰도가 올라가게 되었고, 가게의 평균 별점을 통해 개인의 맛집 여부를 판단할수 있게 되었다.

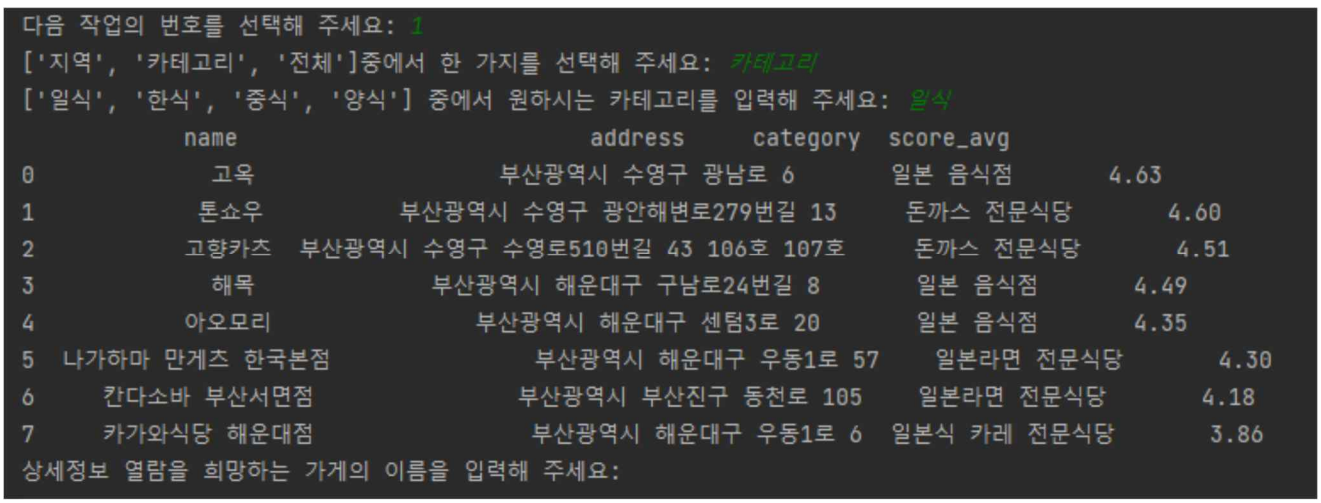
## 4.3. 출력 시스템




[그림 10] 지역-부산진구-순위 및 '냉수탕가든' 상세 정보




[그림 11] 작업이 끝난 후 다음 작업 선택 화면




[그림 12] 카테고리-일식 출력 화면

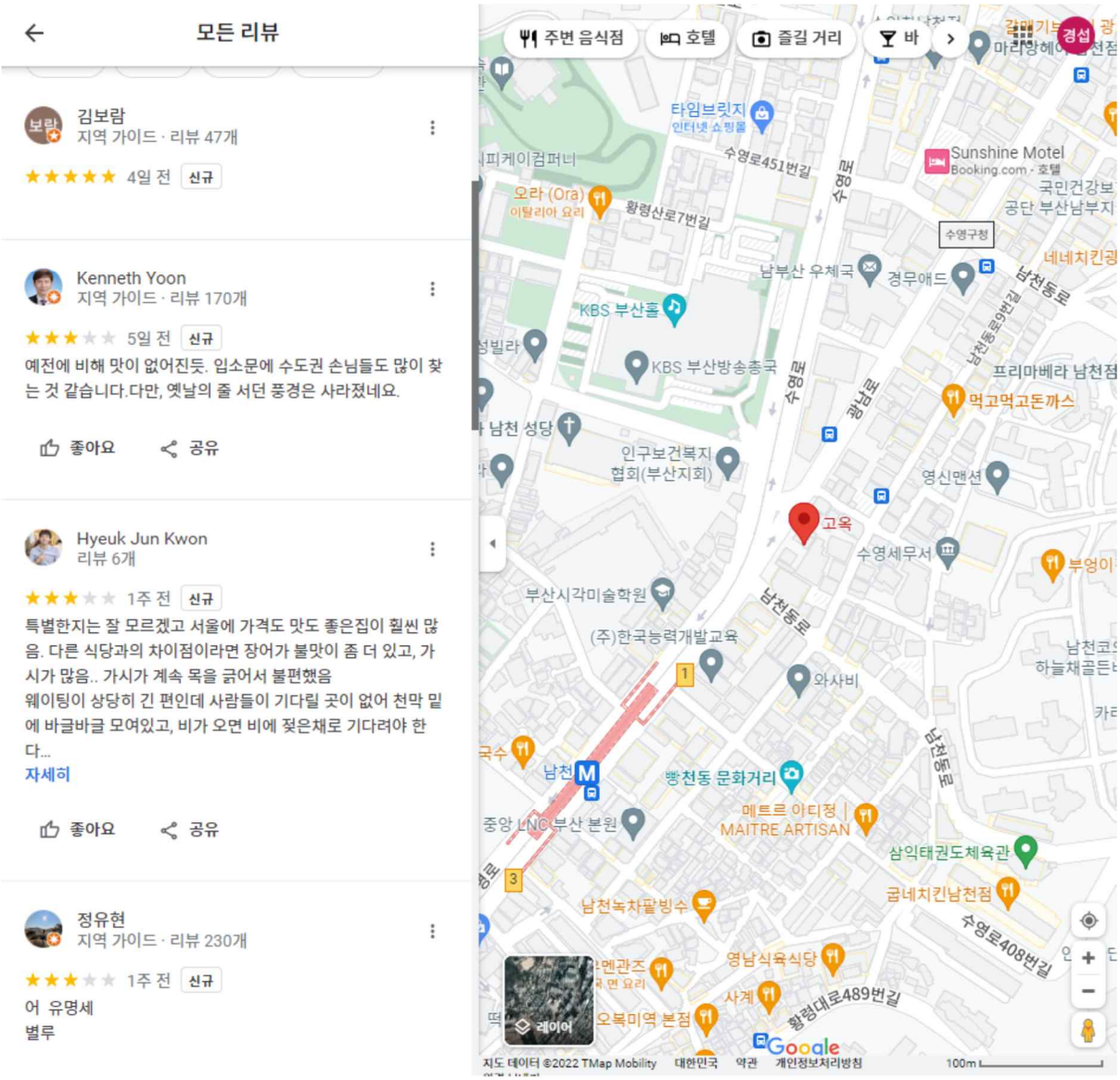
```
['지역', '카테고리', '전체']중에서 한 가지를 선택해 주세요: 전체
전체 가게 중 상위 몇개의 가게가 출력되길 원하십니까? (10, 20, 50, 전체 중 택 1): 20
                name                                address      category  score_avg
1422              고옥                  부산광역시 수영구 광남로 6        일본 음식점       4.63
3972         영진돼지국밥                부산광역시 사하구 신평동 628-6      돼지국밥 전문점       4.60
2917            톤쇼우            부산광역시 수영구 광안해변로279번길 13      돈까스 전문식당       4.60
3881            사까에             부산광역시 해운대구 해운대해변로 296           NaN       4.54
3558         이재모 피자                부산광역시 중구 광복중앙로 31        피자 전문점       4.53
1234          냉수탕가든          부산광역시 부산진구 가야3동 가야공원로 107       오리요리전문점       4.53
575             고향카츠   부산광역시 수영구 수영로510번길 43 106호 107호      돈까스 전문식당       4.51
2876              해목                부산광역시 해운대구 구남로24번길 8        일본 음식점       4.49
3603         평사리순두부          부산광역시 연제구 연산동 중앙천로19번길 40      두부 요리 전문점       4.48
262       금수복국 해운대본점             부산광역시 해운대구 중동1로43번길 23     복어 요리 전문식당       4.47
3834            금강만두       4번지 KR 부산광역시 동래구 92 금강만두 1층           음식점       4.45
1138            청산곱창                  부산광역시 해운대구 반여1동       곱창구이 전문점       4.42
1328  엘올리브 el olive              부산광역시 수영구 좌수영로 129-1       이탈리아 음식점       4.40
1596            풍년곱창                부산광역시 남구 감만1동 77-12         곱창전문점       4.39
44          국제밀면 본점    부산광역시 연제구 거제1동 중앙대로1235번길 23-6          국수 전문점       4.39
2148          참전복구이                  부산광역시 수영구 남천1동           음식점       4.38
1817          청송양곱창                부산광역시 수영구 남천2동 6-10           한식당       4.38
359        양산박 소금구이                  부산광역시 남구 못골로 75  한식 고기구이 레스토랑       4.38
1092        개미집 서면점             부산광역시 부산진구 신천대로62번길 73          낙지전문점       4.37
131         연산낙지해물탕             부산광역시 연제구 연산9동 과정로 121          낙지전문점       4.37
상세정보 열람을 희망하는 가게의 이름을 입력해 주세요:
```

[그림 13] 전체-평점 기준 상위 20개 음식점 출력

```
1. 홈
2. 현재 모드
3. 종료
다음 작업의 번호를 선택해 주세요: 3
감사합니다. 다음에 다시 이용해 주세요.

Process finished with exit code 0
```

[그림 14] 3번 모드 선택 시 시스템이 종료되는 모습

출력되는 평점이 과거 리뷰에 의해 변질되는 현상을 막을 수 있게 되었다. 또한 허위 리뷰로 의심되는 데이터들의 별점을 통계에 포함시키지 않으면서 더욱 신뢰성 높은 자료를 얻을 수 있게 되었다. 아래 [그림 15]을 보면 리뷰가 없이 별점만 주어진 리뷰, 너무 짧은 리뷰 등을 필터링하면서 객관적인 지표를 가질 수 있게 되었다 것에 의의가 있다. 별점 리뉴얼의 결과는 [표 1]을 통해 기존 지표와 차이점이 발생했음을 알 수 있다.




[그림 15] 수영구 남천동에 위치한 고옥의 최신 리뷰 중 일부

| 음식점 이름 | 구글 맵 평점 | 프로젝트 평점 |
|---|---|---|
| 고옥 | 4.20 | 4.63 |
| 영진돼지국밥 | 4.50 | 4.60 |
| 톤쇼우 | 4.60 | 4.60 |
| 사까에 | 4.60 | 4.54 |
| 이재모 피자 | 4.40 | 4.53 |
| 냉수탕가든 | 4.20 | 4.53 |
| 고향카츠 | 4.30 | 4.51 |
| 해목 | 4.30 | 4.49 |
| 평사리순두부 | 4.30 | 4.48 |
| 금수복국 해운대본점 | 4.10 | 4.47 |

[표 2] 프로젝트 내 평점 상위 10개 음식점 평점 비교

**수집 데이터 필터링 예시**

**1) 크롤링 단계**

| Number | score | date | ers_review | content | 명사, 형용사 | 동사, 부사 |
|---|---|---|---|---|---|---|
| 1 | 5 | 1시간 전 | 1 | 제가 먹어본 국밥 중에 제일 맛있는 곳입니다. 강추! | 0.57 | 0.14 |
| 2 | 5 | 1일 전 | 5 | 맛있어요. | 0.5 | 0.0 |
| 3 | 4 | 1일 전 | 37 | 오랜만에 갔다왔습니다. 늘 번창하세요. 감사합니다. | 0.27 | 0.36 |
| 4 | 4 | 2년 전 | 98 | 적당히 매콤하고 감칠맛나고 맛있게 잘먹었습니다. | 0.44 | 0.22 |
| 5 | 5 | 7일 전 | 12 | 맛이 없는데 줄서는거 이해불가 피순대 진짜 아닌듯 | 0.8 | 0.1 |

1번 데이터: 작성자의 총 리뷰 개수(ers_review)가 3개 미만이므로 크롤링 단계에서 리뷰 배제

| Number | score | date | ers_review | content | 명사, 형용사 | 동사, 부사 |
|---|---|---|---|---|---|---|
| 2 | 5 | 1일 전 | 5 | 맛있어요. | 0.5 | 0.0 |
| 3 | 4 | 1일 전 | 37 | 오랜만에 갔다왔습니다. 늘 번창하세요. 감사합니다. | 0.27 | 0.36 |
| 4 | 4 | 2년 전 | 98 | 적당히 매콤하고 감칠맛나고 맛있게 잘먹었습니다. | 0.44 | 0.22 |
| 5 | 5 | 7일 전 | 12 | 맛이 없는데 줄서는거 이해불가 피순대 진짜 아닌듯 | 0.8 | 0.1 |

4번 데이터: 1년이 지난 데이터이므로 크롤링 단계에서 리뷰 배제

**2) 전처리 단계**

| Number | score | date | ers_review | content | 명사, 형용사 | 동사, 부사 |
|---|---|---|---|---|---|---|
| 2 | 5 | 1일 전 | 5 | 맛있어요. | 0.5 | 0.0 |
| 3 | 4 | 1일 전 | 37 | 오랜만에 갔다왔습니다. 늘 번창하세요. 감사합니다. | 0.27 | 0.36 |
| 5 | 5 | 7일 전 | 12 | 맛이 없는데 줄서는거 이해불가 피순대 진짜 아닌듯 | 0.8 | 0.1 |

2번 데이터: 리뷰 길이가 10미만이므로 데이터 세트에서 삭제

| Number | score | date | ers_review | content | 명사, 형용사 | 동사, 부사 |
|---|---|---|---|---|---|---|
| 3 | 4 | 1일 전 | 37 | 오랜만에 갔다왔습니다. 늘 번창하세요. 감사합니다. | 0.27 | 0.36 |
| 5 | 5 | 7일 전 | 12 | 맛이 없는데 줄서는거 이해불가 피순대 진짜 아닌듯 | 0.8 | 0.1 |

3번 데이터: 형태소 분석을 활용하여 명사, 형용사 비율과 동사, 부사 비율을 계산했다. 그 결과 명사, 형용사 비율이 동사, 부사 비율보다 낮으므로 삭제

| Number | score | date | ers_review | content | 명사, 형용사 | 동사, 부사 |
|---|---|---|---|---|---|---|
| 5 | 5 | 7일 전 | 12 | 맛이 없는데 줄서는거 이해불가 피순대 진짜 아닌듯 | 0.8 | 0.1 |

5번 데이터: 별점과 review의 긍정 및 부정 일관성이 지켜지지 않으므로 삭제

review의 감정 value 변환값은 -0.315798로 부정평가를 하고 있으므로, 별점이 3개 이하로 형성되어야 신뢰성 있다 판단할 수 있다.

# 5. 결론 및 향후 연구 방향

## 5.1. 결론

본 졸업 과제에서는 부산 지역 음식점 데이터를 구축하여 머신 러닝을 통해 리뷰 필터링을 진행했으며, 사용한 알고리즘으로는 TF-IDF 가중치 부여 방법을 사용했다. 또한 크롤링한 리뷰 데이터를 토대로 감정 분석 연구를 진행했다. 그 결과 현재 구글 맵의 평점과 다르게 신뢰성 높은 평점 및 리뷰 데이터를 보유하는데 성공했다. 또한 리뷰 감정 분석에서 상당수의 키워드가 긍정적인 value 값을 가지는 것을 확인했고, 이를 재조정하는 방안을 제시했다.

### 5.2. 향후 연구 방향

(1) 현재 학습을 통해 긍정 부정 단어를 분류하고 그 분류한 단어를 토대로 진짜리뷰 가짜 리뷰를 나누었지만 100% 정확하지는 않다. 긍정 단어와 부정 단어를 정확하게 분류할 수 있는 트레이닝 데이터 셋을 구축하는 방향으로 연구를 진행한다.

(2) 부산으로 한정된 서비스가 아닌 전국단위 서비스로 확대하여 전국의 맛집을 데이터 표본으로 설정한다.

(3) 과적합 문제를 해결할 수 있는 방안을 강구하여 구글 맵 뿐만 아닌 네이버, 카카오 등 다른 곳의 리뷰를 추가로 수집한다. 정확도와 신뢰도가 더 높은 시스템 구축을 위한 연구를 진행한다.

(4) 실시간 서버를 구축하여 정확도 높은 시스템 구축을 강구해본다.

## 6. 참고 문헌

[1] Nitin Jindal and Bing Liu, "Opinion spam and analysis,", pp. 219-230, WSDM
[2] yoonkt200. Korean_stopwords.txt [Online]. Available: "https://raw.githubusercontent.com/yoonkt200/FastCampusDataset/master/korean_stopwords.txt"