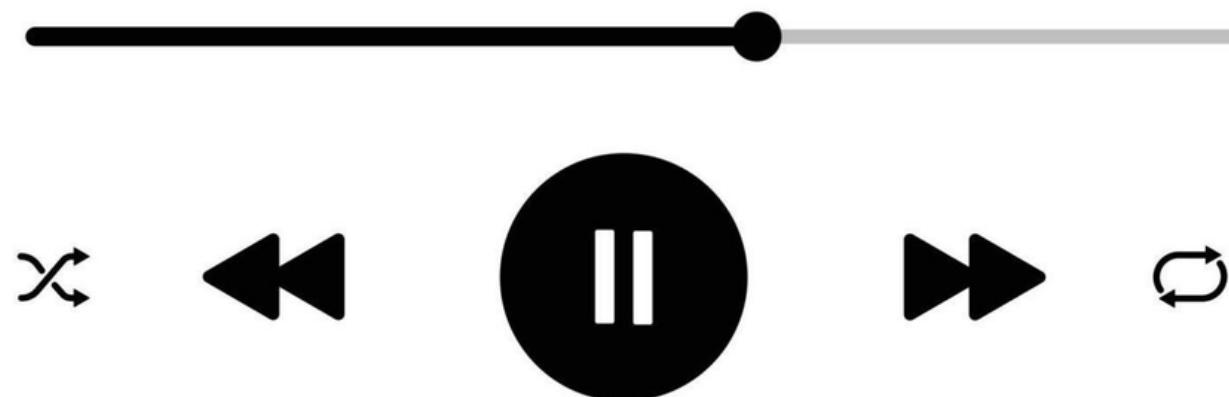


# Weather Can "Pic" Your Music



직접 찍은 날씨 사진에 어울리는 노래 추천 시스템

김기남 | 김하나 | 최주희





# Table of Content

- 주제 선정
- 데이터 수집
- 데이터 전처리
- 추천 방식
- 이미지 분류 - CNN
- 전체 구현 - streamlit
- 의의 / 한계점 및 발전 가능성





01

## 주제 선정 배경

- 날씨는 우리의 감정과 밀접한 관계
- 날씨에 어울리는 노래를 주변 사진을 찍어 즉각적으로 추천 받을 수 있다면 좋겠다는 바람에서 주제를 선정

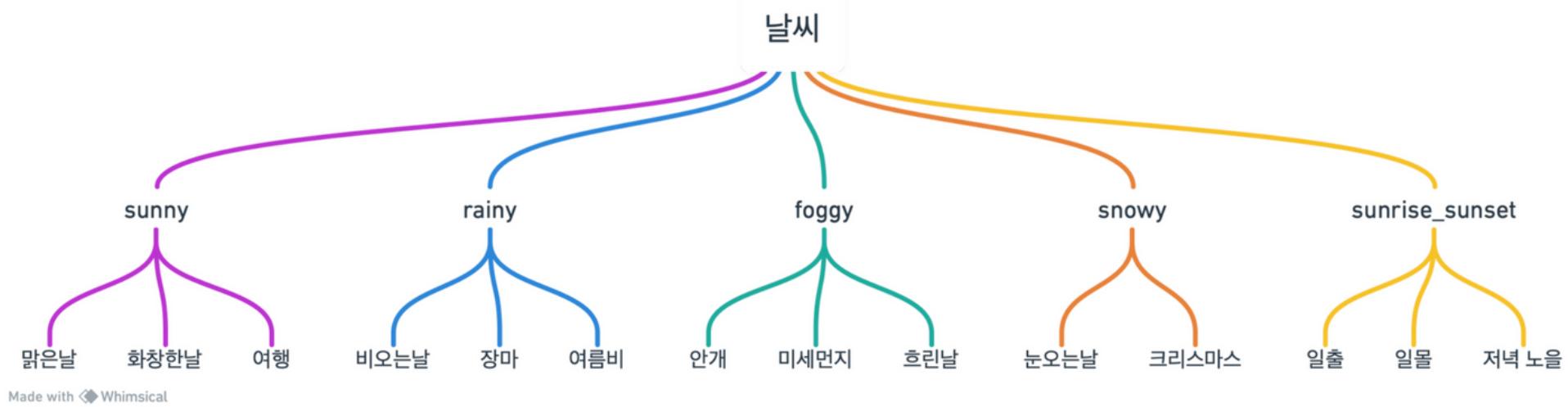




## 02

# 데이터 수집-플레이리스트

### 1) 5개의 날씨 label에 알맞는 플레이리스트 검색을 위한 세부 검색어 설정



**sunny:** 맑은날, 화창한날, 여행

**rainy:** 비오는날, 장마, 여름비

**foggy:** 안개, 미세먼지, 흐린날

**snowy:** 눈오는날, 크리스마스

**sunrise\_sunset:** 일출, 일몰, 저녁 노을, 화창한 아침, 노을

### 2) Melon과 genie 2개의 사이트에서 각 검색어를 넣은 후 나오는 플레이리스트 속 노래들의 정보 크롤링 진행

- 제목
- 아티스트
- 플레이리스트에 있는 세부 태그 (#비오는날 #감성 ...)
- 음원 좋아요 수
- 앨범 커버 이미지





## 02

# 데이터 수집-플레이리스트

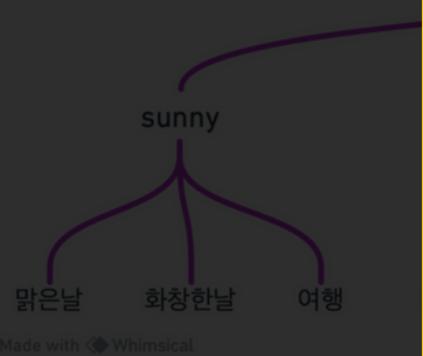
1) 5개의 날씨 `label`에 아마도 프레이리스트 검색을 위한 세부 검색어 선정

## Selenium과 BeautifulSoup의 조합

Selenium	Beautifulsoap
웹 동작	html 정보 파싱
라이브러리 자체가 무거움, 자주 막힘	심플함, 빠름

2) 각 검색어를

- 제목
- 아티스트
- 플레이리스트에 있는 세부 태그 (#비오는날 #감성 ...)
- 음원 좋아요 수
- 앨범 커버 이미지



Made with Whimsical

저녁 노을, 화창한 아침, 노을





# 사이트마다 다른 플레이리스트 구조

## Melon genie

**비 오는 날마다 듣는 아끼는 플리**  
DJ eternel

2023.08.24 수정

#비 #비오는날 #드라이브 #감성 #여름 #버스 #여행 #흐린날 #흐림

834 신고 >

수록곡 (23)

번호	곡정보	앨범	좋아요	듣기	담기	다운	뮤비
1	Reality	La boum (Bande originale ...	7,663				
2	Falling For You	Falling For You	3,109				
3	10,000 Hours	Good Things	144,940				
4	스パー클	スパークル	1,718				

플레이리스트 한 페이지 안에 좋아요 수를 포함한 모든 정보가 다 갖춰져 있음  
requests 라이브러리로 웹 정보를 받고, BeautifulSoup로 파싱  
시간 소요↓

**비 좀 맞으면 어때** 이 리듬에 툭툭 털지 뭐

젖은 마음 툭툭 털어내기 좋은 적당한 리듬과 청량함을 가진 락 플리

제작자	DJ 자몽
곡수	15곡
조회수	60,195
최초생성	2023.08.22
업데이트	2023.08.22
연관태그	#오늘의선곡 #DJ 자몽 #락/메탈 #출/퇴근길 #드라이브 #거리 #기분전환 #아침 #오후 #밝은 #신나는 #여름 #비오는날 #2010년대 #2020년대 #TODAY PICK #해외음악

수록곡

번호	곡정보	듣기	추가	담기	다운	뮤비	더보기
1	Burndt Jamb (Album Ver.) Weezer   Maladroit						
2	Pick A Part That's New Stereophonics   Performance & Cocktails						
3	Ain't It Fun Paramore   Paramore						
4	Barbaric Blur   The Ballad of Darren						

플레이리스트에 각 노래의 좋아요 수 표시 X,  
글 정보 버튼을 눌러 링크로 연결된 페이지에 들어가 추가적인 웹 동작 필요  
시간 소요↑





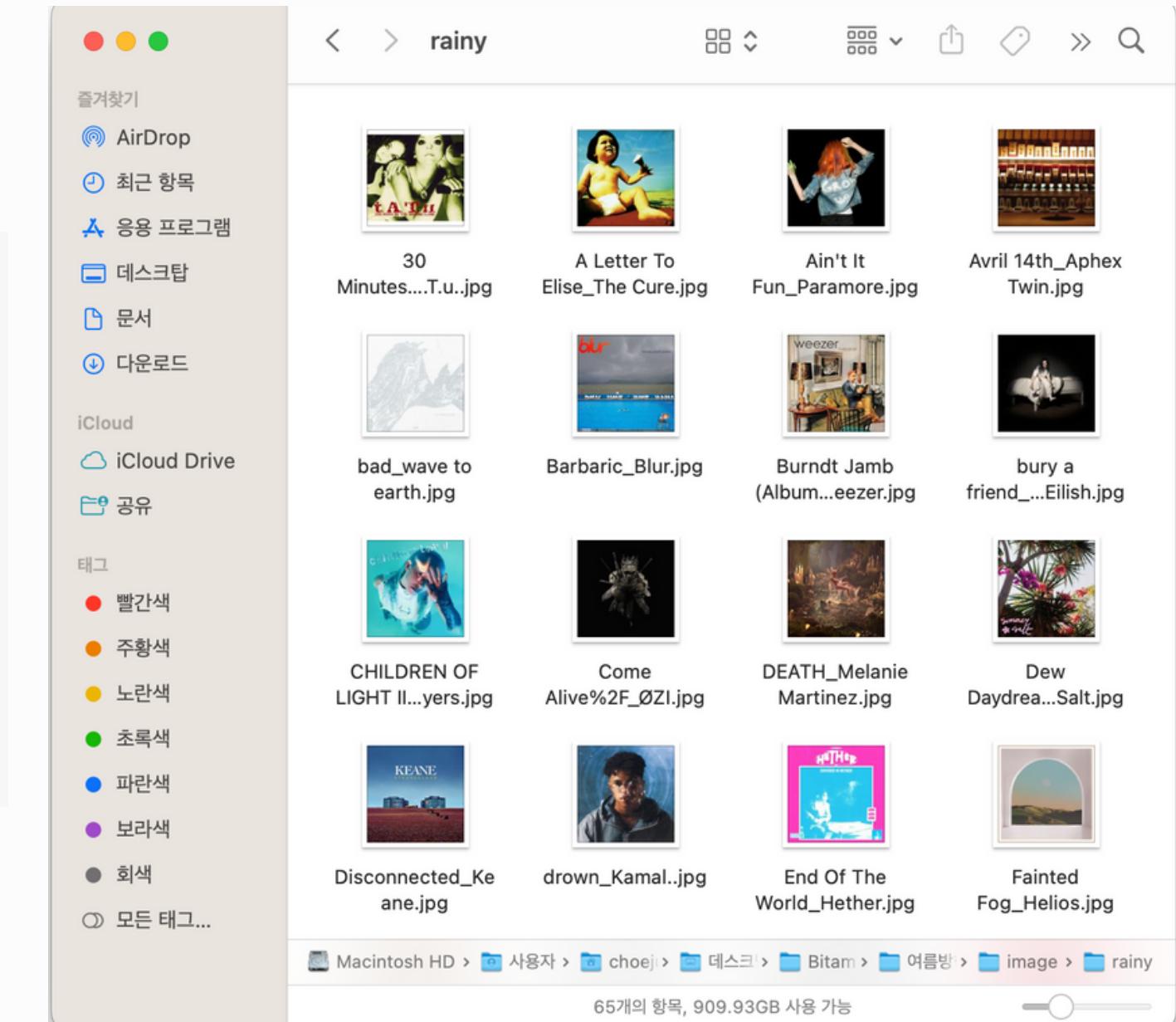
# 앨범 커버 이미지 저장

```
def sanitize_filename(filename):
    # Define a regular expression pattern
    # for characters not allowed in filenames
    invalid_chars = r'[\/*?<>|]'

    # Replace invalid characters with '%2F' using re.sub()
    new_filename = re.sub(invalid_chars, '%2F', filename)

    return new_filename
```

- 앨범 이미지 파일은 '노래 제목\_아티스트 이름.jpg'으로 저장.
- 그러나 파일명에 사용할 수 없는 특수문자(ex. \/:/\*?<>|)가 노래 제목 or 아티스트 이름에 들어가 있는 경우 존재. (ex.NO:EL, HA:TFELT)
- '%2F'로 바꿔서 저장.





## 03

# 데이터 수집-날씨 이미지 데이터

### 1) kaggle

- Multi-class Weather Dataset
- Weather Classification Dataset
- Weather Image Recognition Dataset

Multi-class Weather Dataset  
Image Dataset provides a platform for recognizing different weather conditions.  
[k https://www.kaggle.com/datasets/pratik2901/multiclass-weather-datas...](https://www.kaggle.com/datasets/pratik2901/multiclass-weather-datas...)

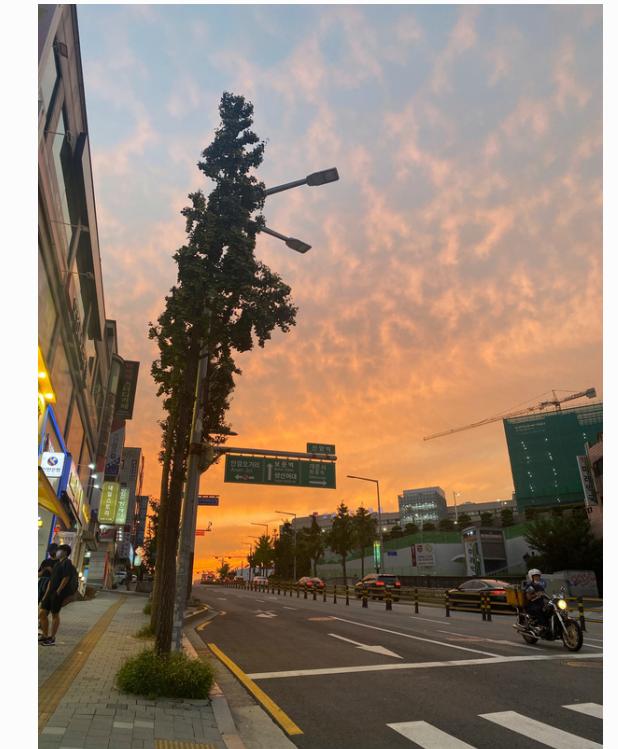


### 2) 수동으로 이미지를 수집하여 라벨링 작업

- 구글 이미지 크롤링
- 개인적으로 직접 촬영한 이미지



sunny



sunrise\_sunset





## 04

# 데이터 전처리

추천 시스템에 필요한 연관성 매개변수 제작 필요

01

**number**

해당 노래(제목, 아티스트 정보)가 각 label로 판단된 적이 몇 번 있는가?

ex) '비도 오고 그래서'-헤이즈: 'rainy' label이 달린 50개의 플레이리스트에서 등장

02

**similar**

**Word2Vec**을 사용하여 단어 임베딩 진행

각 노래마다 specific\_tag 단어들이 해당 label과 연관이 있음을 가정. 특정 label에 해당하는 태그가 다른 노래에서도 많이 나올수록 유사도가 높아짐.

03

**likes**

각 노래의 좋아요 수. 인기가 많다고 해서 해당 label과의 연관성이 높다고 할 수 없음.  
다른 변수들에 비해 낮은 가중치 부여.





# number 변수 만들기

```
unique_pl = playlist[['title', 'artist', 'label']].value_counts().to_frame(name='number')
unique_pl.head()
```

number			
title	artist	label	number
비도 오고 그래서 (Feat. 신용재)	헤이즈 (Heize)	rainy	50
비	풀킴	rainy	46
Rain	태연 (TAEYEON)	rainy	44
우산 (Feat. 윤하) 에픽하이 (EPIK HIGH)	에픽하이 (EPIK HIGH)	rainy	40
All I Want for Christmas Is You	Mariah Carey	snowy	40

```
merged_df = pd.merge(playlist, unique_pl, on=['title', 'artist', 'label'])
```

	title	artist	likes	tag	specific_tag	label	number
0	pony	잔나비	14090	맑은날 ['드라이브', '기분전환', '출퇴근길', '여행', '산책', '휴식', '힐링...]	sunny	2	
1	pony	잔나비	14093	여행 ['드라이브', '기분전환', '출퇴근길', '여행', '산책', '휴식', '힐링...]	sunny	2	
2	한강에서 (Feat. BIG Naughty)	풀킴	25744	맑은날 ['드라이브', '기분전환', '출퇴근길', '여행', '산책', '휴식', '힐링...]	sunny	6	
3	한강에서 (Feat. BIG Naughty)	풀킴	25746	여행 ['드라이브', '기분전환', '출퇴근길', '여행', '산책', '휴식', '힐링...]	sunny	6	
4	한강에서 (Feat. BIG Naughty)	풀킴	25746	여행 ['드라이브', '기분전환', '운전', '외출', '여행', '힐링', '상쾌한...]	sunny	6	
...	...	...	...	...	...	...	...
33801	Against The Wind (Radio Edit)	Together In Hope	117	노을 ['아경', '혼술', '잔잔한', '감성', '새벽', '센치', '퇴근길', '...]	sunrise_sunset	1	
33802	Rainy Days (Feat. Bindi X)	Debauch	14	노을 ['아경', '혼술', '잔잔한', '감성', '새벽', '센치', '퇴근길', '...]	sunrise_sunset	1	
33803	Place In Heaven (Live)	Jasmine Kelly	11	노을 ['아경', '혼술', '잔잔한', '감성', '새벽', '센치', '퇴근길', '...]	sunrise_sunset	1	
33804	you're the only one	keepitinside	109	노을 ['아경', '혼술', '잔잔한', '감성', '새벽', '센치', '퇴근길', '...]	sunrise_sunset	1	
33805	Hold It Still	Mark Diamond	94	노을 ['아경', '혼술', '잔잔한', '감성', '새벽', '센치', '퇴근길', '...]	sunrise_sunset	1	

제목, 아티스트, label을 기준으로 value\_counts() -> 데이터프레임으로 변환  
(name = 'number') => 이후 기준 데이터프레임에 merge 진행

"아직 제목, 아티스트, label 기준으로 중복된 행 제거는 못한 상태"  
ex) 한강에서-풀킴-sunny 조합이 6개. specific\_tag는 조금씩 다름.

# 중복 행 제거

```
def special_tag(df):
    total = []
    small = []
    numlist = df['number'].tolist()
    taglist = df['specific_tag'].tolist()
    titlelist = df['title'].tolist()
    artistlist = df['artist'].tolist()
    labellist = df['label'].tolist()

    for i in range(len(df)-1):
        if titlelist[i] == titlelist[i+1] and \
            artistlist[i] == artistlist[i+1] and \
            labellist[i] == labellist[i+1]:
            for k in taglist[i]:
                if k in small:
                    continue
                else:
                    small.append(k)
        else:
            for k in taglist[i]:
                if k in small:
                    continue
                else:
                    small.append(k)
        total.append(small)
        small = []

    num1 = numlist[-1]
    #마지막 행 처리
    for j in range(num1):
        for k in taglist[-j-1]:
            if k in small:
                continue
            else:
                small.append(k)

    total.append(small)
    return total
```

**special\_tag 함수:** 제목/가수/label이 같은 데이터프레임 행들의 specific\_tag 통합

앞뒤로 제목, 아티스트, label이 모두 같다면  
small 리스트에 각 태그 고유값 추가.

같지 않으면 small을 total에 추가. small을  
다시 빈 리스트로 지정.

=> total에는 중복이 제거된 각 노래마다의  
총 specific\_tag들이 담겨 있음

마지막 i는 [i+1] 인덱싱이 안되기 때문에  
따로 진행





```
def remove_duplicate(lst):
    no_duplicate = []
    for i in lst:
        plus = 1
        no_duplicate.append(i)
        while plus < i:
            lst.remove(lst[i+plus-1])
            plus += 1
    return no_duplicate
```

```
def preprocessing(df):
    # 문자열에서 리스트로 변환
    for i in range(len(df)):
        df['specific_tag'][i] = eval(df['specific_tag'][i])

    #number가 1과 1이 아닌 것 구분
    df_1 = df[df['number']==1]
    df_1.drop('tag', axis=1, inplace=True)
    df_2 = df[df['number']!=1]
    df_2.drop('tag', axis=1, inplace=True)

    #number와 title, label로 정렬, 중복 제거된 데이터와 순서가 같아야 함.
    df_2 = df_2.sort_values(by=['number', 'title'])
    unique_pl = df_2[['title', 'artist', 'label']].value_counts().to_frame(name='number')
    unique_pl = unique_pl.sort_values(by=['number', 'title'])

    #special tag가 합쳐진 리스트 뽑기
    special_taglist = special_tag(df_2)

    #likes 수 중복되지 않게 리스트 만들기
    numlist = df_2['number'].tolist()
    rem_dup_list = remove_duplicate(numlist)
    likelist = df_2['likes'].tolist()
    rem_dup_like_list = []
    plus = 0
    for i in rem_dup_list:
        rem_dup_like_list.append(likelist[plus])
        plus += i

    # unique_pl의 specific_tag, likes 붙이기
    unique_pl.reset_index(inplace=True)
    unique_pl['likes'] = rem_dup_like_list
    unique_pl['specific_tag'] = special_taglist
    # 변수 순서 바꿔주기
    unique_pl = unique_pl[['title', 'artist', 'specific_tag', 'label', 'number', 'likes']]

    df_3 = pd.concat([df_1, unique_pl], ignore_index = True)
    return df_3
```

## remove\_duplicate 함수:

number 변수의 중복을 제거한 후 새로운 리스트로 변환하는 함수

## preprocessing 함수: 최종 전처리 함수

['specific\_tag']: 형태는 리스트지만 실제로는 전부 문자형. eval() 함수로 리스트로 변환

number가 1인 것은 그대로 붙일 수 있음. 2인 것을 기준으로 추가 전처리 진행

number, title, label 기준으로 정렬

special\_taglist, rem\_dup\_list, rem\_dup\_like\_list 도출  
(모두 중복을 없앤 리스트)

각 리스트는 다시 최종 df의 변수로 지정

number가 1인 df와 전처리된 df끼리 결합.



# Word2Vec으로 연관성 지표 제작

## 1. 토큰화하기 전 전처리

```
text_columns = ['specific_tag', 'label']
```

```
text = df[text_columns]
```

```
text.head()
```

	specific_tag	label
0	['미세먼지', '먼지', '저리가', '날려버려', '날씨', '신나는']	foggy
1	['클래식', '여름', '연주곡', '장마', '휴식']	rainy
2	['노을', '케이팝', '감성', '해질녘', '잔잔한', '휴식', '힐링', ...]	sunrise_sunset
3	['가요', '여름', '여행', '기분전환', '드라이브', '휴가', '시원한'...]	sunny
4	['비오는날', '봄비', '비노래', '봄', '봄노래', '그리움', '비내리는...']	rainy

```
# text의 모든 칼럼들을 1개의 칼럼으로 공백을 기준으로 합쳐줌  
text['sentence'] = text[text_columns].apply(lambda row: ' '.join(row.values.astype(str)), axis = 1)
```

```
# 전처리 함수 선언  
def preprocessing(text):  
    # 개행문자 제거  
    text = re.sub('\\\\n', ' ', text)  
    # 한글, 영문만 남기고 모두 제거하도록 합니다.  
    text = re.sub('[^가-힣-ㅎㅏ-ㅣ a-zA-Z]', ' ', text)  
    return text
```

```
# 텍스트 학습전 전처리 적용  
sentences=text['sentence'].apply(preprocessing)
```

```
sentences
```

0	미세먼지	먼지	저리가	날려버려	날씨	신나는	foggy
1	클래식	여름	연주곡	장마	휴식	rainy	...
2	노을	케이팝	감성	해질녘	잔잔한	휴식	힐링
3	가요	여름	여행	기분전환	드라이브	휴가	시원한
4	비오는날	봄비	비노래	봄	봄노래	그리움	비내리는...
23214	크리스마스	겨울	감성	기분전환	휴식	힐링	따뜻한...
23215	드라이브	기분전환	운전	외출	여행	힐링	상쾌한...
23216	크리스마스	겨울	캐롤	인디	알앤비	일상	소품샵...
23217	아이돌	보이그룹	캐롤	청량한	상큼한	크리스마스	...
23218	감수성	가슴이	먹먹해지는	멜로디	눈오는날	밤중	누군가 문득...

Name: sentence, Length: 23219, dtype: object

```
# 중복되는 줄 제거  
sentences.drop_duplicates(inplace = True)
```





# Word2Vec으로 연관성 지표 제작

## 2. sentence 데이터 토큰화

```
tokenizer = WordPunctTokenizer()
tokens = sentences.apply(tokenizer.tokenize)
tokens[:]

0      [미세먼지, 먼지, 저리가, 날려버려, 날씨, 신나는, foggy]
1      [클래식, 여름, 연주곡, 장마, 휴식, rainy]
2      [노을, 케이팝, 감성, 해질녘, 잔잔한, 휴식, 힐링, 한강, 아이돌, 저녁, s...
3      [가요, 여름, 여행, 기분전환, 드라이브, 휴가, 시원한, 청량한, 더위, 명곡, ...
4      [비오는날, 봄비, 비노래, 봄, 봄노래, 그리움, 비내리는날, 여름비, 설레임, ...

23209     [눈오는날, 좋은노래, 아이랑, 아기랑, 크리스마스, 괜찮아, snowy]
23210     [크리스마스송, 캐롤송, 분위기, 있는, 분위기깡패, 눈, 오는날, 화이트, 크리스...
23212     [겨울, 발라드, 년대, 추운, 눈, 눈오는날, 사랑, 이별, 따뜻한, 감성, 기분...
23213     [눈, 이별, 잔잔한, 발라드, 마음, 겨울, 추위, 추억, 눈오는날, 겨울감성, ...
23214     [크리스마스, 겨울, 감성, 기분전환, 휴식, 힐링, 따뜻한, 캐롤, 설렘, 드라이...

Name: sentence, Length: 3567, dtype: object
```

WordPunctTokenizer()를 활용해 앞서 전처리한  
sentences 데이터 토큰화

## 3. Word2Vec 학습

```
model = word2vec.Word2Vec(tokens, min_count=1, workers = 4)
```

```
# 모델 학습에 쓰인 단어들 개수 (중복X)
len(model.wv.key_to_index)
```

```
1186
```

```
model.wv.key_to_index
```

```
{'감성': 0,
'기분전환': 1,
'드라이브': 2,
'휴식': 3,
'잔잔한': 4,
'비오는날': 5,
'여름': 6,
'카페': 7,
'sunny': 8,
'신나는': 9,
'힐링': 10,
'비': 11,
```

Word2Vec에 토큰을 입력으로 넣고 모델 학습 진행



# Word2Vec으로 연관성 지표 제작

## 4. 연관성 지표(similar) 제작

```
# Querying the model for top 10 words similar to "sunny"
similar_to_sunny = model.wv.most_similar("sunny", topn=10) if "sunny" in model.wv.key_to_index else "Word not in vocabulary"

similar_to_sunny
[('경쾌한', 0.9917008876800537),
 ('활기찬', 0.9838064908981323),
 ('바캉스', 0.9832884669303894),
 ('기분좋은음악', 0.981160044670105),
 ('노동요', 0.9809963703155518),
 ('여유', 0.9780014157295227),
 ('Mix', 0.9771247506141663),
 ('맑은날', 0.9769969582557678),
 ('달달한', 0.9768879413604736),
 ('휴양지', 0.9756937026977539)]
```

```
all_similar_to_sunny = model.wv.most_similar("sunny", topn=1186) if "sunny" in model.wv.key_to_index else []
all_similar_to_foggy = model.wv.most_similar("foggy", topn=1186) if "foggy" in model.wv.key_to_index else []
all_similar_to_rainy = model.wv.most_similar("rainy", topn=1186) if "rainy" in model.wv.key_to_index else []
all_similar_to_snowy = model.wv.most_similar("snowy", topn=1186) if "snowy" in model.wv.key_to_index else []
all_similar_to_sunrise = model.wv.most_similar("sunrise", topn=1186) if "sunrise" in model.wv.key_to_index else []
all_similar_to_sunset = model.wv.most_similar("sunset", topn=1186) if "sunset" in model.wv.key_to_index else []
```

```
def list_to_dictionary(weather_list):
    dictionary = {}
    for i in weather_list:
        key = i[0]
        value = i[1]
        dictionary[key] = value
    return dictionary
```

```
sunny_dict = list_to_dictionary(all_similar_to_sunny)
foggy_dict = list_to_dictionary(all_similar_to_foggy)
rainy_dict = list_to_dictionary(all_similar_to_rainy)
snowy_dict = list_to_dictionary(all_similar_to_snowy)
sunrise_dict = list_to_dictionary(all_similar_to_sunrise)
sunset_dict = list_to_dictionary(all_similar_to_sunset)
```

'sunny'라는 단어와 가장 유사한 단어 top10

해당 label에 대한 모든 specific\_tag들의 유사도  
추출(topn=1186, 중복 없는 단어들 개수)

딕셔너리 형태로 제작





# Word2Vec으로 연관성 지표 제작

## 4. 연관성 지표(similar) 제작

```
#sunrise와 sunset은 하나의 label로 되어 있기 때문에 다시 전처리해서 딕셔너리 합침  
#(2개의 값 평균으로 설정. sunrise와 sunset의 경우 본인의 딕셔너리에서 값을 1로 두고 계산)  
  
sunrise_sunset_dict = {}  
for key, value in sunrise_dict.items():  
    if key == 'sunset':  
        sunset_value = 1  
    else:  
        sunset_value = sunset_dict[key]  
    avg = (value + sunset_value) / 2  
    sunrise_sunset_dict[key] = avg  
  
sunrise_sunset_dict['sunrise'] = (1+sunset_dict['sunrise']) / 2  
sunrise_sunset_dict
```

```
{'sunset': 0.9891597926616669,  
'해질녘': 0.9806974530220032,  
'노을': 0.9786683619022369,  
'글귀': 0.977260410785675,  
'한강': 0.9773780405521393,  
'지하철': 0.9761185944080353,  
'저녁': 0.9765937626361847,  
'여름끝자락': 0.9707570970058441,  
'퇴근길': 0.9743486642837524,  
'여름밤': 0.9542985558509827,  
'가을': 0.9545343816280365,  
'버스': 0.9389948546886444,
```

sunrise\_sunset의 경우, 전처리 과정에서  
두 단어가 분리됨.

=> 각 딕셔너리 내의 값들의 평균을  
sunrise\_sunset\_dict에 추가

=> 본인의 딕셔너리에서 (본인 점수 1 +  
다른 레이블 점수) / 2

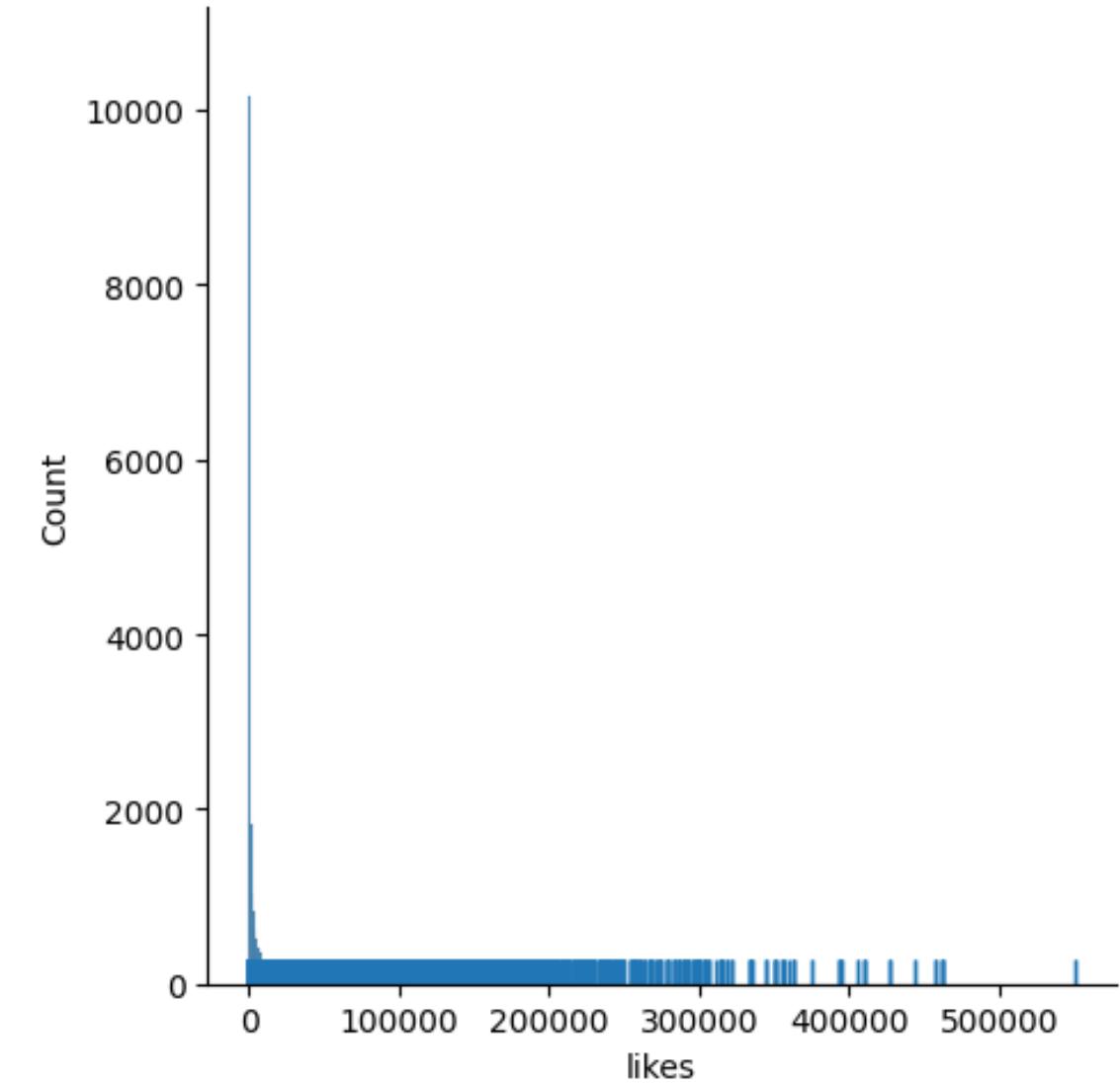
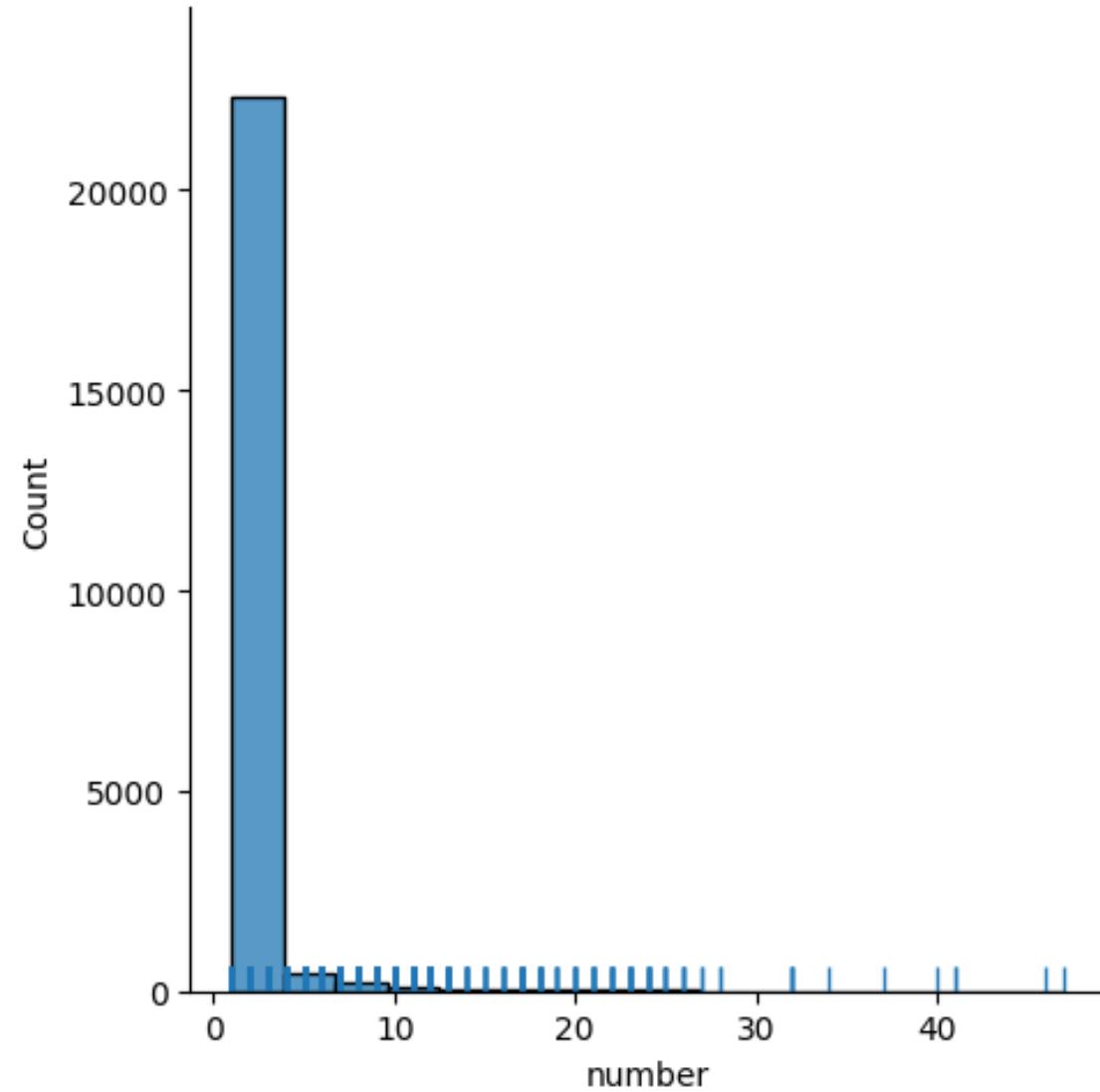
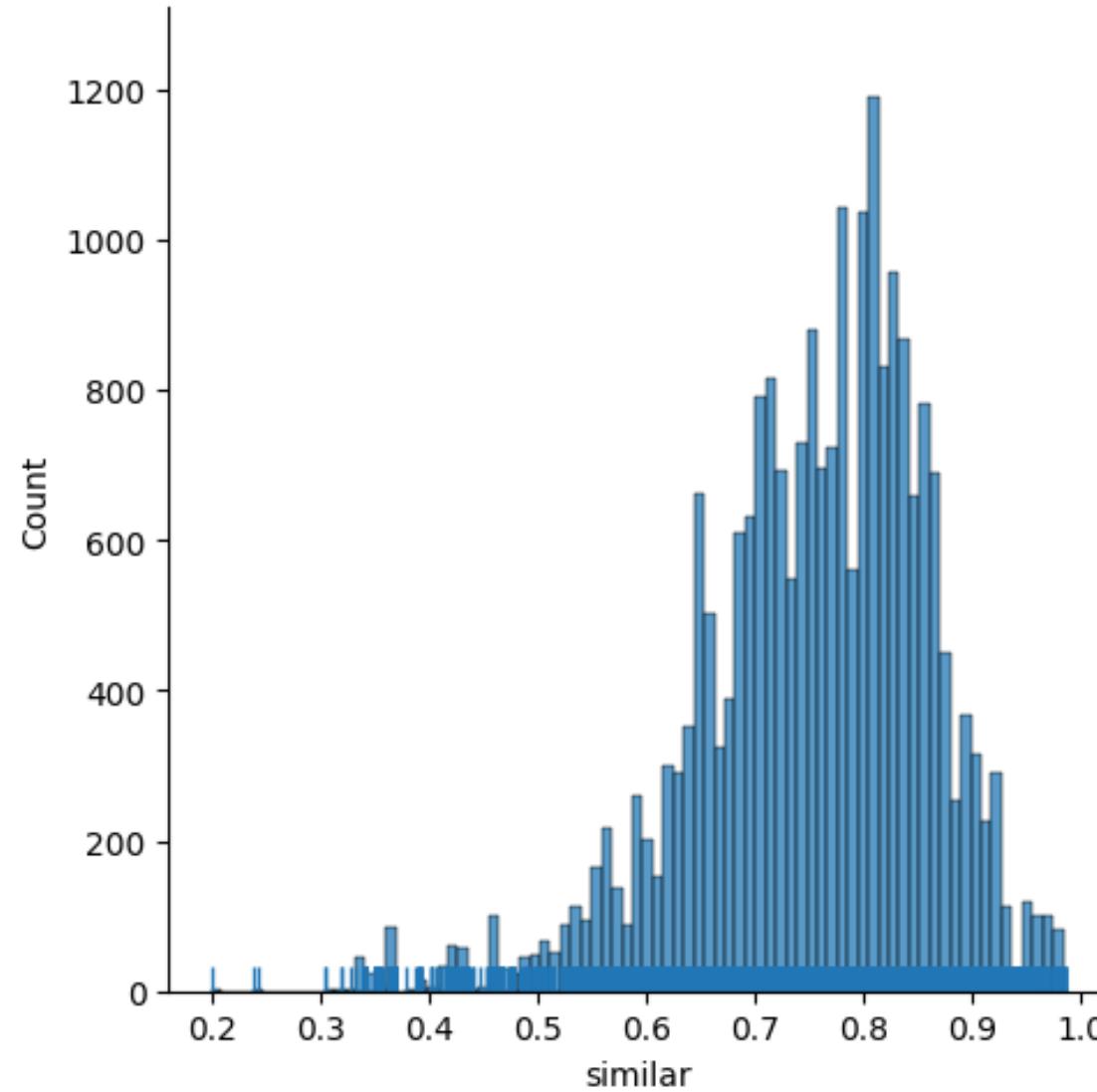
```
dictionary_group = {'sunny': sunny_dict, 'foggy': foggy_dict, 'rainy': rainy_dict, \  
                   'snowy': snowy_dict, 'sunrise_sunset': sunrise_sunset_dict}  
  
similar = []  
for i in range(len(df)):  
    label = df['label'][i]  
    dictionary = dictionary_group[label]  
    small = 0  
    for j in df['specific_tag'][i]:  
        # 토큰 리스트에 없는 단어 존재. 값을 0으로 지정  
        if j not in no_duplicate:  
            continue  
        else:  
            small += dictionary[j]  
    # 평균값 구하기  
    small = small / len(df['specific_tag'][i])  
    similar.append(small)  
  
df['similar'] = similar
```

레이블에 해당하는 딕셔너리에서 specific\_tag마다의 유사도를 다 더한 후,  
노래 1개마다의 tag 개수로 나눠 평균 점수 도출

['similar']이라는 이름으로 변수 지정



# 연관성 지표 Scaling



number와 likes가 similar처럼 모두 0과 1 사이의 값이 되도록 MinMaxScaler를 사용





# 05

# 추천 방식

```
df['final'] = df['number'] * 4.5 + df['likes'] * 1 + df['similar'] * 4.5
df[df['label']=='snowy'].sort_values(by='final', ascending=False).head(10)
```

	title	artist	specific_tag	label	number	likes	similar	final
650	All I Want for Christmas Is You	Mariah Carey	[감성, 크리스마스, 추운날씨, 눈오는 날, 겨울, 크리스마스캐롤, 겨울노래, pub...]	snowy	0.717391	0.395711	0.731114	6.913984
11129	Santa Tell Me	Ariana Grande	[감성, 크리스마스, 추운날씨, 눈오는 날, 겨울, 크리스마스캐롤, 겨울노래, pub...]	snowy	0.586957	0.422504	0.736969	6.380167
22402	첫 눈	EXO	[감성, 크리스마스, 추운날씨, 눈오는 날, 겨울, 탑백, TOP100, 인기, 차트...]	snowy	0.543478	0.384723	0.726388	6.099123
11738	Snowman	Sia	[감성, 크리스마스, 추운날씨, 눈오는 날, 겨울, 크리스마스캐롤, 겨울노래, pub...]	snowy	0.500000	0.450776	0.751866	6.084171
18806	미리 메리 크리스마스 (Feat. 천둥 Of MBLAQ)	아이유	[감성, 크리스마스, 추운날씨, 눈오는 날, 겨울, 탑백, TOP100, 인기, 차트...]	snowy	0.521739	0.310736	0.751019	6.038147
13810	Underneath The Tree	Kelly Clarkson	[감성, 크리스마스, 추운날씨, 눈오는 날, 겨울, 크리스마스캐롤, 겨울노래, pub...]	snowy	0.521739	0.090465	0.736463	5.752377
7145	Last Christmas	Ariana Grande	[감성, 크리스마스, 추운날씨, 눈오는 날, 겨울, 팝송, 포근한, 캐롤, 행복한, ...]	snowy	0.413043	0.109332	0.796661	5.553004
17897	눈 (Feat. 이문세)	Zion.T	[겨울, 눈오는날, 따뜻한, 발라드, 잔잔 한, 추억, 추위, 연말, 포근한, 일상, ...]	snowy	0.478261	0.226585	0.700026	5.528877
9499	Oh Santa! (Feat. Ariana Grande, Jennifer Hudson)	Mariah Carey	[크리스마스, 크리스마스캐롤, 겨울노래, pub, 겨울감성, 추위, 따뜻한, 찬바람...]	snowy	0.369565	0.032409	0.838463	5.468535
22588	성시경, 박효신, 이석훈, 서인국, VIXX (박스)	크리스마스니까	[눈오는날, 탑백, TOP100, 인기, 차트, 눈, 겨울, 첫눈, 눈_오는_날, ...]	snowy	0.369565	0.294894	0.748814	5.327598

=> likes : number : similar = 1 : 4.5 : 4.5

가중치를 지정해 최종 점수(final) 도출

```
def recommend(input):
    df_input = df[df['label']==input]
    df_input.sort_values(by='final', ascending=False, inplace=True)
    df_input = df_input[:len(df_input)//20] # 상위 5% 추출
    output = df_input.sample(n=10, replace=False)

    return output
```

recommend('sunny')

	title	artist	specific_tag	label	number	likes	file_name	image	similar	final
2965	Cupid (Feat. PENOMEKO)	pH-1	[드라이브, 맑은날, 맑은날씨, 화창한날씨, 아침, 시작, 출근, 퇴근, 여행, 내...]	sunny	0.021739	0.097553	Cupid (Feat. PENOMEKO).pH-1	<a href="https://cdnimg.melon.co.kr/cm/album/images/101...">https://cdnimg.melon.co.kr/cm/album/images/101...</a>	0.919450	4.332902
22329	차와 이야기를 나눠요	레피 (repi)	[파이노, 초여름, 뉴에이지피아노, 여름, 늦봄, 카페, 휴식, 감성, 맑은날, 아...]	sunny	0.543478	0.000036	차와 이야기를 나눠요.레피 (repi)	<a href="https://cdnimg.melon.co.kr/cm2/album/images/10...">https://cdnimg.melon.co.kr/cm2/album/images/10...</a>	0.747428	5.809117
22114	조깅	LUCY	[드라이브, 맑은날, 맑은날씨, 화창한날씨, 아침, 시작, 출근, 퇴근, 여행, 내...]	sunny	0.065217	0.058409	조깅_LUCY	<a href="https://cdnimg.melon.co.kr/cm2/album/images/10...">https://cdnimg.melon.co.kr/cm2/album/images/10...</a>	0.872326	4.277356
8265	Maniac	Conan Gray	[드라이브, 기분전환, 출퇴근길, 여행, 산책, 휴식, 힐링, 스트레스, 운동, 맑...]	sunny	0.152174	0.302861	Maniac_Conan Gray	<a href="https://cdnimg.melon.co.kr/cm2/album/images/10...">https://cdnimg.melon.co.kr/cm2/album/images/10...</a>	0.750274	4.363878
22574	퀸카 (Queencard)	(여자)아이들	[아이돌, 여름, 기분전환, 더위, 여행, 드라이브, 신나는, 청량감, 트로피컬, ...]	sunny	0.021739	0.238871	퀸카 (Queencard)_여자아이들	<a href="https://cdnimg.melon.co.kr/cm2/album/images/11...">https://cdnimg.melon.co.kr/cm2/album/images/11...</a>	0.872528	4.263072
3663	ETA	NewJeans	[아이돌, 여름, 기분전환, 더위, 여행, 드라이브, 신나는, 청량감, 트로피컬, ...]	sunny	0.043478	0.165113	ETA_NewJeans	<a href="https://cdnimg.melon.co.kr/cm2/album/images/11...">https://cdnimg.melon.co.kr/cm2/album/images/11...</a>	0.872528	4.287140
23116	화용월태(花容月態)	도화	[뉴에이지, 사무실, 월요병, 연주곡, 배경음악, 공부할때, 화창한날, 북카페, 커...]	sunny	0.217391	0.000013	화용월태(花容月態)_도화	<a href="https://cdnimg.melon.co.kr/cm2/album/images/10...">https://cdnimg.melon.co.kr/cm2/album/images/10...</a>	0.729081	4.259137
20926	에잇(Prod.&Feat. SUGA of BTS)	아이유	[방탄소년단, 최신노래, 맑은날, 월요병, 봄, 사랑, 아이돌, 아이유, 신나는, ...]	sunny	0.065217	0.716451	에잇(Prod.&Feat. SUGA of BTS)_아이유	<a href="https://cdnimg.melon.co.kr/cm2/album/images/10...">https://cdnimg.melon.co.kr/cm2/album/images/10...</a>	0.793126	4.578997
4466	Freak (Prod. Slom)	릴보이 (IIIBOI), 원슈타인, Chillin Homie, 스카이민혁 (Skym...)	[맑은날, 봄날, 따스한, 습은명곡]	sunny	0.000000	0.141145	Freak (Prod. Slom).릴보이 (IIIBOI), 원슈타인, Chillin...	<a href="https://cdnimg.melon.co.kr/cm2/album/images/10...">https://cdnimg.melon.co.kr/cm2/album/images/10...</a>	0.918976	4.276538
20584	아무노래	지코 (ZICO)	[여행, 드라이브, 신나는, 불률업, 댄스, 도입부맛집, 흥폭발]	sunny	0.000000	0.506208	아무노래_지코 (ZICO)	<a href="https://cdnimg.melon.co.kr/cm2/album/images/10...">https://cdnimg.melon.co.kr/cm2/album/images/10...</a>	0.867799	4.411304

final 변수를 기준으로 내림차순 정렬 후 상위 5% 내에서 랜덤으로 10개 추출





# 06

# 날씨 이미지 분류 - CNN

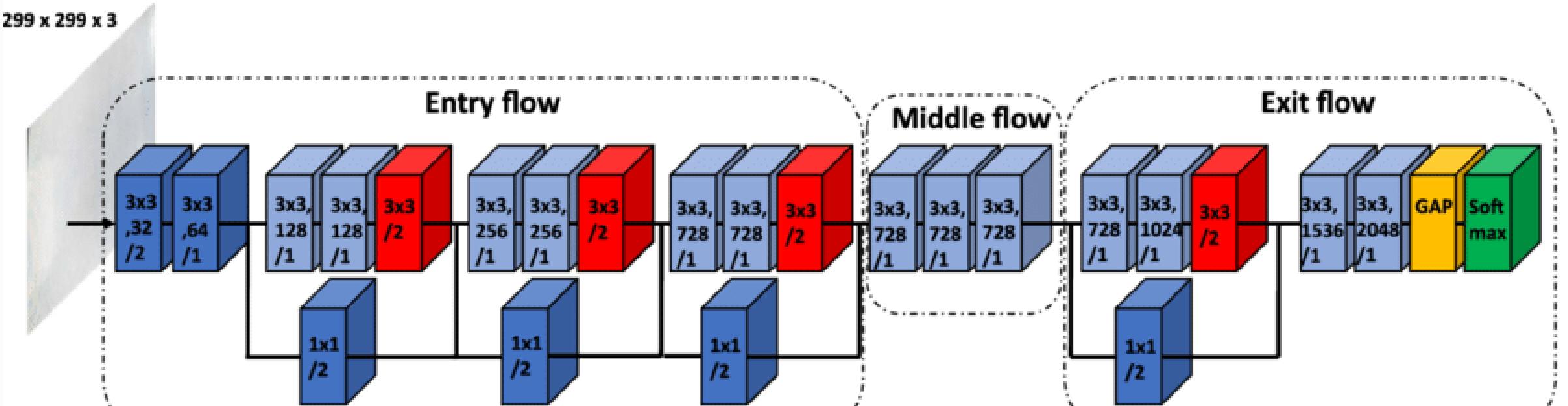
## Frameworks

K Keras TF TensorFlow

ResNet50  
VGG16  
**Xception**

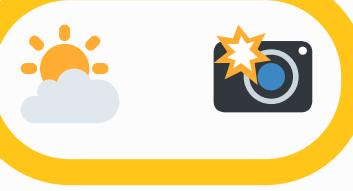
ResNet101  
VGG19  
InceptionV3

Model



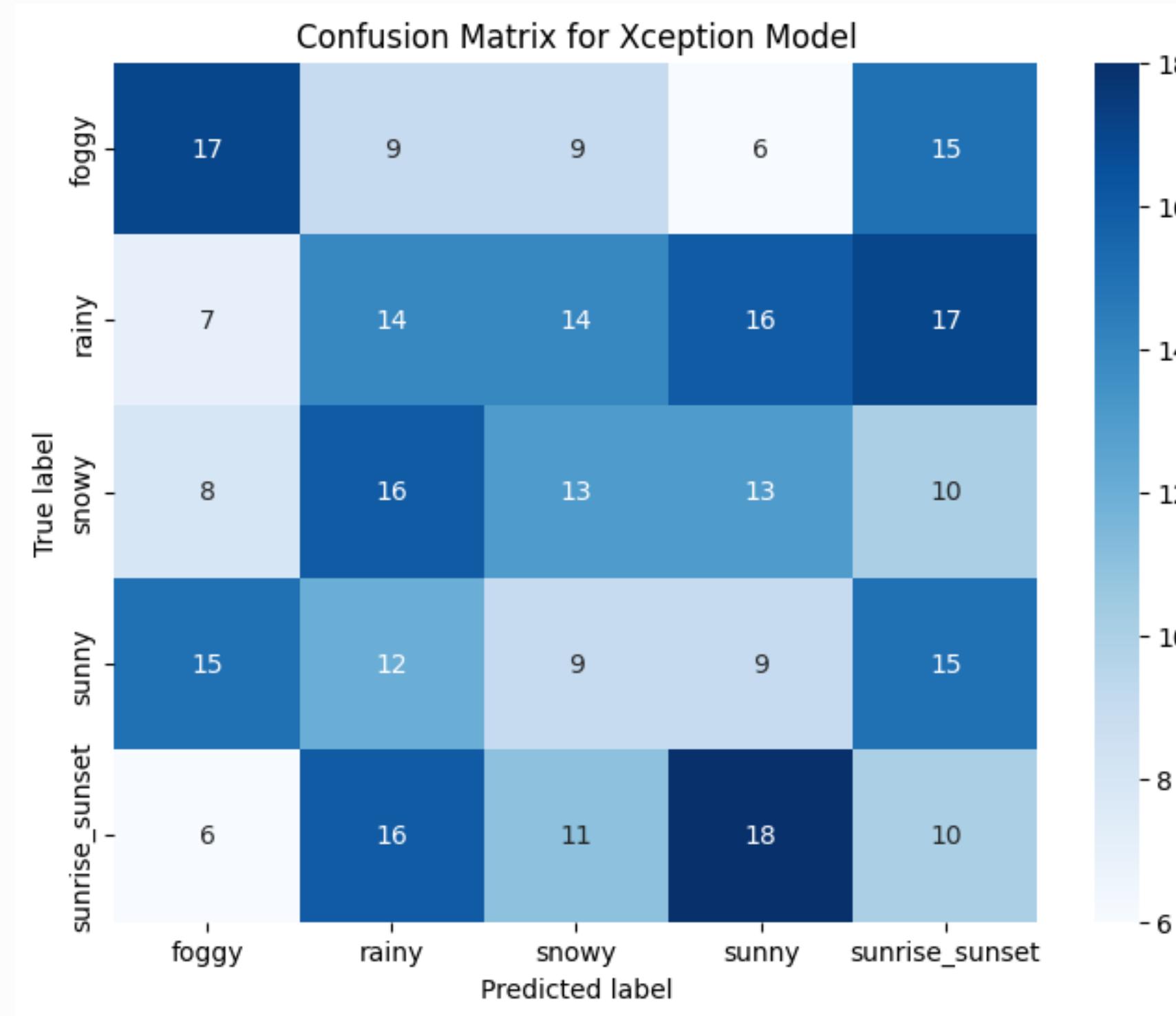
foggy  
rainy  
**snowy**  
sunny  
inrise\_sunset





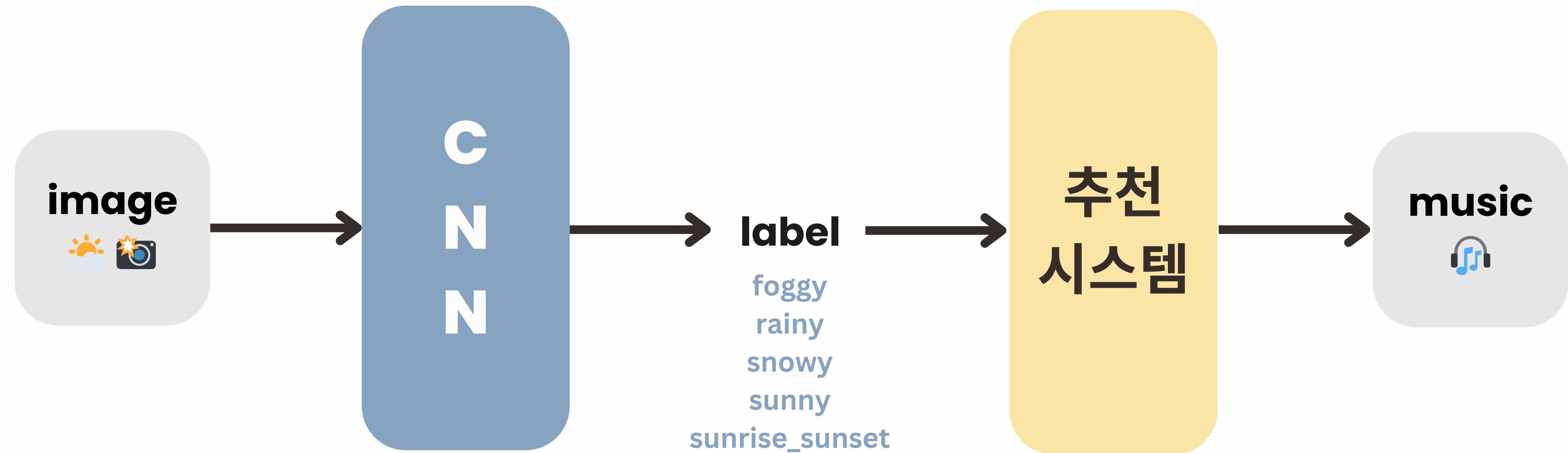
06

# 날씨 이미지 분류 - CNN





# 전체 모델 구조

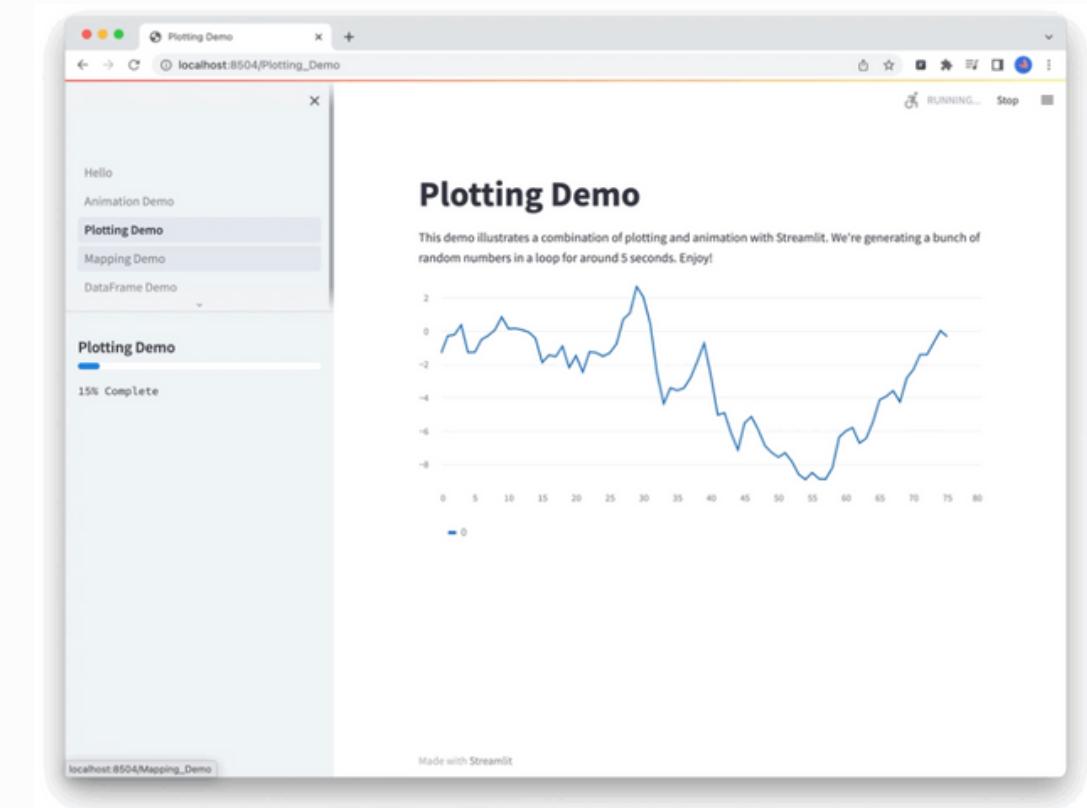
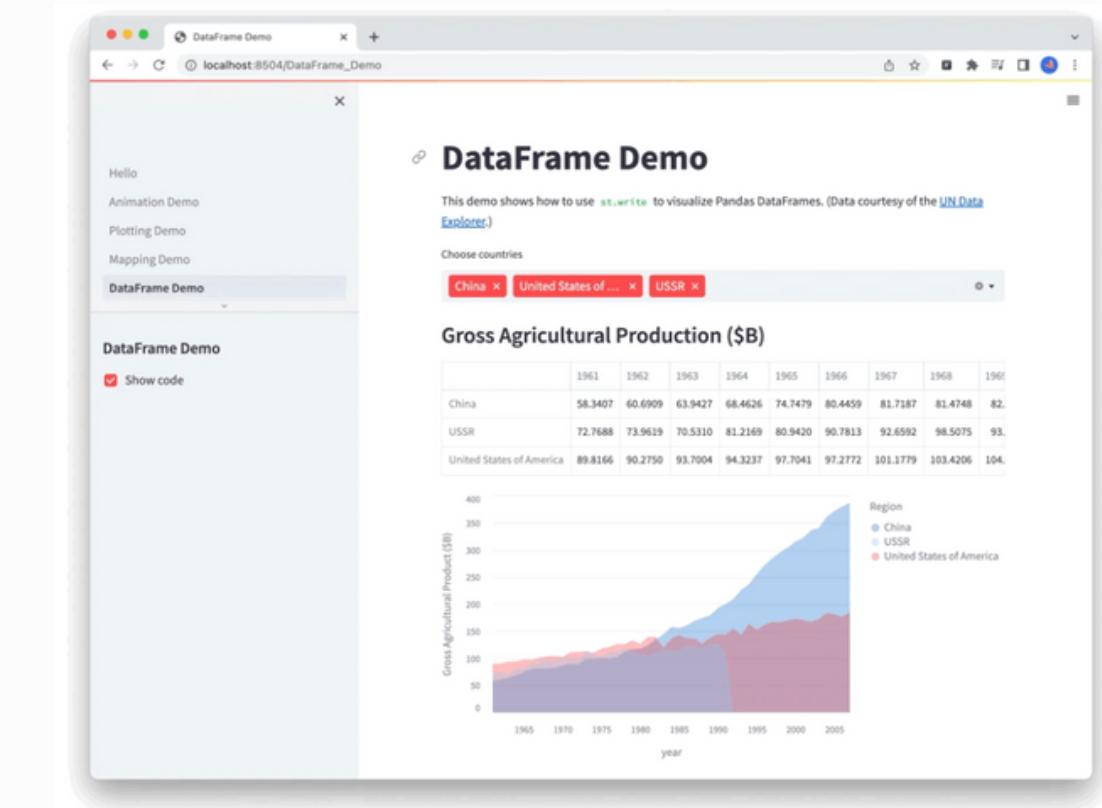
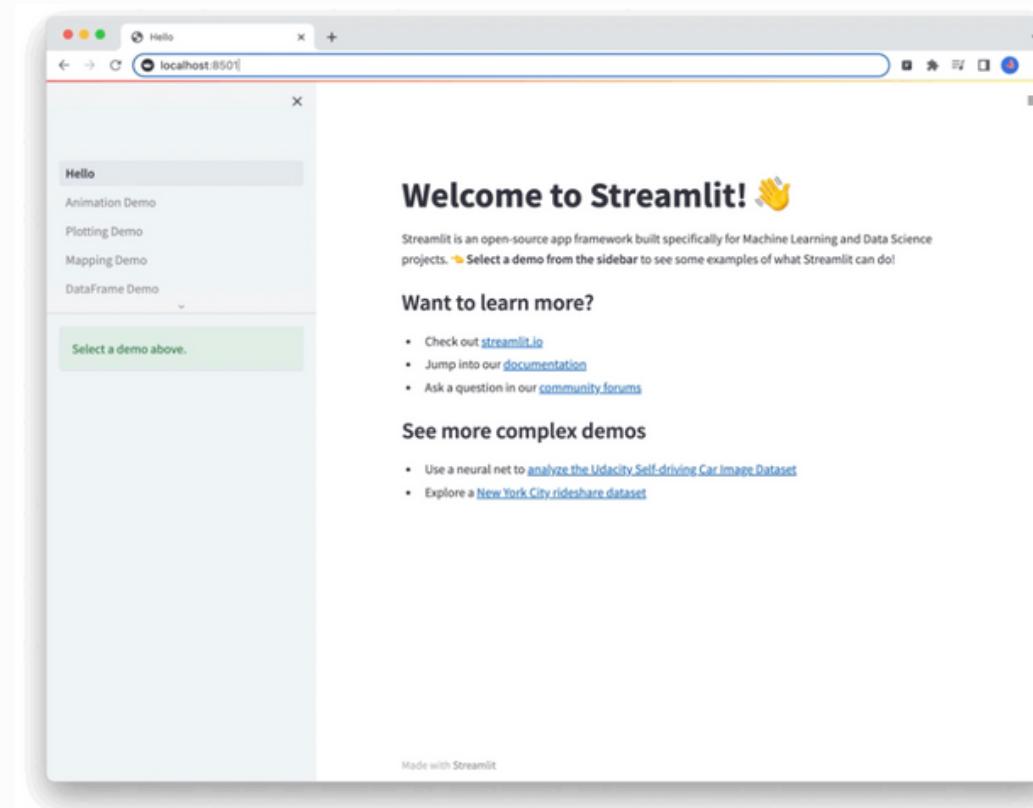




Streamlit

# Streamlit

: 데이터 과학 및 기계 학습 프로젝트를 위해  
개발자가 대화형 웹 응용 프로그램을 신속하게  
만들 수 있는 오픈 소스 파이썬 라이브러리



07

# 전체 모델 구현



Streamlit



## Home

The screenshot shows a Streamlit application window titled "Home". On the left, there's a sidebar with a "Home" tab and a "Upload Your Picture" button. The main content area features a large graphic of a yellow sun and blue clouds with raindrops, overlaid with the text "Weather Can \"Pic\" Your Music" and "직접 찍은 날씨 사진에 어울리는 노래 추천 시스템". Below this is a control bar with a play/pause button and navigation arrows. At the bottom, there's a yellow banner with the text "Weather Can \"Pic\" Your Music" and "에 오신 것을 환영합니다! ☀️📸🎧". A small note at the very bottom says "본인이 직접 찍은 사진에 어울리는 음악을 추천해드립니다."



07

# 전체 모델 구현



Streamlit



## 📸 Upload Your Picture

The screenshot shows a Streamlit application window titled "Upload Your Picture". The window has a "Home" button on the left. In the center, there is a message in Korean: "📸 이미지를 업로드해주시길 바랍니다!" (Please upload your image). Below this is a file upload interface with a "Drag and drop file here" placeholder and a "Browse files" button. A note specifies "Limit 200MB per file • PNG, JPG, JPEG". At the bottom of the window, it says "Made with Streamlit". The window is running on "localhost" and is the active tab in a browser window.



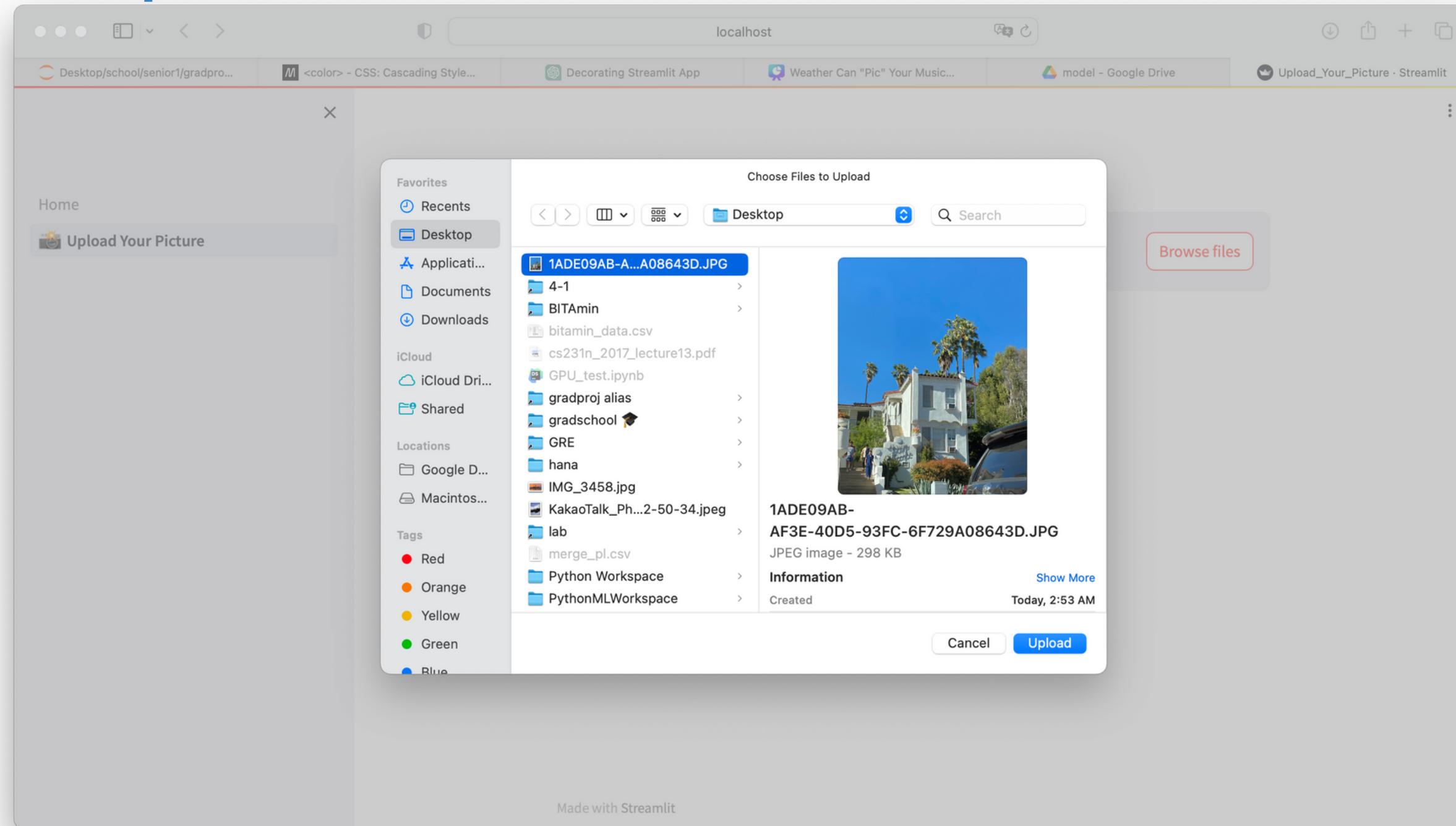
07

# 전체 모델 구현



Streamlit

## 📸 Upload Your Picture



# 07

# 전체 모델 구현



# Streamlit



## 📸 Upload Your Picture

CNN이 분류한 날씨 label

추천 시스템을 거쳐  
선별된 노래 10곡

앨범 이미지  
노래 제목  
가수 이름  
specific tag  
youtube video



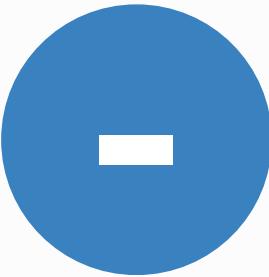


08

## 의의 / 한계점 및 발전 가능성



- 음악 플랫폼의 실제 데이터를 활용하여 추천하는 시스템
- 실생활에 사용이 가능한 시스템
- CNN + 추천 시스템을 둘 다 활용



- 커스텀 데이터셋 → 데이터 양 ↓
- 날씨 **label** 5개로 한정 = 다양한 날씨 고려 X
  - 데이터셋 크기를 늘려 CNN 성능 ↑
- 현재는 날씨 사진만 보고 추천을 해줌
  - 실내에서 사진을 찍어도 추천해주는 시스템 구축



# Thank You For Listening

