

융복합 프로젝트 : 반려견 산책 어플 개발_빅데이터 보고서

강민구, 김주희, 손지호

1. 빅데이터 주요 업무 및 협력사안

- a. 견종별 산책 가이드 라인, 전체 견종 리스트 데이터
- b. 지도 위에 표시되는 데이터
- c. 실시간 날씨 오픈 API 데이터

2. 빅데이터 주요 업무 세부사항

- a. 자연어 분석
 - i. 네이버 카페 크롤링
 - ii. KoNLPy 분석
 - iii. 시각화 및 인사이트 도출
- b. 지도 표시 데이터 수집
 - i. 공공데이터 포털
 - ii. 여기다 댕댕이 크롤링
 - iii. 실시간 날씨 정보 수집 Open API
- c. 카페 추천 시스템
 - i. 자연어 분석
 - ii. 네이버 지도 리뷰 크롤링
 - iii. 리뷰자수 가중치 계산
 - iv. 카페 추천 시스템, 공원 근접 카페 추가 가중치 계산
 - v. 추천시스템 알고리즘
 - vi. 추천 시스템 모듈화

3. 개선방안

4. 데이터 목록

5. 참고 문헌

1. 빅데이터 주요 업무 및 협력사안

a. 견종별 산책 가이드 라인, 전체 견종 리스트 데이터

견종별 산책 가이드 라인 데이터를 전체 견종 리스트 데이터와 산책 가이드 라인 데이터로 정규화하여 AWS RDS에 저장한 후, 사용자가 견종을 등록하면 클라우드반에서 구현한 AWS lambda를 통해 해당 견종의 일일 권장 산책 가이드라인 정보(산책 횟수, 시간 및 거리)가 불러와져서 “반려견 관리”창에 나타난다.

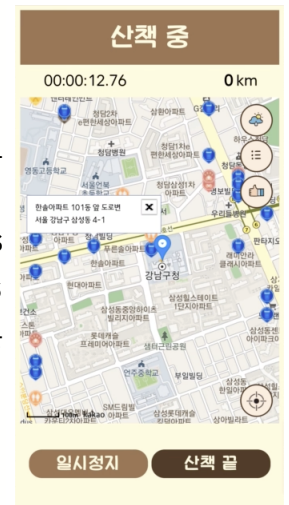
이 후에 사용자가 반려견을 데리고 산책을 다녀오면 산책 데이터는 AWS RDS에 이미지는 AWS S3에 저장된 후, AWS lambda를 통해 불러와져서 한눈에 일별 권장 가이드 라인과 비교 분석이 가능하도록 구현하였다.



b. 지도 위에 표시되는 데이터

공공데이터포털에서 쓰레기통과 공중화장실 현황 데이터를 csv파일로 받아오고, 여기다댕당이 사이트에서 애견 동반식당, 애견동반카페, 애견카페, 운동장, 공원, 동물병원 정보 데이터를 크롤링해와서 카카오 지도 API를 이용하여 주소를 통해 각 좌표값을 알아내었다.

모든 데이터를 지도 위에 표시되는 데이터로 하나로 합치는 데이터 정규화를 진행하여 AWS RDS에 저장한 후, 사용자가 지도에서 필요한 편의 시설을 선택할 시에 클라우드반에서 구현한 AWS lambda를 통해 해당 편의 시설에 대한 정보가 불러와져서 좌표가 지도 위에 마커로 표시되고, 마커를 클릭 시, 시설에 대한 정보(장소명과 주소)가 나타난다.



c. 실시간 날씨 오픈 API 데이터

IoT의 온습도 기기를 활용하여 산책 전후로 실시간 날씨 정보를 전달하는 것에 한계점이 발생하여, “Open Weather”와 “한국환경공단”에서 제공하는 Open API를 통해 실시간 날씨 정보를 제공하기로 하였다.

이에 실시간 크롤링 코드를 설계하여 한 시간 단위로 AWS lambda가 실행되게 하여 이를 통해 가져온 실시간 날씨 데이터를 AWS RDS에 저장하여 웹앱에서 Ajax를 통해 AWS RDS 데이터를 가져와 사용자가 산책하기 전과 산책하는 중에 확인할 수 있게 표시하였다.



a. 자연어 분석

1. 네이버 카페 크롤링

[illegible]

<p>반려견 산책용 할당시간 알림</p>	<p>오늘도인지 무리짐 강아지와 함께 산책할줄하고있었어요 산책을 다들 반려견 보호자를 만나어요 저희 강아지는 순종종목에 큰오류를 아직까지 없는데 아직 사회화가 미숙해서 다른 강아지들보면 쟁쟁하더라 다가가고싶어해도 늘 조심하고 목줄 착용과 긴장해요 혹시, 무리짐 강아지가 어떤 놀발행동들출지 모르나면요 근데, 다들 산책중 만난 반려견주 분이 물어오시더라구요 예는 몰아요? 라는 저희강아지를 가리키면서 말미요 이해는합니다 저희 강아지가 쟁쟁하더라 다가가려하니 각각되서서 그러할수있다는걸요~ 제가 그정도 모르죠~ 어떤행동을출지 단정할수없어요 그래서 재가독을 짧게 작성고 있어요~ 그리고왔더니 불만하십니까 지나가시더군요* 저는 지나가시디라고 하고 재잘질 가려는데 뒤돌아서서 저보고 무리짐 강아지는 밭 들거든요! 라고하시며 가시는데 제 기분요..영 속상하네요ㅠ 제가 예민한가요? 견주분들은 예는 몰아요?라는 질문받심.. 뭐라고 답할필할것같은디~ㅠ 괜히 산책 수 합법한마음에 하소연 해드립니다* *</p>	<p>농성입니다 별 이상한사람 다들~같이요. 아베라구요? 하시죠 ㅋㅋ 2023.01.31, 11:35</p>
<p>요즘 미세먼지 계속 심하네 요~ㅜㅜ 반려견 산책 어떻게 하세요?</p>	<p>지금 미세먼지가 심해서 4월째 반려견 산책을 못시키고 있네요 ㅜㅜ 주위분들 물어보니까 ... 1. 주차장에서 시키는 분 2. 집에서 직접하러 보내주는 분 3. 그냥 간식주면서 삼피우요 ;; 이렇게 하시던데 좋은 방안 없을까요?;ㅜㅜ</p>	<p>취우두유두 못하고있어요 ㅜㅜ 시무룩해있어요 2023.01.08, 13:29</p>
<p>탄네스 하네스 반려견 산책 줄</p>	<p>탄네스 하네스 라지사이즈 미사용 제품 개봉해서 박스는 없어요~ 우리 아이에서 사이즈가 작아요 ㅜㅜ 팩트 1. 제품명/색/사이즈 2. 상품 상태(개봉/봉/고); 3. 구매 일자; 4. 하자 사유(용/무); 5. 상품 설명 반려견 강아지 영양이 하네스 목줄 산책줄</p>	<p>[Q]</p>
<p>제 취미는 반려견 산책이 예요</p>	<p>오 며칠 날이 너무 추워서 산책을 잘 못하고 있네요 집이 서늘술 근처라 주로 서늘술이랑 성수동 골목들 한두시간 걸어다니는리 저의 가장 큰 행복이네요 성수동 다들 밤에 가면 조망이 너른 이베요 순회호 개만전도 근처에서 하고 있네요 간헐히 조망이나 삼시세끼들이 바뀌어서 세로유우 논은 서늘술 다워 이베요 달려 달려~~~~!!!</p>	<p>별 너무 멋진 리트레버네요... 산책이 취미이면 강아지는 얼마나 행복할까요... 🐾 2023.01.27, 14:51</p>
<p>오늘같이 미안 나쁜날 반려 견 산책하나요?</p>	<p>오늘 미세먼지,초미세먼지 매우나빠서라서 산책을 나갈까 망설미네요.ㅜㅜ 이렇게 미세먼지 매우나쁜날 반려견 산책하시나요?</p>	<p>포도알포도알 캐리지도 마루레드스~ 2023.01.07, 14:25</p>
<p>반려견 산책주의(영양학)</p>	<p>아침밥에 영양학들을 양형 하려하는데 반려견 산책나갈때 산책 착용시키세요. 오늘 아침 산책나가는데 우리애들이 발을 들고 다니니까 영양학습매매리하는 걸 알았어요. 발바닥 화상입는답니다.</p>	<p>보라계좌로당 있어요~ 특히 어떤 도로 횡단보도 건널 때 조심시키워요 ㅜ 2022.12.16, 22:24</p>

2. KoNLPy 분석

크롤링한 데이터를 한 단위별(recode=카페글제목+본문+댓글)로 묶어 한글을 제외한 모든 문자를 삭제하여 전처리 하였다. 그리고나서 KoNLPy Kkma를 이용해 형태소 단위로 토큰나이징을 진행해 명사와 동사만을 추출하였다.

```
1 import konlpy
2 from konlpy.tag import Kkma
3
4 kkma = Kkma()
5
6 textNVs = list()
7
8 for i in range(0, len(texts)):
9     if texts[i] == "" or texts[i] is None:
10         pass
11     else:
12         textNoun = kkma.nouns(texts[i])
13         textNVs.append(textNoun)
14         textVerb = [x for (x, y) in kkma.pos(texts[i]) if y == 'VV']
15         textNVs.append(textVerb)
16 print(textNVs)
```

[[['반려', '반려견', '견', '산책', '산책중', '중', '일'], ['오늘', '오늘오전', '오전', '우리집', '강아지', '동네', '동네산책', '산책', '책', '책중', '중', '반려', '반려견', '견', '보통자', '저희', '산책중', '진', '사회화', '미숙', '해', '조심', '목줄', '간장', '돌발', '돌발행동', '행동', '만난', '반려견주', '견주', '분', '물으시더라구요', '라', '말', '이해', '각정', '수', '걸', '저', '단정', '제가', '제가목줄', '있어요', '불안', '길', '데', '기분', '가요', '견주분', '애', '요', '라', '질문', '질문받으심', '받으심', '뭐', '답변', '후', '마음', '하소연'], ['하', '사', '만나', '짓', '까', '보', '깡깡거리', '다가가', '잡', '하', '모르', '문', '가리키', '하', '깡깡거리', '다가가', '그려', '있', '그리하', '오르', '하', '잡', '하', '지나가', '지나가', '하', '가', '가', '뒤돌', '스', '물', '하', '가시', '물', '해보'], ['오늘', '미세', '미세먼지', '먼지', '반려', '반려견', '견', '산책'], ['하'], ['미세', '미세먼지', '먼지', '일', '반려', '반려견', '견', '산책', '주위', '주차장', '분', '진', '간식', '방안'], ['시키', '불', '물어보', '시키', '놀', '주', '실땀', '이탈', '하'], ['터', '터네스', '네스', '하네', '오늘오전': 1, '동네': 51, '동네산책': 3, '책중': 4, '사회화': 636, '미숙': 1, '조심': 38, '돌발행동': 5, '만난': 3, '반려견주': 3, '물으시더라구요': 1, '단정': 1, '제가목줄': 1, '있어요': 1, '요라': 1, '질문받으심': 1, '받으심': 1, '하소연': 3, '깡깡거리': 4, '가리키': 2, '뒤돌': 2, '미세': 12, '미세먼지': 12, '방안': 1, '실땀': 1, '이탈': 1, '터네스': 2, '네스': 23, '하네스': 22, '산책중': 4, '라지': 1, '라지사이즈': 1, '미사용': 1, '제품개봉': 1, '개봉': 3, '박스': 5, '팩트': 1, '제품평': 2, '제품평브랜드': 2, '상태미개봉중고': 2, '유무': 4, '유무유무': 2, '실용반려견': 1, '명명이': 3, '서울술': 3, '근처': 44, '상수동': 2, '글목': 4, '같이다니는거지': 1, '다들함': 1, '올밤': 1, '전도': 1, '설치물': 1, '논논': 1, '초미세': 1, '초미세먼지': 1, '지내': 4, '망설이': 1, '산책주의열화갑상': 1, '주의': 36, '어젯밤': 1, '착용': 32, '영화감독배우': 1, '화상': 3, '어린이집': 2, '등원': 2, '공지': 1, '수호': 1, '커피': 16, '열화': 1, '요해': 2, '날씨': 85, '홍은네': 1, '세트': 5, '카메라': 695, '문제': 677, '반려견교육훈련': 642, '교육': 691, '훈련': 677, '발생': 664, '장소': 53, '카페': 1, '활성화': 4, '이벤트': 1, '취소': 3, '가을산책만남': 1, '매니지': 1, '번호': 2, '멤버': 5, '멤버분': 1, '네이버카페': 1, '인증사진': 1, '모집기간': 3, '당첨인원': 2, '당첨상품': 2, '이벤트신청글': 1, '영화': 1, '산책이야기': 1, '캠페인산책': 1, '필수미션': 3, '보호자분': 4, '놀이하기': 1, '개롱': 636, '맷캣': 5, '지키기': 3, '이중문': 1, '이중문설치': 1, '인식표달기': 1, '달기': 3, '견개': 1, '견개구좌': 1, '구좌': 1, '만들기': 5, '기부하기': 1, '박스': 5, '응원': 3, '응원댓글': 1, '신청기재': 1, '목요일': 5, '세트게': 1, '반려견산책이벤트': 1, '일산책': 4, '일산책캠페인': 1, '교육놀이하기': 1, '맷캣지키기': 1, '이중문설치하기': 1, '반려견교육']]]

명사와 동사만 추출한 데이터에서 검색한 키워드, 한글 불용어와 상관없는 단어 그리고 한음절 단어를 필터링 하여 남은 데이터에서 언급된 수 만큼 카운팅 하였다.

```
finalTexts의 단어들에서 다음의 단어들을 삭제하고 언급 수를 세어 textDict에 저장
```

- 불용어와 상관없는 단어
- '동'과 '품', '번', '평'을 제외한 1개짜리 단어
- 검색한 키워드

```
In [64]: 1 textDict={}
2
3 for word in finalTexts:
4     if word not in stop_words:
5         if len(word)>1 or word in ["동","품","번","평"]:
6             if word not in keywords:
7                 if word not in textDict:
8                     textDict[word]=1
9                 else:
10                    textDict[word]+=1
11 print(textDict)
```

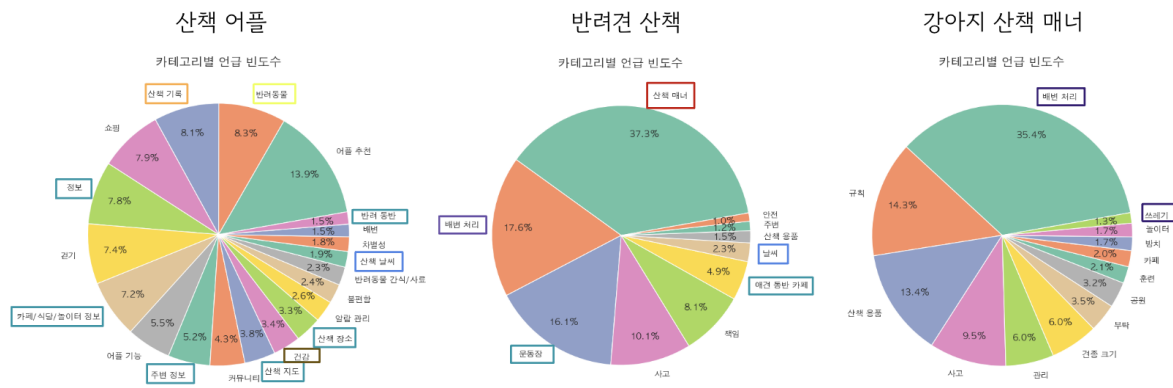
{'오늘오전': 1, '동네': 51, '동네산책': 3, '책중': 4, '사회화': 636, '미숙': 1, '조심': 38, '돌발행동': 5, '만난': 3, '반려견주': 3, '물으시더라구요': 1, '단정': 1, '제가목줄': 1, '있어요': 1, '요라': 1, '질문받으심': 1, '받으심': 1, '하소연': 3, '깡깡거리': 4, '가리키': 2, '뒤돌': 2, '미세': 12, '미세먼지': 12, '방안': 1, '실땀': 1, '이탈': 1, '터네스': 2, '네스': 23, '하네스': 22, '산책중': 4, '라지': 1, '라지사이즈': 1, '미사용': 1, '제품개봉': 1, '개봉': 3, '박스': 5, '팩트': 1, '제품평': 2, '제품평브랜드': 2, '상태미개봉중고': 2, '유무': 4, '유무유무': 2, '실용반려견': 1, '명명이': 3, '서울술': 3, '근처': 44, '상수동': 2, '글목': 4, '같이다니는거지': 1, '다들함': 1, '올밤': 1, '전도': 1, '설치물': 1, '논논': 1, '초미세': 1, '초미세먼지': 1, '지내': 4, '망설이': 1, '산책주의열화갑상': 1, '주의': 36, '어젯밤': 1, '착용': 32, '영화감독배우': 1, '화상': 3, '어린이집': 2, '등원': 2, '공지': 1, '수호': 1, '커피': 16, '열화': 1, '요해': 2, '날씨': 85, '홍은네': 1, '세트': 5, '카메라': 695, '문제': 677, '반려견교육훈련': 642, '교육': 691, '훈련': 677, '발생': 664, '장소': 53, '카페': 1, '활성화': 4, '이벤트': 1, '취소': 3, '가을산책만남': 1, '매니지': 1, '번호': 2, '멤버': 5, '멤버분': 1, '네이버카페': 1, '인증사진': 1, '모집기간': 3, '당첨인원': 2, '당첨상품': 2, '이벤트신청글': 1, '영화': 1, '산책이야기': 1, '캠페인산책': 1, '필수미션': 3, '보호자분': 4, '놀이하기': 1, '개롱': 636, '맷캣': 5, '지키기': 3, '이중문': 1, '이중문설치': 1, '인식표달기': 1, '달기': 3, '견개': 1, '견개구좌': 1, '구좌': 1, '만들기': 5, '기부하기': 1, '박스': 5, '응원': 3, '응원댓글': 1, '신청기재': 1, '목요일': 5, '세트게': 1, '반려견산책이벤트': 1, '일산책': 4, '일산책캠페인': 1, '교육놀이하기': 1, '맷캣지키기': 1, '이중문설치하기': 1, '반려견교육'}

그리고 나서 중복된 의미의 단어들을 연고나 카테기로 묶는 전처리를 진행한 다음 연관 카테기 별 퍼센트를 구하여 파이 그래프로 시각화를 진행 하였다.

카테기별 언급 빈도 수 파이 그래프 분석

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 sns.set(font="AppleGothic",
6         rc={"axes.unicode_minus":False},
7         style='darkgrid')
8
9 sns.set_palette('Set2')
10
11 df_word_freq_sorted['freq'].plot(kind='pie',
12                                 figsize=(7, 8),
13                                 autopct='%1.1f%%', # 퍼센트 % 표시
14                                 startangle=10, # 파이 조각을 나누는 시작점(각도 표시)
15                                 pctdistance=0.8
16                                 )
17
18 plt.title('카테기별 언급 빈도수', size=16)
19 plt.axis('equal') # 파이 차트의 비율을 같게 (원에 가깝게) 조정
20 plt.axis('off')
21 plt.show()
22 plt.savefig('/Users/juheckim/Desktop/coding/mycodesource/Multicampus/4jo/mine/NPL/pie_반려견산책.png', bbox_inches = 'tight')
```

3. 시각화 및 인사이트 도출



기존 산책 어플에 대한 사용자들의 후기를 알고자 “산책 어플” 키워드로 검색하여 크롤링한 게시글 1000건 분석 결과, 반려동물과 산책 어플 간의 큰 상관관계와 산책 기록에 대한 수요, 산책 날씨에 대한 수요, 카페/식당/놀이터 정보와 같은 색상의 박스로 표시된 단어들을 통해 장소 정보에 대한 수요를 확인 할 수 있었다.




반려견 산책에 대한 반려인들의 관심을 알고자 “반려견 산책” 키워드로 검색하여 크롤링한 게시글 1000건 분석 결과, 배변 처리에 대한 관심, 날씨 정보 제공에 대한 수요, 운동장과 같은 색상의 박스로 표시된 단어들을 통해 장소 정보에 대한 수요를 확인 할 수 있었다.

반려견 산책 분석에서 가장 많이 언급된 “강아지 산책 매너” 키워드로 검색하여 크롤링한 게시글 1000건 분석 결과, 배변처리에 대한 관심이 크다는 것을 확인하였다.

위와 같은 결과로 실시간 날씨 정보를 산책 전과 산책 중에 제공하고, 반려 동반 카페/식당, 애견 카페, 놀이터, 동물 병원, 등의 장소 위치 정보를 제공하며 마지막으로, 배변 처리를 돕고자 쓰레기통 위치 정보와 공중 화장실 위치 정보를 제공하기로 하였다.

b. 지도 표시 데이터 수집

1. 공공 데이터 포털

 서울시 주요 공원현황_리스트.csv	크롤링, 수집 데이터
 쓰레기통_리스트.csv	크롤링, 수집 데이터
 공중화장실_리스트.csv	크롤링, 수집 데이터

산책 시 보이는 지도 위에 표시할 데이터들을 공공 데이터 포털에서 모두 다운 받아 동작구, 강남구 지역 데이터만 전처리를 완료했다.

2. 여기다 댕댕이 크롤링

자연어 분석결과 반려견과 함께 방문할 수 있는 장소에 대한 수요를 확인하여 “여기다 댕댕이”이라는 사이트가 가지고 있는 반려견 동반 가능 매장이나 장소를 아래 코드를 통해 크롤링하였다. 애견 동반 카페나 식당, 애견 카페, 병원, 공원, 간식용품점 등 다양한 장소 카테고리가 있었지만, 모든 장소를 다 가져와 사용하기 보다 사용자의 수요를 확인한 애견 동반

카페와 애견 카페, 동물병원, 공원, 반려견 놀이터의 주소만 크롤링하여 데이터를 수집하였다. 이에 “여기다 댕댕이”의 페이지에서 우리가 원하는 장소 카테고리를 누른 후 아래 코드를 실행시키면 해당 카테고리에 있는 장소의 이름과 주소를 모두 가져와서 만들어 놓은 dog_place_name과 dog_place_addr이라는 list에 담기게 된다. 페이지에서 매장 카테고리를 선택하면, 많은 매장 리스트가 나오는데, 리스트를 눌러야 해당 매장의 주소가 나오기 때문에, 리스트의 매장들을 모두 한번씩 눌러 해당 매장 설명 페이지에서 주소를 가져왔고, 주소를 가져온 후 페이지 탭을 삭제하여 본래 리스트 탭으로 이동하는 과정까지 설계하여 넣었다.

```

: # 값을 리스트
dog_place_name = []
dog_place_addr = []

: # 해당 코드를 매장 카테고리 별로 돌려서 사용함
# 리스트 열기
place_list = driver.find_element(By.CSS_SELECTOR, "body > div.app-wrapper > div > div.map-footer > div > div > div:nth-child(1) > button")
place_list.click()
time.sleep(1)
for i in range(1,51,2):

    # 매장 이름
    place_name = driver.find_element(By.CSS_SELECTOR, "#place_popup > a:nth-child("+str(i)+") > div.div-content > div.popup-content > p.title")
    dog_place_name.append(place_name.text)
    print(place_name.text)

    # 매장 주소 정보 페이지 클릭
    place_page = driver.find_element(By.CSS_SELECTOR, "#place_popup > a:nth-child("+str(i)+")")
    place_page.click()
    time.sleep(1)

    # 매장 주소 탭으로 이동
    driver.switch_to.window(driver.window_handles[-1])

    # 매장 주소 크롤링
    place_addr = driver.find_element(By.CSS_SELECTOR, "body > div > div > div.p_info > p:nth-child(5)")
    dog_place_addr.append(place_addr.text)
    print(place_addr.text)
    # 매장주소 탭 닫기

    driver.close()
    time.sleep(1)
    # 리스트로 복귀
    driver.switch_to.window(driver.window_handles[-1])

```

위의 크롤링 과정을 통해서 동작구와 강남구에 있는 애견 동반 카페와 애견 카페, 동물병원, 공원, 반려견 놀이터의 이름과 주소를 가져왔고 이를 데이터 프레임으로 변환한 후 csv파일로 저장하였다.

```
dog_cafe_result = pd.DataFrame({"매장명":dog_place_name, "주소":dog_place_addr})
```

dog_cafe_result

	매장명	주소
0	펫타리움	서울특별시 강남구 청담동 68-7 6, 7층
1	강아지동 애견카페	서울특별시 강남구 논현동 170
2	카페더왈츠	서울특별시 강남구 역삼동 834-8
3	봉브라더스	서울특별시 강남구 논현동 181-12 2층
4	전설의 강아지	서울특별시 강남구 논현동 265-23
5	반려문화	서울특별시 강남구 논현동 85-13

강아지_견종.csv	크롤링, 수집 데이터
견종별산책가이드라인.csv	-폴더 옮김
네이버_애견카페_리뷰_크롤링.csv	지도 표시 데이터, 네이버 리뷰 크롤링 데이터
네이버카페_애견동반카페_리뷰_크롤링.csv	지도 표시 데이터, 네이버 리뷰 크롤링 데이터
댕댕이무료운동장_크롤링_리스트.csv	크롤링, 수집 데이터
동물병원_크롤링_리스트.csv	크롤링, 수집 데이터
애견동반식당_크롤링_리스트.csv	크롤링, 수집 데이터
애견동반카페_리뷰크롤링_방문자가중치.csv	애견카페, 애견동반카페 네이버 크롤링 결과 및 가중치
애견동반카페_추천시스템.csv	공원 가중치 입력한 추천시스템 rpm데이터
애견동반카페_크롤링_리스트.csv	크롤링, 수집 데이터
애견카페_리뷰크롤링_방문자가중치.csv	애견카페, 애견동반카페 네이버 크롤링 결과 및 가중치
애견카페_추천시스템.csv	공원 가중치 입력한 추천시스템 rpm데이터

3. 실시간 날씨 정보 수집 Open API

IoT기기를 통해 산책 중의 온습도를 측정하여 사용자에게 전달하려 하였으나, IoT기기의 실행가능성, 안전성, 정확성 등을 고려하여 IoT기기로 측정하여 보여주는 것이 아닌, “Open Weather”와 “한국환경공단”에서 제공하는 Open API를 통해 정보를 제공하기로 하였다. GPS기기로 사용자의 좌표 정보를 가져와, 해당 좌표가 어느 지역구인지 확인한 후 해당 지역구의 실시간 기상과 온습도, 미세먼지 정보를 제공하였다.

아래의 코드는 임시로 위치 좌표 정보를 넣어 산출된 코드입니다. “Open Weather”의 경우 좌표값만 주면 되지만, “한국환경공단”의 경우 좌표가 아닌 어느 지역구에 있는지를 넣어야하기에, kakao 지도 API를 사용하여 좌표값을 지역구로 빼주는 과정을 먼저 실시하였다.

```
# gps 정보
lat = "37.517445953351"
lon = "127.046798553125"

# 지역구 설정
url = "https://dapi.kakao.com/v2/local/geo/coord2regioncode.json?x="+lon+"&y="+lat+"&input_coord=WGS84"
headers = {"Authorization": "KakaoAK " + "33927ef813f722d8057368bc783fe744"}
api_test = requests.get(url, headers=headers)
url_text = json.loads(api_test.text)
location = url_text['documents'][0]['region_2depth_name']
```

위에서 지역구를 받아오면 “한국환경공단”의 형식에 맞춰 요청하면 해당 지역구의 미세먼지와 초미세먼지를 가져올 수 있게 설정하였고 온습도의 경우에도 “Open Weather”의 형식에 맞춰 요청하여 기상과 온습도 데이터를 1시간 단위로 가져올 수 있게 코드를 구성하였다.

```
## 미세먼지 정보
url = "https://api.odcloud.kr/api/RItnArpltnInforInquireSvc/v1/getMsrstnAcctoRItnMeasureDnsty?numOfRows=1&stationName="+location+"&dataKey = "GTeJTdf4x2FxLUY7gFp1B69Ji%2B17qofxnGFSHJzY3136y7rsS1RW6UjpZhWEoDDC74%2FiNB80YX8pJ2C0HCmU04eQ%3D%3D"
key2 = "GTeJTdf4/xLUY7gFp1B69Ji+17qofxnGFSHJzY3136y7rsS1RW6UjpZhWEoDDC74/iNB80YX8pJ2C0HCmU04eQ=="
url_1 = url+key # 지역 설정은 아직 안함.

html = requests.get(url_1).content

soup = BeautifulSoup(html, "html.parser")

datetime = soup.find('datetime').text
pm10 = soup.find('pm10grade1h').text
pm25 = soup.find('pm25grade1h').text

### 온습도 정보

apiKey = "ec333136c187b5df55402e02ddb12a8"
api = f"https://api.openweathermap.org/data/2.5/weather?lat="+lat+"&lon="+lon+"&appid="+apiKey

result = requests.get(api).content
soup = BeautifulSoup(result, "html.parser")
soup

json_object = json.loads(str(soup))
weather = json_object['weather'][0]['main']
weather_id = json_object['weather'][0]['id']
temp = json_object['main']['temp']
humidity = json_object['main']['humidity']
```

“Open Weather”에서 제공하는 기상정보의 종류가 너무 많아 이를 유사한 날씨끼리 묶어 같은 날씨로 분류되어 넘어갈 수 있게 아래와 같이 코드를 설계하였다.


```

if weather_id in Thunderstorm:
    weather_id = "뇌우"
elif weather_id in Drizzle:
    weather_id = "이슬비"
elif weather_id in Rain:
    weather_id = "비"
elif weather_id in Snow:
    weather_id = "눈"
elif weather_id in Fog:
    weather_id = "안개"
elif weather_id in Squall:
    weather_id = "돌풍"
elif weather_id in Tornado:
    weather_id = "태풍"
elif weather_id in Ash:
    weather_id = "화산재"
elif weather_id in Dust:
    weather_id = "먼지"
else:
    weather_id = "머지"

Thunderstorm = [200,201,202,210,211,212,221,230,231,232] # 뇌우
Drizzle = [300,301,302,310,311,312,313,314,321] # 이슬비
Rain = [500,501,502,503,504,511,520,521,522,531] # 비
Snow = [600,601,602,611,612,613,615,616,620,621,622] # 눈
Fog = [701,721,741] # 안개
Squall = [771] # 돌풍
Tornado = [781] # 태풍
Ash = [762] # 화산재
Dust = [711,731,751,761] # 먼지
Clear = [800] # 맑음
Clouds = [801,802,803,804] # 구름

```

해당 코드를 하나의 함수로 묶었으며, 이를 실행한 결과 다음과 같은 딕셔너리로 return할 수 있게 하였다.

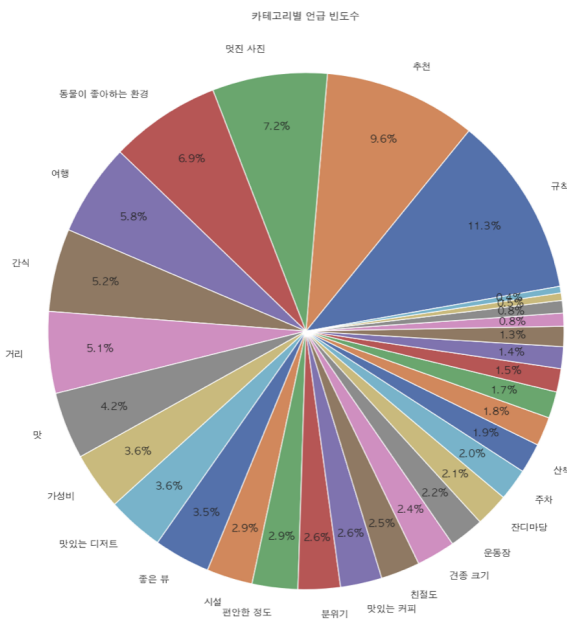
```
{'location': '강남구', 'datetime': '2023-02-09 01:00', 'pm10': '1', 'pm25': '2', 'weather_id': '맑음', 'temp': 270.7, 'humidity': 59}
```

c. 카페 추천 시스템

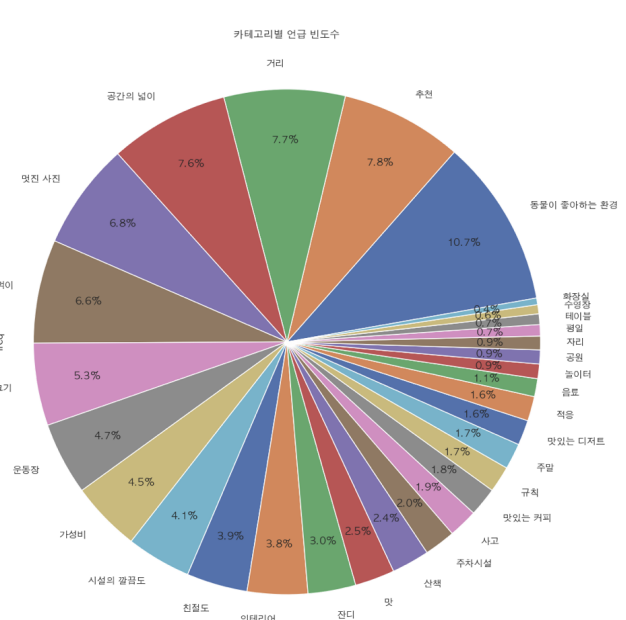
자연어 분석을 통해 산책 시 반려인이 반려견과 함께 휴식할 수 있는 공간을 필요로 한다는 것을 확인하였다. 그래서 사용자의 현위치로부터 특정 거리 안에 있으며 사용자가 선택한 선호도를 고려하여 애견동반카페와 애견카페를 추천할 수 있는 시스템을 계획하였다.

1. 자연어 분석

카페 추천 시스템을 만들기 위해 “애견동반카페”와 “애견카페” 키워드로 자연어 분석을 실시하였다. 네이버 카페에



애견동반카페 자연어 처리 결과



애견카페 자연어 처리 결과

각각의 키워드로 검색한 후, 카페 글 제목과 글 본문, 댓글을 크롤링하여 자연어 분석할 문장들을 수집하였다. 그 결과, 애견동반카페와 애견카페에서 아래와 같은 결과가 나왔고, 멋진 사진, 동물들이 좋아하는 환경, 가성비, 맛있는 디저트, 좋은 뷰,

```

# 네이버 지도 리뷰 방문자 리뷰의 이런 점이 좋았어요 카테고리들
comment_text_list = [
    "음식이 맛있어요", "인테리어가 멋져요", "친절해요", "가성비가 좋아요", "재료가 신선해요", "뷰가 좋아요", "매장이 청결해요", "특별한 날 가기 좋아요",
    "혼밥하기 좋아요", "특별한 메뉴가 있어요", "단체모임 하기 좋아요", "왕이 많아요", "매장이 넓어요", "화장실이 깨끗해요", "주차하기 편해요", "대화하기 좋아요",
    "오래 머무르기 좋아요", "음식이 좋아요", "술이 다양해요", "자분한 분위기에요", "아늑해요", "고기 질이 좋아요", "사진이 잘 나와요", "혼술하기 좋아요",
    "음료가 맛있어요", "기본 안주가 좋아요", "병이 맛있어요", "커피가 맛있어요", "집중하기 좋아요", "좌석이 편해요", "디저트가 맛있어요", "비싼 만큼 가치있어요",
    "품이 잘 되어있어요", "반려동물과 가기 좋아요", "현지 맛에 가까워요", "메뉴 구성이 알차요", "아외 공간이 멋져요", "컨셉이 독특해요"]

```


시설, 분위기, 친절도, 인테리어, 등의 키워드가 네이버 지도의 리뷰의 방문자 리뷰(이런 점이 좋았어요)와 상당 부분 겹친다고 판단하였다. 그래서 네이버 지도의 리뷰를 크롤링 하여 카페 추천에 사용될 점수로 이용하기로 하였다.

2. 네이버 지도 리뷰 크롤링

selenium을 이용해 동적 크롤링을 진행하였다. 네이버 지도의 경우, iframe을 이용하여 frame을 switch하여 “이런 점이 좋았어요”의 각 카테고리별 점수, 방문자리뷰 참여자수, 별점 그리고 별점 참여자 수를 크롤링 하였다.

```

50
59
60 #2가지의 iframe 존재, 경우에 따라 다르게 frame switch
61 if check_exists_by_css_selector("#entryIframe"):
62     iframe = driver.find_element(By.CSS_SELECTOR, '#entryIframe')
63     driver.switch_to.frame(iframe)
64 else:
65     iframe = driver.find_element(By.CSS_SELECTOR, '#searchIframe')
66     driver.switch_to.frame(iframe)
67     key5 = driver.find_element(By.CSS_SELECTOR, ".P7gyV:nth-child(1)")
68     key5.click()
69     time.sleep(2)
70     driver.switch_to.default_content()
71     iframe = driver.find_element(By.CSS_SELECTOR, '#entryIframe')
72     driver.switch_to.frame(iframe)

```

홈	메뉴	리뷰	사진
이런 점이 좋았어요 ①	나도 참여	이런 점이 좋았어요 ①	이런 점이 좋았어요 ①
✓ 632회 (564명 참여)		이런 점이 좋았어요 ①	이런 점이 좋았어요 ①
☺ "카페가 맛있어요"	466	☺ "카페가 맛있어요"	466
☺ "인테리어가 멋지어요"	199	☺ "인테리어가 멋지어요"	199
☺ "친절해요"	107	☺ "친절해요"	107
☺ "음료가 맛있어요"	100	☺ "음료가 맛있어요"	100
☺ "매장이 청결해요"	82	☺ "매장이 청결해요"	82
☺ "부가 좋아요"	63	☺ "부가 좋아요"	63
☺ "대화하기 좋아요"	59	☺ "대화하기 좋아요"	59
☺ "사진이 잘 나와요"	56	☺ "사진이 잘 나와요"	56
☺ "디저트가 맛있어요"	55	☺ "디저트가 맛있어요"	55

네이버카페_매장등판카테고리_리뷰_크롤링																		
매장명	카페가 있었어요	인테리어가 멋지어요	친절해요	음료가 맛있어요	매장이 청결해요	부가 좋아요	대화하기 좋아요	디저트가 맛있어요	사진이 잘 나와요	특별한 메뉴가 있어요	가성비가 좋아요	집중하기 좋아요	화장실이 깨끗해요	좌석이 편해요	리뷰자수	별점	별점자수	
말미커피 도산공원점	447	191	100	94	78	58	56	55	53	31	26	23	23	13	4	545	4.39	562
말미아미아	292	442	117	209	73	78	92	523	306	161	20	2	41	32	17	1000	4.35	1872
말미	230	261	130	178	82	52	131	149	91	51	17	35	42	53	4	608	4.46	1368
홍차가게 황달	22	97	101	123	50	15	42	123	50	53	14	3	28	11	3	191	4.51	246
말미 가루수집점	80	30	58	26	25	5	15	167	8	54	10	3	2	0	0	216	4.81	233
뉴욕라조베리우스	161	164	227	118	108	142	123	0	108	230	41	34	28	40	11	942	4.54	1157
말미도노	16	0	13	13	1	0	10	1	0	1	3	1	0	3	1	32	4.13	36
말미커피신사점	33	20	21	18	12	4	10	18	12	9	0	2	2	3	0	62	4.46	98
카노피 카페	28	6	10	5	4	0	3	0	3	6	6	3	0	1	0	34	4.32	150
카페 노르드	26	27	15	15	15	2	11	14	7	6	2	3	4	5	0	60	4.48	121
카페 노르드 황달	1029	1183	1515	655	1048	92	220	5747	619	1971	348	41	143	218	64	6896	4.44	11719
말미 카페	35	16	41	18	31	4	5	35	9	24	6	0	0	6	8	59	4.81	129
수목원 카페	31	32	7	10	10	6	8	13	11	8	1	6	3	4	0	57	4.54	254
카페엔터프라이즈 논현점	110	93	84	78	67	1	50	29	15	19	7	39	18	30	6	181	4.47	225
말미커피 신사점	74	33	41	28	33	3	15	52	16	22	5	1	9	3	0	133	4.45	274
말미커피 신사점	12	15	74	8	18	2	2	10	8	17	16	0	3	4	3	116	4.48	286
무디점 도산점	77	92	89	35	60	15	32	50	16	40	5	5	20	9	1	251	4.56	208
카페말미	99	56	23	30	20	5	30	31	11	7	4	7	7	7	2	134	4.52	279
말미커피	255	128	81	78	20	2	49	148	34	39	22	10	11	11	1	397	4.51	770
802dessert	7	1	47	3	20	2	1	34	0	33	13	0	2	0	6	45	4.53	50
Small Batch Seoul	39	4	29	9	9	3	2	6	4	8	1	1	2	0	0	46	4.83	18
카페말미	273	176	101	98	109	13	58	99	47	45	6	16	33	19	10	350	4.56	499
말미커피	21	4	46	37	21	3	9	6	1	8	34	2	2	2	0	44	4.73	57
말미도노	28	32	30	9	28	3	9	15	5	6	0	7	7	6	5	50	4.87	21

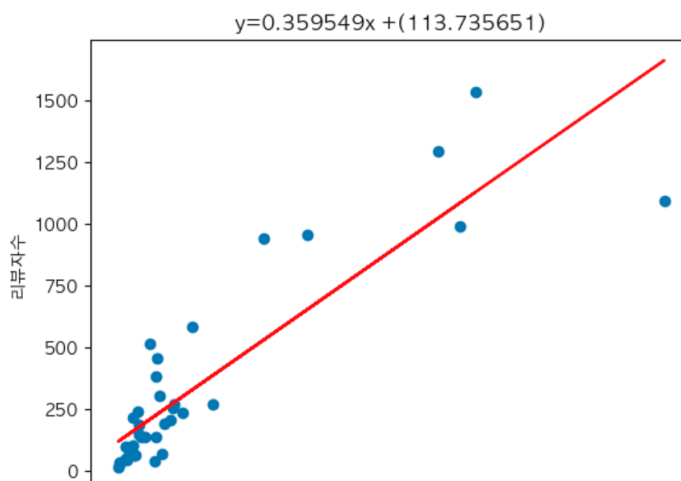
3. 리뷰자수 가중치 계산

네이버 리뷰 크롤링을 통해 얻은 리뷰자수, 별점, 별점자수를 가지고 진행하였다. 별점이 없는 가게들은 최솟값으로 대체하여 사용하였다. 별점자수는 리뷰자수와 상관분석을 통해 예측 별점자수로 대체하여 사용하였다. 별점자수의 예측값이 소숫점이 나왔을 경우엔 소숫점을 모두 버림하여 사용하였다.

```

In [49]:
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 plt.scatter(diner_2['별점자 수'],diner_2['리뷰자수'])
5 fit = np.polyfit(diner_2['별점자 수'], diner_2['리뷰자수'], 1)
6 trend_f = np.poly1d(fit)
7 plt.plot(diner_2['별점자 수'], trend_f(diner_2['별점자 수']),"r-")
8 plt.title("y={:.6f}x + {:.6f}".format(fit[0],fit[1]))
9 plt.xlabel("별점자수")
10 plt.ylabel("리뷰자수")
11 plt.show()

```



그 후, 별점과 별점자수를 곱하여 각 가게들의 전체 별점을 구하였다. 다음으로 모든 가게 리스트의 리뷰자수의 총합을 나누어 방문객 가중치를 구하였다. 값이 너무 작게 나와 편의를 위해 소숫점 네번째 자리까지만 사용하기로 하였다.

$$\text{방문객 가중치} = \text{별점} * \text{별점자수} / \text{리뷰자수총합}$$

위에서 구한 방문객 가중치를 사용하여 최종점수를 구하였다. 먼저 각 가게들 별로 리뷰자수가 모두 달랐기 때문에 기준을 맞추기 위해 리뷰자수와 방문객 가중치를 곱한 값을 사용했다. 그 후 각 칼럼의 숫자로 나누어 추천 시스템에 사용할 최종 점수를 구하였다.

$$\text{최종점수} = \text{sum}(\text{각 칼럼} / \text{리뷰자수} * \text{방문객 가중치})$$

4. 카페 추천 시스템, 공원 근접 카페 추가 가중치 계산

리뷰자 수 가중치를 넣은 카페 추천 시스템의 코사인 유사도를 계산한 후 리뷰자수 가중치 외에 추가로 산책하기 좋은 공원과 거리가 가까운 매장에 추가 가중치를 주었다. 공원 가중치의 경우, 카페 추천 시스템의 선택지인 애견동반카페와 애견카페의 각각 4가지 선택 카테고리 중 대표 컬럼에 표준편차를 구한 후 100m~500m 거리에 공원이 있을 경우, 100m 단위로 표준편차값의 3배~1배까지 0.5배 단위로 가중치를 부여하였다.

첫번째로 강남구와 동작구에 있는 모든 매장과 공원 사이의 거리를 구하였다. 아래의 함수를 통해서 결과물을 만들었으며, 해당 함수의 파라미터 중 df는 모든 카페의 이름과 좌표를 가지고 있는 데이터 프레임이 들어가고 park는 공원의 이름과 좌표가 있는 데이터 프레임이 들어간다. 이중 for문을 이용하여 하나의 매장이 지정되면 모든 공원이 돌아 하나의 카페와 모든 공원 사이의 거리를 계산할 수 있게 하였다.

```
def nearest_data(df, park):
    # 결과프레임 미리 세팅
    result = pd.DataFrame()

    # 모든 카페 회전
    for c_idx in df.index:
        x = df.loc[c_idx].lat
        y = df.loc[c_idx].lng
        xy = (x,y)

        # 하나의 카페에 대한 전체 공원과 거리 계산
        for p_idx in park.index:
            park_name = []
            park_dist = []
            x1 = park.loc[p_idx].lat
            y1 = park.loc[p_idx].lng
            xy1 = (x1,y1)
            # 매장과 공원의 좌표를 기준으로 서로의 거리를 m단위로 계산
            dist = haversine(xy, xy1, unit='m')

            result.loc[c_idx,p_idx] = dist

    return result
```

다음으로 위에 함수를 돌려 나온 결과물 중 모든 레코드, 곧 각각의 카페에서 가장 가까운 공원의 이름과 거리를 가장 가까운 순으로 5개씩을 슬라이싱하여 특정 카페에서 가장 가까운 공원이 무엇이며, 몇 m나 떨어져있는지 보기 위해 딕셔너리로 변화하여 정리하였다. 딕셔너리로 변환한 이유는 각각의 카페에서 가장 가까운 공원은 모두 다를 것이기에 데이터프레임보다 딕셔너리로 정리하여 보고 공유하는 것이 깔끔하다고 판단하여 딕셔너리로 다시 정리하였다.

```
{ '펠트커피 도산공린점': [Index(['도산근린공원', '청담근린공원', '봉은공원', '도곡근린공원', '국립현충원'], dtype='object'),
array([ 112.97571487, 1575.57158985, 2198.4967727 , 4018.32459973,
        6154.63945745])],
'맘마미야': [Index(['도산근린공원', '청담근린공원', '봉은공원', '도곡근린공원', '국립현충원'], dtype='object'),
array([ 141.66777201, 1651.27441307, 2272.71794523, 4063.17056168,
        6107.31624236])],
'정월': [Index(['도곡근린공원', '도산근린공원', '봉은공원', '청담근린공원', '국립현충원'], dtype='object'),
array([2075.94343165, 2403.52903457, 2631.50317466, 2877.86544034,
        4862.36967671])],
'홍차가게 청담': [Index(['청담근린공원', '봉은공원', '도산근린공원', '도곡근린공원', '광평근린공원'], dtype='object'),
array([ 330.45016684, 565.21262109, 1847.78409522, 3296.01279493,
        4768.84755508])],
'월비 가로수길점': [Index(['도산근린공원', '청담근린공원', '봉은공원', '도곡근린공원', '국립현충원'], dtype='object'),
array([1383.42023164, 2791.75258626, 3108.52535413, 3859.14927734,
        4690.60549502])],
'국립현충원': [Index(['도산근린공원', '청담근린공원', '봉은공원', '도곡근린공원', '국립현충원'], dtype='object'),
array([ 112.97571487, 1575.57158985, 2198.4967727 , 4018.32459973,
        6154.63945745])]
```

모든 카페와 가까운 공원과 그 공원과의 거리를 확인하여, 거리에 맞게 가중치를 넣어주었다. 아래의 함수를 기반으로 하였으며 대표 컬럼의 표준편차를 구하여, 공원과의 거리가 특정 거리 안에 있으면 해당 표준편차에 각각 거리에 따른 N 값을 곱한 후 더해주었다.

```
def park_plus(df, std):
    category = ["맛", "사진", "시설", "가격", "전체"]
    for cate in category:
        for idx in df.index:
            dist = df.loc[idx, dist]]
            if dist < 100:
                df.loc[idx, cate+"가중치"] = std[cate]*3 # 1.5
            elif dist < 200:
                df.loc[idx, cate+"가중치"] = std[cate]*2.5
            elif dist < 300:
                df.loc[idx, cate+"가중치"] = std[cate]*2
            elif dist < 400:
                df.loc[idx, cate+"가중치"] = std[cate]*1.5
            elif dist < 500:
                df.loc[idx, cate+"가중치"] = std[cate]*1
            else:
                df.loc[idx, cate+"가중치"] = 0
    return df

# 대표 컬럼의 표준편차 계산
dog_std = dog_rpm_zero[["맛", "사진", "시설", "가격", "전체"]].describe().loc["std"]
cafe_std = cafe_rpm[["맛", "사진", "시설", "가격", "전체"]].describe().loc["std"]

dog_park_1 = park_plus(dog_dict, dog_std)
cafe_park_1 = park_plus(cafe_dict, cafe_std)
```

공원 가중치를 준 후 특정 지점에서 추천시스템을 돌린 결과는 다음과 같다.

수정 전		address	lat	lng	rpm	dist
이오직동맹이 로데오점	서울특별시 강남구 신사동 862-9 1층	37.527315	127.038773	0.112652	607.439458	
아가결라도	서울특별시 강남구 신사동 564-3 1층	37.521687	127.027088	0.108062	645.497407	
니콜스	서울특별시 강남구 신사동 551-11	37.522245	127.023709	0.102035	925.798024	
천장지구	서울특별시 강남구 논현동 93-15 2층	37.522180	127.038350	0.091378	392.922149	
한남베르코 도산	서울특별시 강남구 신사동 647-14 1층 105로	37.525018	127.036962	0.088578	315.976093	
수정 후		address	lat	lng	rpm	dist
이오직동맹이 로데오점	서울특별시 강남구 신사동 862-9 1층	37.527315	127.038773	0.123312	607.439458	
한남베르코 도산	서울특별시 강남구 신사동 647-14 1층 105로	37.525018	127.036962	0.115227	315.976093	
아가결라도	서울특별시 강남구 신사동 564-3 1층	37.521687	127.027088	0.108062	645.497407	
펄트커피 도산공점점	서울특별시 강남구 신사동 645-21	37.525638	127.035435	0.107381	286.050015	
우디집 도산점	서울특별시 강남구 신사동 644-22	37.526549	127.036000	0.106814	398.769695	

공원 가중치가 잘 들어갔으며, 기존대비 공원에 가까운 매장일수록 더 높은 순위를 갖게 되거나 혹은 기존에 없었지만 공원으로 인해 추천될 수 있게 되었다.

5. 추천시스템 알고리즘

아이템 기반 협업 필터링 추천시스템을 이용해 카페를 추천하기로 하였다. 아이템 기반 협업 필터링 추천시스템의 아이템 즉, recode를 가게로, column을 네이버지도 리뷰 카테고리로 설정하였고, 가게에 따라 네이버 지도 리뷰의 카테고리 값이 없는 경우는 '0'으로 대체하여 진행하였다. cosine_similarity를 이용하여 가게 간의 유사도를 산출하였다. 그리고 가게 유사도와 가게의 기존 데이터를 기반으로 모든 가게의 예측 점수를 계산하였다. 그리하여 사용자가 원하는 네이버 지도 리뷰 카테고리를 선택했을 때, 그 카테고리 값의 예측 점수가 높은 가게를 추천하도록 구성하였다. 마지막으로 ,네이버 지도 리뷰 카테고리의 종류가 많아서 연관성이 높은 카테고리들을 묶어 총 4가지의 카테고리로 단순화 하였다.

```
1 def recommend (abc, efg):
2
3     # 애견동반카페와 애견카페 구분
4     if abc == "애견동반카페" :
5         store = cafe.copy(deep=True)
6     else :
7         store = dog.copy(deep=True)
8
9     # 유사도 분석
10    store_count=store.values
11
12    from sklearn.metrics.pairwise import cosine_similarity
13
14    store_sim = cosine_similarity(store_count, store_count)
15    store_sim_df = pd.DataFrame(data=store_sim, index=store.index, columns=store.index)
16
17    store.iloc[0]
18    rating_1st=store.iloc[0]
19
20    import numpy as np
21
22    def predict_rating(rating_arr, store_sim_arr):
23        ratings_pred=rating_arr.dot(store_sim_arr)/np.array([np.abs(store_sim_arr).sum(axis=1)])
24        return ratings_pred
25
26    store3=store.T
27    ratings_pred=predict_rating(store3.values, store_sim_df.values)
28    ratings_pred_matrix = pd.DataFrame(data=ratings_pred, index=store3.index, columns=store3.columns)
29
30    numpy_matrix=ratings_pred_matrix.T.to_numpy()
31
32    # 최종점수와 예측 점수 퍼센트 계산
33    rate=store['가중치'].to_numpy().reshape(len(store.index),1)
34
35    totalRate=numpy_matrix*0.9997+rate*0.0003
36
37    ratings_pred_matrix_np=pd.DataFrame(data=totalRate.T, index=ratings_pred_matrix.index, columns=ratings_pred_matrix.columns)
38
39    # 계산 완료
40    rpm = ratings_pred_matrix_np.T
41
42    # 카테고리 결정
43    if abc == "a":
44        if efg == 1 :
45            category = ['디저트가 맛있어요', '커피가 맛있어요', '음료가 맛있어요']
46        elif efg == 2 :
47            category = ['인테리어가 멋져요', '사진이 잘 나와요', '화장실이 깨끗해요']
48        elif efg == 3 :
49            category = ['동물을 배려한 환경이에요', '공간이 넓어요', '시설이 깔끔해요']
50        else:
51            category = ['가성비가 좋아요', '가격이 합리적이에요', '친절해요']
52    else :
53        if efg == 1:
54            category = ['커피가 맛있어요', '음료가 맛있어요', '디저트가 맛있어요', '특별한 메뉴가 있어요']
55        elif efg == 2:
56            category = ['인테리어가 멋져요', '뷰가 좋아요', '사진이 잘 나와요']
57        elif efg == 3:
58            category = ['화장실이 깨끗해요', '좌석이 편해요', '집중하기 좋아요']
59        else :
60            category = ['가성비가 좋아요', '대화하기 좋아요', '친절해요']
61
62    list = pd.DataFrame()
63    for i in category:
64        i = pd.DataFrame(rpm[i].sort_values(ascending=False)[:6])
65        i.drop(i, axis=1, inplace=True)
66        list = pd.concat([list, i])
67
68    store_list = list.drop_duplicates()
69    display(store_list)
```

6. 추천 시스템 모듈화

위에서 만든 추천 시스템을 통채로 aws에 넣어 사용자가 요청할 때마다 돌아가는 것은 상당히 비효율적이기 때문에, 코사인 유사도를 구한 값을 csv파일로 만들어서 요청을 받았을 때, 바로 해당 파일을 읽어 추천하는 방법으로 진행하였다. 그 결과 추천 시스템을 요청하는 코드파일 하나와 요청 받아 추천 카페를 넘겨주는 함수 코드파일 하나를 각각 만들어 정리하게 되었다.

첫번째 요청 경우, 사용자의 위치 좌표와 사용자가 선택한 카페 종류와 카테고리를 받아와 조건에 맞춰 상위 5개의 카페를 요청하는 코드로 다음과 같다. 사용자의 좌표는 결과물 산출을 위해 임시로 넣어두었다.

```
#!/ 5개의 카페를 추천해주는 함수가 있는 파일을 import
from .recommend import recommend

def result_1(request):
    # 사용자가 누른 카테고리를 받아오는 것.
    queryinput = request.GET
    # 사용자 좌표
    user_coordinate = (37.5119303471109, 127.03808107256)
    lat = user_coordinate[0]
    lng = user_coordinate[1]

    if queryinput:
        # 장소 / 애견카페 혹은 애견동반카페
        place = queryinput.get("place")
        # 좋아하는 카테고리
        like = queryinput.get("category")
        # 원하는 거리
        distance = float(queryinput.get("distance"))

        # 추천 시스템 알고리즘 함수 from .recommend import recommend
        context = recommend(place, like, distance, lat, lng)
        # 장소, 카테고리, 거리, 좌표

    # 사용자가 선택하지 않았을 경우 그냥 위치를 띄워주는 것.
    else:
        imsi = (lat, lng) # 실시간으로 받아 올 좌표
        x = imsi[0]
        y = imsi[1]
        context = {
            "x": x,
            "y": y
        }

    return render(request, "queryoutput_test_1.html", context)
```

위에서 import하는 함수, 즉 요청을 받아 상위 5개의 카페를 주는 함수는 추천 시스템의 코사인 유사도가 계산된 데이터를 읽어와 돌아가는 과정으로 추천시스템 코드와 똑같으며, rpm을 계산하는 과정만 생략되어있다.

3. 개선 사항

a. 추천 시스템

사용자 누적 기록이 없어서 아이템 기반 협업 추천 시스템 알고리즘 밖에 쓰지 못했지만, 추후에 사용자의 리뷰나 평점 기능이 추가되어 사용자 기반 협업 추천 시스템으로 확장될 수 있다.

b. 누적 센서 데이터 분석

사용자 산책 누적 기록이 쌓인다면, GPS 센서를 통해 사용자들이 주로 산책을 많이 나가는 장소 데이터, 산책 패턴 데이터 등을 분석해 어플 사용자들에게 반려견 산책이 잦은 구역과 잦은 시간, 드문 구역과 드문 시간 정보를 제공할 수 있다.

c. 산책지수

산책하기 좋은 날씨에 대한 기준을 만들어 그날의 온도나 기상, 미세먼지, 자외선 등을 종합하여 야외에서 산책하기 적합한 날씨인지 지수화하여 사용자에게 전달하는 것이다. 만약 기준에 적합하지 않은 날씨라면, 실내 산책이 가능한 장소를 추천해줄 수 있다.

d. 경로 추천 시스템

사용자가 산책 중, 원하는 서비스를 선택한 후 선택한 서비스를 모두 거쳐가는 경로 추천 알고리즘을 구현한다. 사용자가 지정한 장소까지의 최단 거리, 최단 시간이나 사용자가 원하는 키로수, 산책 시간 등을 기준으로 정하여 사용자가 원하는 기준을 선택하여 경로를 추천해줄 수 있다.

4. 데이터 목록

여기다 댕댕이, 댕댕지도 (애견카페, 반려 동반 카페, 반려 동반 식당, 동물 병원, 무료 놀이터)

https://shop.yeogida-dog.com/dangcash/map?b_region=seoul&m_region=22&category=1&c=n

견종 백과사전 <https://www.akc.org/dog-breeds/>

견종별 산책 가이드 라인

<https://fairmountpetservice.com/Blog/pet-services-blog/dog-walking/dog-exercise-needs-breed-guide-chart/>

쓰레기통 <https://www.data.go.kr/data/15038054/fileData.do?recommendDataYn=Y>

화장실 <https://www.data.go.kr/data/15012892/standard.do?recommendDataYn=Y>

Open Weather <https://openweathermap.org/api>

한국환경공단 <https://www.keco.or.kr/kr/main/index.do>

네이버 카페 글 & 댓글

<https://section.cafe.naver.com/ca-fe/home/search/articles?q=%EB%B0%98%EB%A0%A4%EA%B2%AC%20%EC%82%B0%EC%B1%85&pr=6>

네이버 지도 리뷰 <https://map.naver.com/v5/entry/place/1051733906?c=15.0.0.0.dh>

5. 참고 문헌

황원경, 손광표, 한국 반려동물보고서-반려가구 현황과 노령견 양육실태, KB금융지주 경영연구소, 2021 .

오연서, 박진호, 감각 자극에 따른 반려견의 정서 상태 연구, 한국콘텐츠학회논문지, 2020.

박가현, 성장하는 펫케어 산업 최신트렌드와 우리 기업의 글로벌 경쟁력 강화방안, 한국무역협회, 2022.

01. 자연어 처리(natural language processing) 준비하기 - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

형태소 분석 및 품사 태깅 — KoNLPy 0.6.0 documentation

워드 클라우드(Word Cloud) 생성하기(with 파이썬 DataFrame) (tistory.com)