



14주차 발표

DS팀 황선경 김경민 김도하

목차

#01 문서 군집화

#02 문서 유사도

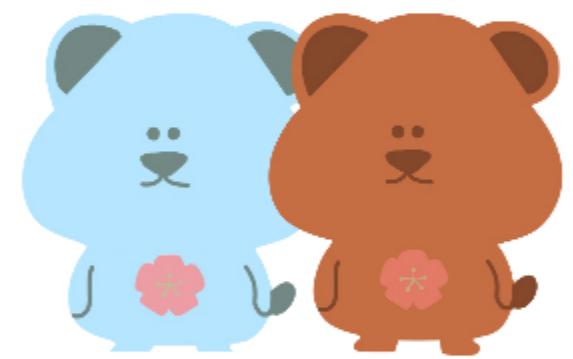
#03 한글 텍스트 데이터 처리

#04 캐글 실습 : 대형 온라인 쇼핑몰 제품 가격 예측

#05 추가 노트북



01.문서 군집화



1.1 문서 군집화(Document Clustering)

문서 군집화

: 비슷한 텍스트 구성의 문서를 군집화 하는 것

문서 군집화	텍스트 분류 기반의 문서 분류
동일한 군집에 속하는 문서를 같은 카테고리 소속으로 분류	
학습 데이터 세트가 필요 없는 비지도 학습 기반으로 동작	사전에 결정 카테고리 값을 가진 학습 데이터 세트가 필요

분석 흐름

텍스트 전처리 → 벡터화 → 군집화 알고리즘 적용 → cluster_centers(추출) 통해 군집별 핵심 단어 추출

1.2 Opinion Review Data 실습



Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Check out the [beta version](#) of the new UCI Machine Learning Repository we are currently testing
[try out the new site](#).

Opinosis Opinion / Review Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: This dataset contains sentences extracted from user reviews on a given topic. Example topics are nano.

Data Set Characteristics:	Text	Number of Instances:	51	Area:	Computer
Attribute Characteristics:	N/A	Number of Attributes:	N/A	Date Donated	2010-07-06
Associated Tasks:	N/A	Missing Values?	N/A	Number of Web Hits:	56789

51개의 텍스트 파일로 구성

Tripadvisor(호텔), Edmunds.com(자동차), Amazon.com(전자제품) 사이트에서 가져온 리뷰 문서

1.2 Opinion Review Data 실습 – 텍스트 전처리

```
import pandas as pd
import glob, os
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_colwidth', 700)

path = r'/Users/dohakim/Downloads/OpinosisDataset1.0/topics'
# path로 지정한 디렉터리 밑에 있는 모든 .data 파일들의 파일명을 리스트로 취합
all_files = glob.glob(os.path.join(path, "*.data"))
filename_list = []
opinion_text = []

# 개별 파일들의 파일명은 filename_list 리스트로 취합,
# 개별 파일들의 파일 내용은 DataFrame 로딩 후 다시 string으로 변환하여 opinion_text 리스트로 취합
for file_ in all_files:
    # 개별 파일을 읽어서 DataFrame으로 생성
    df = pd.read_table(file_, index_col=None, header=0, encoding='latin1')

    # 절대경로로 주어진 file 명을 가공. 만일 Linux에서 수행시에는 아래 \\를 / 변경.
    # 맨 마지막 .data 확장자도 제거
    filename_ = file_.split('/')[-1]
    filename = filename_.split('.')[0]

    # 파일명 리스트와 파일 내용 리스트에 파일명과 파일 내용을 추가.
    filename_list.append(filename)
    opinion_text.append(df.to_string())

# 파일명 리스트와 파일 내용 리스트를 DataFrame으로 생성
document_df = pd.DataFrame({'filename':filename_list, 'opinion_text':opinion_text})
document_df.head()
```

	filename	opinion_text
0	battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb.\n0 I love this ipod except for the battery life.\n1 ...
1	gas_mileage_toyota_camry_2007	Ride seems comfortable and gas mileage fairly good averaging 26 city and 30 open road.\n0 ...
2	room_holiday_inn_london	We arrived at 23,30 hours and they could not recommend a restaurant so we decided to go to Tesco, with very limited choices but when you are hingry you do not careNext day they rang the bell at 8,00 hours to clean the room, not being very nice being waken up so earlyEvery day they gave u...
3	location_holiday_inn_london	Great location for tube and we crammed in a fair amount of sightseeing in a short time.\n0 All in all, a normal chain hotel on a nice lo...
4	staff_bestwestern_hotel_sfo	Staff are friendl...

텍스트 전처리

for문을 통해 여러 개의 파일을 DataFrame으로 로딩

각 파일 이름 자체만으로 의견의 텍스트가 어떤 제품/서비스에 관한 리뷰인지 알 수 있음

1.2 Opinion Review Data 실습 – 벡터화

```
# pip install nltk

from nltk.stem import WordNetLemmatizer
import nltk
# nltk.download('punkt')
# nltk.download('wordnet')
#nltk.download()
import string

remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
lemmar = WordNetLemmatizer()

# 입력으로 들어온 token단어들에 대해서 lemmatization 어근 변환.
def LemTokens(tokens):
    return [lemmar.lemmatize(token) for token in tokens]

# TfIdfVectorizer 객체 생성 시 tokenizer인자로 해당 함수를 설정하여 lemmatization 적용
# 입력으로 문장을 받아서 stop words 제거-> 소문자 변환 -> 단어 토큰화 -> lemmatization 어근 변환.
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vect = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english', \
                             ngram_range=(1,2), min_df=0.05, max_df=0.85)

#opinion_text 컬럼값으로 feature vectorization 수행
feature_vect = tfidf_vect.fit_transform(document_df['opinion_text'])
```

TF-IDF 기반 Vectorization 적용 및 KMeans 군집화 수행

- : TfidfVectorizer의 tokenizer인자로 사용될 lemmatization 어근 변환 함수를 설정
 - : Stemming과 Lemmatization 같은 어근 변환은 TfidfVectorizer에서 직접 지원하진 않으나 tokenizer 파라미터에 커스텀 어근 변환 함수를 적용하여 어근 변환을 수행할 수 있음
 - : TfidfVectorizer 생성자의 tokenizer인자로 위에서 생성 LemNormalize 함수 설정
-
- min_df, max_df를 통해 피처의 개수 제한
 - TfidfVectorizer의 fit_transform 인자를 통해 개별 문서 텍스트에 대해 TF-IDF 변환된 피처 벡터화된 행렬 구함

1.2 Opinion Review Data 실습 – 군집화 알고리즘 적용

```
from sklearn.cluster import KMeans  
  
# 5개 집합으로 군집화 수행. 예제를 위해 동일한 클러스터링 결과 도출용 random_state=0  
km_cluster = KMeans(n_clusters=5, max_iter=10000, random_state=0)  
km_cluster.fit(feature_vect)  
cluster_label = km_cluster.labels_  
cluster_centers = km_cluster.cluster_centers_  
  
document_df['cluster_label'] = cluster_label  
document_df.head()
```

filename	opinion_text	cluster_label
0 battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb .\n0 I love this ipod except for the battery life .\n1 ...	3
1 gas_mileage_toyota_camry_2007	Ride seems comfortable and gas mileage fairly good averaging 26 city and 30 open road .\n0 ...	2
2 room_holiday_inn_london	We arrived at 23,30 hours and they could not recommend a restaurant so we decided to go to Tesco, with very limited choices but when you are hingry you do not careNext day they rang the bell at 8,00 hours to clean the room, not being very nice being waken up so earlyEvery day they gave u...	0
3 location_holiday_inn_london	Great location for tube and we crammed in a fair amount of sightseeing in a short time .\n0 All in all, a normal chain hotel on a nice lo...	0
4 staff_bestwestern_hotel_sfo	Staff are friendl...	0

군집화 알고리즘 적용

- : 군집이 각 주제별로 유사한 형태로 잘 구성됐는지
- : 문서별 텍스트가 TF-IDF 변환된 피처 벡터와 행렬 데이터에 대해 군집화를 수행해 어떤 문서끼리 군집되는지 확인

군집화 기법 : K-Means

각 데이터별로 할당된 군집의 레이블을 파일명과 파일 내용을 갖고 있는 document_df에 ‘cluster_label’ 칼럼 추가

1.2 Opinion Review Data 실습 – 군집화 알고리즘 적용

```
document_df[document_df['cluster_label']==0].sort_values(by='filename')
```

filename	opinion_text	cluster_label
31 bathroom_bestwestern_hotel_sfo	The room was not overly big, but clean and very comfortable beds, a great shower and very clean bathrooms .\n0 The second room was smaller, with a very inconvenient bathroom layout, but at least it was quieter and we were able to sleep .\n1 ...	0
17 food_holiday_inn_london	The room was packed to capacity with queues at the food buffets .\n0 The over zealous staff cleared our unfinished drinks while we were collecting cooked food and movement around the room with plates was difficult in the crowded circumstances .\n1 ...	0
32 food_swissotel_chicago	The food for our event was delicious .\n0 ...	0
49 free_bestwestern_hotel_sfo	The wine reception is a great idea as it is nice to meet other travellers and great having access to the free Internet access in our room .\n0 They also have a computer available with free internet which is a nice bonus but I didn't find that out till the day before we left but was still able to get on there to check our flight to Vegas the next day .\n1 ...	0
39 location_bestwestern_hotel_sfo	Good Value good location , ideal choice .\n0 Great Location , Nice Rooms , Helpful Concierge!\n1 ...	0
3 location_holiday_inn_london	Great location for tube and we crammed in a fair amount of sightseeing in a short time .\n0 All in all, a normal chain hotel on a nice lo...	0
50 parking_bestwestern_hotel_sfo	Parking was expensive but I think this is common for San Fran .\n0 there is a fee for parking but well worth it seeing no where to park if you do have a car .\n1 ...	0
28 price_holiday_inn_london	All in all, a normal chain hotel on a nice location , I will be back if I do not find anything closer to Picadilly for a better price .\n0 ...	0
2 room_holiday_inn_london	We arrived at 23,30 hours and they could not recommend a restaurant so we decided to go to Tesco, with very limited choices but when you are hungry you do not careNext day they rang the bell at 8,00 hours to clean the room, not being very nice being waken up so earlyEvery day they gave u...	0
46 rooms_bestwestern_hotel_sfo	Great Location , Nice Rooms , H...	0
30 rooms_swissotel_chicago	The Swissotel is one of our favorite hotels in Chicago and the corner rooms have the most fantastic views in the city .\n0 The rooms look like they were just remodeled and upgraded, there was an HD TV and a nice iHome docking station to put my iPod so I could set the alarm to wake up with my music instead of the radio .\n1 ...	0
16 service_bestwestern_hotel_sfo	Both of us having worked in tourism for over 14 years were very disappointed at the level of service provided by this gentleman .\n0 The service was good, very friendly staff and we loved the free wine reception each night .\n1 ...	0
27 service_holiday_inn_london	not customer, oriented hotelvery low service levelboor reception\n0 The room was quiet, clean, the bed and pillows were comfortable, and the serv...	0
13 service_swissotel_hotel_chicago	Mediocre room and service for a very extravagant price .\n0 ...	0
4 staff_bestwestern_hotel_sfo	Staff are friendl...	0
20 staff_swissotel_chicago	The staff at Swissotel were not particularly nice .\n0 Each time I waited at the counter for staff for several minutes and then was waved to the desk upon my turn with no hello or anything, or apology for waiting in line .\n1 ...	0

cluster #0
호텔에 관한 리뷰

```
document_df[document_df['cluster_label']==1].sort_values(by='filename')
```

filename	opinion_text	cluster_label
33 accuracy_garmin_nuvi_255W_gps	, and is very, very accurate .\n0 but for the most part, we find that the Garmin software provides accurate directions, wherever we intend to go .\n1 This functi...	1
34 directions_garmin_nuvi_255W_gps	You also get upscale features like spoken directions including street names and programmable POIs .\n0 I used to hesitate to go out of my directions but no...	1
48 display_garmin_nuvi_255W_gps	3 quot widescreen display was a bonus .\n0 This made for smoother graphics on the 255w of the vehicle moving along displayed roads, where the 750's display was more of a jerky movement .\n1 ...	1
21 features_windows7	I had to uninstall anti, virus and selected other programs, some of which did not have listings in the Programs and Features Control Panel section .\n0 This review briefly touches upon some of the key features and enhancements of Microsoft's latest OS .\n1 ...	1
12 keyboard_netbook_1005ha	, I think the new keyboard rivals the great hp mini keyboards .\n0 Since the battery life difference is minimum, the only reason to upgrade would be to get the better keyboard .\n1 The keyboard is now as good as t...	1
10 satellite_garmin_nuvi_255W_gps	It's fast to acquire satellites .\n0 If you've ever had a Brand X GPS take you on some strange route that adds 20 minutes to your trip, has you turn the wrong way down a one way road, tell you to turn AFTER you've passed the street, frequently loses the satellite signal, or has old maps missing streets, you know how important this stuff is .\n1 ...	1
8 screen_garmin_nuvi_255W_gps	It is easy to read and when touching the screen it works great !\n0 and zoom out buttons on the 255w to the same side of the screen which makes it a bit easier .\n1 ...	1
25 screen_ipod_nano_8gb	As always, the video screen is sharp and bright .\n0 2, inch screen and a glossy, polished aluminum finish that one CNET editor described as looking like a Christmas tree ornament .\n1 ...	1
37 screen_netbook_1005ha	Keep in mind that once you get in a room full of light or step outdoors screen reflections could become annoying .\n0 I've used mine outside...	1
7 size_asus_netbook_1005ha	A few other things I'd like to point out is that you must push the micro, sized right angle end of the ac adapter until it snaps in place or the battery may not charge .\n0 The full size right shift k...	1
6 speed_garmin_nuvi_255W_gps	Another feature on the 255w is a display of the posted speed limit on the road which you are currently on right above your current displayed speed .\n0 I found myself not even looking at my car speedometer as I could easily see my current speed and the speed limit of my route at a glance .\n1 ...	1
40 speed_windows7	Windows 7 is quite simply faster, more stable, boots faster, goes to sleep faster, comes back from sleep faster, manages your files better and on top of that it's beautiful to look at and easy to use .\n0 , faster about 20% to 30% faster at running applications than my Vista , seriously\n1 ...	1
19 updates_garmin_nuvi_255W_gps	Another thing to consider was that I paid \$50 less for the 750 and it came with the FM transmitter cable and a USB cord to connect it to your computer for updates and downloads .\n0 update and reroute much _more_ quickly than my other GPS .\n1 UPDATE ON THIS , It finally turned out that to see the elevation contours at lowe...	1
14 video_ipod_nano_8gb	I bought the 8, gig Ipod Nano that has the built, in video camera .\n0 Itunes has an on, line store, where you may purchase and download music and videos which will install onto the ipod .\n1 ...	1
5 voice_garmin_nuvi_255W_gps	The voice prompts and maps are wonderful especially when driving after dark .\n0 I also thought the the voice prompts of the 750 where more pleasant sounding than the 255w's .\n1 ...	1

cluster #1
킨들, 아이팟, 넥북 등의 포터블 전자기기 및 주요 구성요소(배터리, 키보드 등)에 대한 리뷰

1.2 Opinion Review Data 실습 – 군집화 알고리즘 적용

```
document_df[document_df['cluster_label']==2].sort_values(by='filename')
```

	filename	opinion_text	cluster_label
18	comfort_honda_accord_2008	Drivers seat not comfortable, the car itself compared to other models of similar class .\n0 ...	2
43	comfort_toyota_camry_2007	Ride seems comfortable and gas mileage fairly good averaging 26 city and 30 open road .\n0 Seats are fine, in fact of all the smaller sedans this is the most comfortable I found for the price as I am 6', 2 and 250#. \n1 Great gas mileage and comfortable on long trips ...	2
1	gas_mileage_toyota_camry_2007	Ride seems comfortable and gas mileage fairly good averaging 26 city and 30 open road .\n0 ...	2
45	interior_honda_accord_2008	I love the new body style and the interior is a simple pleasure except for the center dash .\n0 ...	2
22	interior_toyota_camry_2007	First of all, the interior has way too many cheap plastic parts like the cheap plastic center piece that houses the clock .\n0 3 blown struts at 30,000 miles, interior trim coming loose and rattling squeaking, stains on paint, and bug splats taking paint off, premature uneven brake wear, on 3rd windsh...	2
35	mileage_honda_accord_2008	It's quiet, get good gas mileage and looks clean inside and out .\n0 The mileage is great, and I've had to get used to stopping less for gas .\n1 Thought gas ...	2
47	performance_honda_accord_2008	Very happy with my 08 Accord, performance is quite adequate it has nice looks and is a great long, distance cruiser .\n0 6, 4, 3 eco engine has poor performance and gas mileage of 22 highway .\n1 Overall performance is good but comfort level is poor .\n2 ...	2
42	quality_toyota_camry_2007	I previously owned a Toyota 4Runner which had incredible build quality and reliability .\n0 I bought the Camry because of Toyota reliability and qua...	2
29	seats_honda_accord_2008	Front seats are very uncomfortable .\n0 No memory seats, no trip computer, can only display outside temp with trip odometer .\n1 ...	2
23	transmission_toyota_camry_2007	After slowing down, transmission has to be kicked to speed up .\n0 ...	2

cluster #2

cluster #1과 비슷하게 킨들 등이 군집에 포함,
주로 차량용 내비게이션으로 군집 구성

```
document_df[document_df['cluster_label']==3].sort_values(by='filename')
```

	filename	opinion_text	cluster_label
9	battery-life_amazon_kindle	After I plugged it in to my USB hub on my computer to charge the battery the charging cord design is very clever !\n0 After you have paged tru a 500, page book one, page, at, a, time to get from Chapter 2 to Chapter 15, see how excited you are about a low battery and all the time it took to get there !\n1 ...	3
0	battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb .\n0 I love this ipod except for the battery life .\n1 ...	3
11	battery-life_netbook_1005ha	6GHz 533FSB cpu, glossy display, 3, Cell 23Wh Li, ion Battery , and a 1 .\n0 Not to mention that as of now...	3
15	performance_netbook_1005ha	The Eee Super Hybrid Engine utility lets users overclock or underclock their Eee PC's to boost performance or provide better battery life depending on their immediate requirements .\n0 In Super Performance mode CPU, Z shows the bus speed to increase up to 169 .\n1 One...	3
24	sound_ipod_nano_8gb	headphone jack i got a clear case for it and it i got a clear case for it and it like prvents me from being able to put the jack all the way in so the sound can b messsed up or i can get it in there and its playing well them go to move or something and it slides out .\n0 Picture and sound quality are excellent for this typ of devic .\n1 ...	3

```
document_df[document_df['cluster_label']==4].sort_values(by='filename')
```

	filename	opinion_text	cluster_label
26	buttons_amazon_kindle	I thought it would be fitting to christen my Kindle with the Stephen King novella UR, so went to the Amazon site on my computer and clicked on the button to buy it .\n0 As soon as I'd clicked the button to confirm my order it appeared on my Kindle almost immediately !\n1 ...	4
36	eyesight-issues_amazon_kindle	It feels as easy to read as the K1 but doesn't seem any crisper to my eyes .\n0 the white is really GREY, and to avoid considerable eye, strain I had to refresh pages every other page .\n1 The dream has always been a portable electronic device that could hold a ton of reading material, automate subscriptions and fa...	4
44	fonts_amazon_kindle	Being able to change the font sizes is awesome !\n0 For whatever reason, Amazon decided to make the Font on the Home Screen ...	4
38	navigation_amazon_kindle	In fact, the entire navigation structure has been completely revised , I'm still getting used to it but it's a huge step forward .\n0 ...	4
41	price_amazon_kindle	If a case was included, as with the Kindle 1, that would have been reflected in a higher price .\n0 lower overall price, with nice leather cover .\n1 ...	4

cluster #3

킨들 리뷰 한 개 포함, 대부분 호텔에 대한 리뷰

cluster #4

토요타와 혼다 등의 자동차 리뷰

1.2 Opinion Review Data 실습 – 군집화 알고리즘 적용

```
from sklearn.cluster import KMeans  
  
# 3개의 집합으로 군집화  
km_cluster = KMeans(n_clusters=3, max_iter=10000, random_state=0)  
km_cluster.fit(feature_vect)  
cluster_label = km_cluster.labels_  
  
# 소속 클러스터를 cluster_label 컬럼으로 할당하고 cluster_label 값으로 정렬  
document_df['cluster_label'] = cluster_label  
document_df.sort_values(by='cluster_label')
```

	filename	opinion_text	cluster_label
0	battery-life_ipod_nano_8gb	short battery life I moved up from an 8gb .\n0 I love this ipod except for the battery life .\n1 ...	0
48	display_garmin_navi_255W_gps	3 quot widescreen display was a bonus .\n0 This made for smoother graphics on the 255w of the vehicle moving along displayed roads, where the 750's display was more of a jerky movement .\n1 ...	0
44	fonts_amazon_kindle	Being able to change the font sizes is awesome !\n0 For whatever reason, Amazon decided to make the Font on the Home Screen ...	0
41	price_amazon_kindle	If a case was included, as with the Kindle 1, that would have been reflected in a higher price .\n0 lower overall price, with nice leather cover .\n1 ...	0
40	speed_windows7	Windows 7 is quite simply faster, more stable, boots faster, goes to sleep faster, comes back from sleep faster, manages your files better and on top of that it's beautiful to look at and easy to use .\n0 , faster about 20% to 30% faster at running applications than my Vista , seriously\n1 ...	0
38	navigation_amazon_kindle	In fact, the entire navigation structure has been completely revised , I'm still getting used to it but it's a huge step forward .\n0 ...	0
37	screen_netbook_1005ha	Keep in mind that once you get in a room full of light or step outdoors screen reflections could become annoying .\n0 I've used mine outsi...	0

cluster #0
포터블 전자기기 리뷰

cluster #1
자동차 리뷰

cluster #2
호텔 리뷰

1.2 Opinion Review Data 실습 – 핵심 단어 추출

```
cluster_centers = km_cluster.cluster_centers_
print('cluster_centers shape :',cluster_centers.shape)
print(cluster_centers)

cluster_centers shape : (3, 4611)
[[0.01005322 0.          0.          ... 0.00706287 0.          0.          ]
 [0.          0.00092551 0.          ... 0.          0.          0.          ]
 [0.          0.00099499 0.00174637 ... 0.          0.00183397 0.00144581]]
```

군집별 핵심 단어 추출

- : 각 군집에 속한 문서는 핵심 단어를 주축으로 군집화되어 있음
- : 각 군집을 구성하는 핵심 단어 확인
- : cluster_centers_
 - Kmeans 객체가 제공하는 속성
 - 각 군집을 구성하는 단어 피처가 군집의 중심을 기준으로 얼마나 가깝게 위치해 있는지
 - 행 : 개별 군집
 - 열 : 개별 피처
 - cluster_centers[0,1] : 0번 군집에서 두번째 피처의 위치 값
 - 각 행의 배열 값 : 각 군집 내의 4611개 피처의 위치가 개별 중심과 얼마나 가까운가를 상대 값으로 나타낸 것
 - 1에 가까울수록 중심과 가까움

1.2 Opinion Review Data 실습 – 핵심 단어 추출

```
# 군집별 top n 핵심단어, 그 단어의 중심 위치 상대값, 대상 파일명들을 반환함.
def get_cluster_details(cluster_model, cluster_data, feature_names, clusters_num, top_n_features=10):
    cluster_details = {}

    # cluster_centers array 의 값이 큰 순으로 정렬된 index 값을 반환
    # 군집 중심점(centroid)별 할당된 word 피처들의 거리값이 큰 순으로 값을 구하기 위함.
    centroid_feature_ordered_ind = cluster_model.cluster_centers_.argsort()[:, ::-1]

    #개별 군집별로 iteration하면서 핵심단어, 그 단어의 중심 위치 상대값, 대상 파일명 입력
    for cluster_num in range(clusters_num):
        # 개별 군집별 정보를 담을 데이터 초기화.
        cluster_details[cluster_num] = {}
        cluster_details[cluster_num]['cluster'] = cluster_num

        # cluster_centers_.argsort()[:, ::-1] 로 구한 index 를 이용하여 top n 피처 단어를 구함.
        top_feature_indexes = centroid_feature_ordered_ind[cluster_num, :top_n_features]
        top_features = [feature_names[ind] for ind in top_feature_indexes]

        # top_feature_indexes를 이용해 해당 피처 단어의 중심 위치 상댓값 구함
        top_feature_values = cluster_model.cluster_centers_[cluster_num, top_feature_indexes].tolist()

        # cluster_details 딕셔너리 객체에 개별 군집별 핵심 단어와 중심위치 상대값, 그리고 해당 파일명 입력
        cluster_details[cluster_num]['top_features'] = top_features
        cluster_details[cluster_num]['top_features_value'] = top_feature_values
        filenames = cluster_data[cluster_data['cluster_label'] == cluster_num]['filename']
        filenames = filenames.values.tolist()
        cluster_details[cluster_num]['filenames'] = filenames

    return cluster_details
```

cluster_centers_ 속성값을 이용해 각 군집별 핵심 단어 찾기

: argsort()[:, ::-1]를 통해 cluster_centers 배열 내 값이 큰 순으로 정렬된 위치 인덱스 반환

: get_cluster_details()

- cluster_centers 배열 내에서 가장 값이 큰 데이터의 위치 인덱스 추출
- 해당 인덱스를 이용해 핵심 단어 이름과 그때의 상대 위치 값을 추출해 cluster_details라는 Dict 객체 변수에 기록하고 반환
- 함수 호출-> dictionary를 원소로 가지는 리스트인 cluster_details 반환

1.2 Opinion Review Data 실습 – 핵심 단어 추출

```
def print_cluster_details(cluster_details):
    for cluster_num, cluster_detail in cluster_details.items():
        print('##### Cluster {}'.format(cluster_num))
        print('Top features:', cluster_detail['top_features'])
        print('Reviews 파일명 :', cluster_detail['filenames'][:7])
        print('=====')  
  
feature_names = tfidf_vect.get_feature_names()  
  
cluster_details = get_cluster_details(cluster_model=km_cluster, cluster_data=document_df,\n                                         feature_names=feature_names, clusters_num=3, top_n_features=10 )  
print_cluster_details(cluster_details)  
  
##### Cluster 0  
Top features: ['screen', 'battery', 'keyboard', 'battery life', 'life', 'kindle', 'direction', 'video', 'size', 'voice']  
Reviews 파일명 : ['battery-life_ipod_nano_8gb', 'voice_garmin_nuvi_255W_gps', 'speed_garmin_nuvi_255W_gps', 'size_a  
sus_netbook_1005ha', 'screen_garmin_nuvi_255W_gps', 'battery-life_amazon_kindle', 'satellite_garmin_nuvi_255W_gps'  
]  
=====  
##### Cluster 1  
Top features: ['interior', 'seat', 'mileage', 'comfortable', 'gas', 'gas mileage', 'transmission', 'car', 'perform  
ance', 'quality']  
Reviews 파일명 : ['gas_mileage_toyota_camry_2007', 'comfort_honda_accord_2008', 'interior_toyota_camry_2007', 'tran  
smisson_toyota_camry_2007', 'seats_honda_accord_2008', 'mileage_honda_accord_2008', 'quality_toyota_camry_2007']  
=====  
##### Cluster 2  
Top features: ['room', 'hotel', 'service', 'staff', 'food', 'location', 'bathroom', 'clean', 'price', 'parking']  
Reviews 파일명 : ['room_holiday_inn_london', 'location_holiday_inn_london', 'staff_bestwestern_hotel_sfo', 'service  
_swissotel_hotel_chicago', 'service_bestwestern_hotel_sfo', 'food_holiday_inn_london', 'staff_swissotel_chicago']  
=====
```

cluster_details

- : 개별 군집 번호, 핵심 단어, 핵심 단어 중심 위치 상대값, 파일명 속성값 정보
- : 이를 보기 좋게 표현하기 위해 print_cluster_details() 함수를 통해 출력

cluster #0

포터블 전자제품 리뷰 군집

배터리 수명 등이 핵심 단어로 군집화

cluster #1

자동차 리뷰 군집

실내 인테리어, 좌석, 연료 효율 등이
핵심 단어로 군집화

cluster #2

호텔 리뷰 군집

방과 서비스 등이 핵심 단어로 군집화

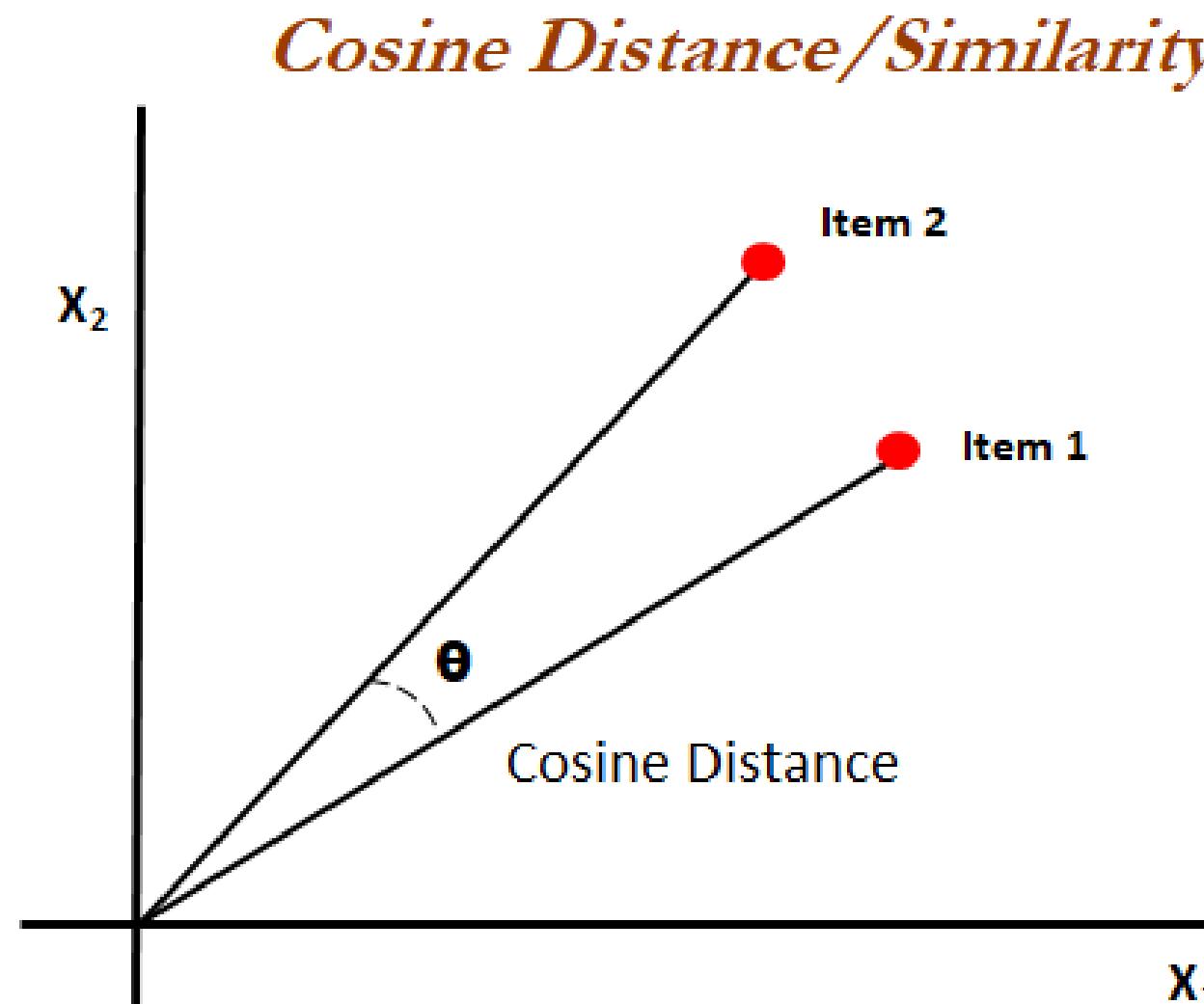
02.문서 유사도



2.1 문서 유사도

문서 유사도 측정 방법 – 코사인 유사도(Cosine Similarity)

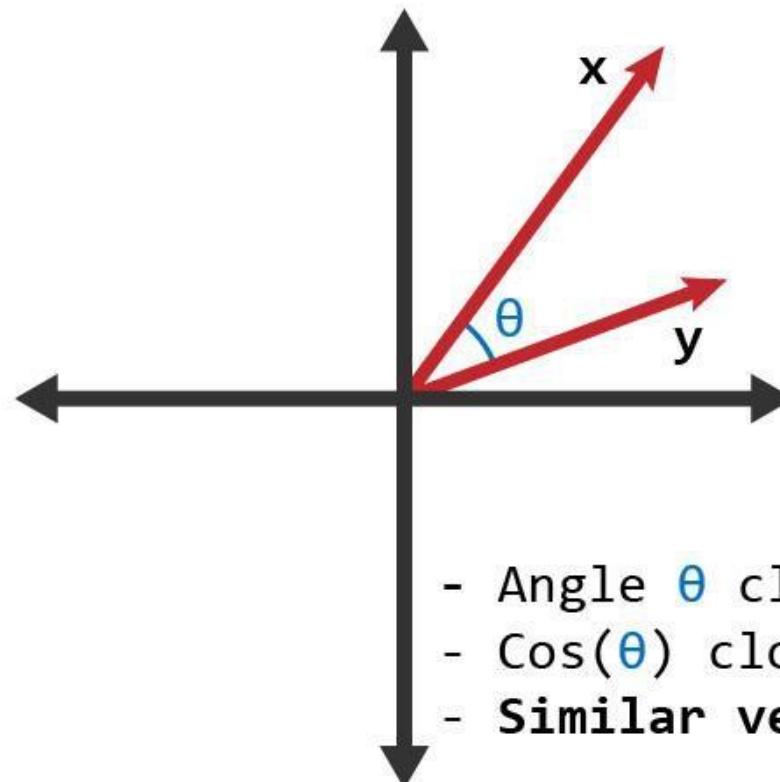
- :: 벡터와 벡터 간의 유사도를 비교할 때 벡터의 크기보다는 벡터의 상호 방향성이 얼마나 유사한지
- : 코사인 유사도는 두 벡터 사이의 사잇값을 구해서 얼마나 유사한지 수치로 적용



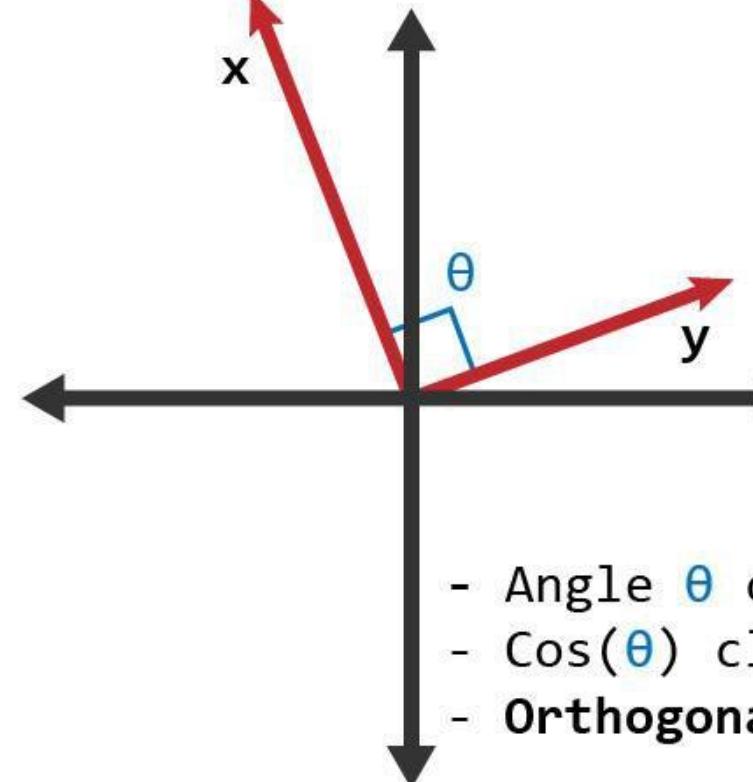
2.1 문서 유사도 – 코사인 유사도

두 벡터 사잇각

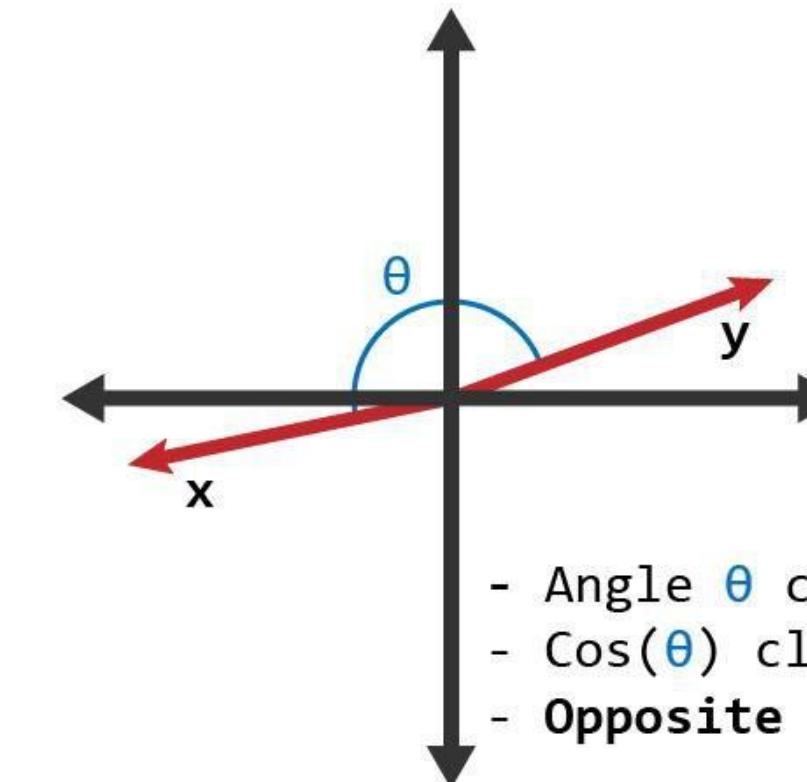
유사 벡터



관련성이 없는 벡터들



반대 관계인 벡터들



$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

2.1 문서 유사도 – 코사인 유사도

코사인 유사도가 문서의 유사도 비교에 자주 사용되는 이유

1. 벡터 간의 크기에 기반한 지표보다 정확도 떨어질 가능성 적음

- 문서를 피처 벡터화 변환하면 차원이 매우 많은 희소 행렬이 되기 쉬움

- 이러한 희소 행렬 기반에서 문서에서 문서 벡터 간의 크기에 기반한 유사도 지표(ex. 유clidean 거리 기반 지표)는 정확도 떨어지기 쉬움

2. 문서가 매우 긴 경우 단어의 빈도수도 더 많을 것이기 때문에 이러한 빈도수에만 기반해서는 공정한 비교할 수 없음

```
import numpy as np

def cos_similarity(v1, v2):
    dot_product = np.dot(v1, v2)
    l2_norm = (np.sqrt(sum(np.square(v1))) * np.sqrt(sum(np.square(v2))))
    similarity = dot_product / l2_norm

    return similarity

from sklearn.feature_extraction.text import TfidfVectorizer

doc_list = ['if you take the blue pill, the story ends',
            'if you take the red pill, you stay in Wonderland',
            'if you take the red pill, I show you how deep the rabbit hole goes']

tfidf_vect_simple = TfidfVectorizer()
feature_vect_simple = tfidf_vect_simple.fit_transform(doc_list)
print(feature_vect_simple.shape)

(3, 18)
```

간단한 문서에 대해 서로 간의 문서 유사도를 코사인 유사도 기반으로 구해보기

cos_similarity() : 두개의 넘파이 배열에 대한 코사인 유사도 구함

doc_lsit로 정의된 3개의 간단한 문서의 유사도를 비교하기 위해 이 문서를 TF-IDF로 벡터화된 행렬로 변환

2.1 문서 유사도 – 코사인 유사도

```
# TFIdfVectorizer로 transform()한 결과는 Sparse Matrix이므로 Dense Matrix로 변환.  
feature_vect_dense = feature_vect_simple.todense()  
  
#첫번째 문장과 두번째 문장의 feature vector 추출  
vect1 = np.array(feature_vect_dense[0]).reshape(-1,)  
vect2 = np.array(feature_vect_dense[1]).reshape(-1,)  
  
#첫번째 문장과 두번째 문장의 feature vector로 두개 문장의 Cosine 유사도 추출  
similarity_simple = cos_similarity(vect1, vect2)  
print('문장 1, 문장 2 Cosine 유사도: {:.3f}'.format(similarity_simple))
```

문장 1, 문장 2 Cosine 유사도: 0.402

```
vect1 = np.array(feature_vect_dense[0]).reshape(-1,)  
vect3 = np.array(feature_vect_dense[2]).reshape(-1,)  
similarity_simple = cos_similarity(vect1, vect3)  
print('문장 1, 문장 3 Cosine 유사도: {:.3f}'.format(similarity_simple))  
  
vect2 = np.array(feature_vect_dense[1]).reshape(-1,)  
vect3 = np.array(feature_vect_dense[2]).reshape(-1,)  
similarity_simple = cos_similarity(vect2, vect3)  
print('문장 2, 문장 3 Cosine 유사도: {:.3f}'.format(similarity_simple))
```

문장 1, 문장 3 Cosine 유사도: 0.404
문장 2, 문장 3 Cosine 유사도: 0.456

반환된 행렬 (3,18)은 희소 행렬이므로 밀집 행렬로 변환

cos_similarity()를 통해 문서 유사도 측정

2.1 문서 유사도 – 코사인 유사도

```
from sklearn.metrics.pairwise import cosine_similarity

similarity_simple_pair = cosine_similarity(feature_vect_simple[0] , feature_vect_simple)
print(similarity_simple_pair)

[[1.          0.40207758 0.40425045]]
```



```
from sklearn.metrics.pairwise import cosine_similarity

similarity_simple_pair = cosine_similarity(feature_vect_simple[0] , feature_vect_simple[1:])
print(similarity_simple_pair)

[[0.40207758 0.40425045]]
```



```
similarity_simple_pair = cosine_similarity(feature_vect_simple , feature_vect_simple)
print(similarity_simple_pair)
print('shape:',similarity_simple_pair.shape)

[[1.          0.40207758 0.40425045]
 [0.40207758 1.          0.45647296]
 [0.40425045 0.45647296 1.          ]]
shape: (3, 3)
```

문서 유사도를 측정하는 사이킷런 API

from sklearn.metrics.pairwise import cosine_similarity

이를 통해 앞 예제의 문서 유사도 측정

cosine_similarity()

: 두개의 입력 파라미터(비교 기준이 되는 문서의 피처 행렬, 비교되는 피처의 행렬)

: 희소행렬, 밀집 행렬 모두 가능, 행렬 또는 배열 모두 가능 -> cos_similarity()와 달리 별도의 변환 작업필요 없음

: 쌍으로 코사인 유사도 값 제공 ndarray 형태

2.2 Opinion Review 데이터 셋을 이용한 문서 유사도 측정

```
from nltk.stem import WordNetLemmatizer
import nltk
import string

remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
lemmar = WordNetLemmatizer()

# 입력으로 들어온 token 단어들에 대해서 lemmatization 어근 변환.
def LemTokens(tokens):
    return [lemmar.lemmatize(token) for token in tokens]

# TfIdfVectorizer 객체 생성 시 tokenizer 인자로 해당 함수를 설정하여 lemmatization 적용
# 입력으로 문장을 받아서 stop words 제거 -> 소문자 변환 -> 단어 토큰화 -> lemmatization 어근 변환.
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))

import pandas as pd
import glob, os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')

path = r'/Users/dohakim/Downloads/OpinosisDataset1.0/topics'
all_files = glob.glob(os.path.join(path, "*.data"))
filename_list = []
opinion_text = []

for file_ in all_files:
    df = pd.read_table(file_, index_col=None, header=0, encoding='latin1')
    filename_ = file_.split('/')[-1]
    filename = filename_.split('.')[0]
    filename_list.append(filename)
    opinion_text.append(df.to_string())

document_df = pd.DataFrame({'filename':filename_list, 'opinion_text':opinion_text})

tfidf_vect = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english', \
                            ngram_range=(1,2), min_df=0.05, max_df=0.85 )
feature_vect = tfidf_vect.fit_transform(document_df['opinion_text'])

km_cluster = KMeans(n_clusters=3, max_iter=10000, random_state=0)
km_cluster.fit(feature_vect)
cluster_label = km_cluster.labels_
cluster_centers = km_cluster.cluster_centers_
document_df['cluster_label'] = cluster_label
```

호텔을 주제로 군집화된 문서를 이용해 특정 문서와 다른 문서 간의 유사도 측정

호텔을 주제로 군집화된 데이터 먼저 추출, 이 데이터에 해당하는 TfIdfVectorizer의 데이터 추출

2.2 Opinion Review 데이터 셋을 이용한 문서 유사도 측정

```
from sklearn.metrics.pairwise import cosine_similarity

# cluster_label=2인 데이터는 호텔로 클러스터링된 데이터임. DataFrame에서 해당 Index를 추출
hotel_indexes = document_df[document_df['cluster_label']==2].index
print('호텔로 클러스터링 된 문서들의 DataFrame Index:', hotel_indexes)

# 호텔로 클러스터링된 데이터 중 첫번째 문서를 추출하여 파일명 표시.
comparison_docname = document_df.iloc[hotel_indexes[0]]['filename']
print('##### 비교 기준 문서명 ', comparison_docname, ' 와 타 문서 유사도#####')

''' document_df에서 추출한 Index 객체를 feature_vect로 입력하여 호텔 클러스터링된 feature_vect 추출
이를 이용하여 호텔로 클러스터링된 문서 중 첫번째 문서와 다른 문서간의 코사인 유사도 측정.'''
similarity_pair = cosine_similarity(feature_vect[hotel_indexes[0]], feature_vect[hotel_indexes])
print(similarity_pair)

호텔로 클러스터링 된 문서들의 DataFrame Index: Int64Index([2, 3, 4, 13, 16, 17, 20, 27, 28, 30, 31, 32, 39, 46, 49, 50],
dtype='int64')
##### 비교 기준 문서명 room_holiday_inn_london 와 타 문서 유사도#####
[[1.          0.19917258 0.22235374 0.37631406 0.26026786 0.15836737
  0.19544761 0.40020673 0.31124876 0.77312013 0.51442299 0.15026112
  0.16717527 0.81484367 0.11154184 0.10831277]]
```

호텔로 군집화된 문서의 인덱스 추출

추출된 인덱스를 이용해 TfidfVectorizer 객체 변수인 feature_vect에서 호텔로 군집화된 문서의 피처 벡터 추출

2.2 Opinion Review 데이터 셋을 이용한 문서 유사도 측정

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# argsort()를 이용하여 앞에제의 첫번째 문서와 타 문서간 유사도가 큰 순으로 정렬한 인덱스 반환하되 자기 자신은 제외.
sorted_index = similarity_pair.argsort()[:, ::-1]
sorted_index = sorted_index[:, 1:]

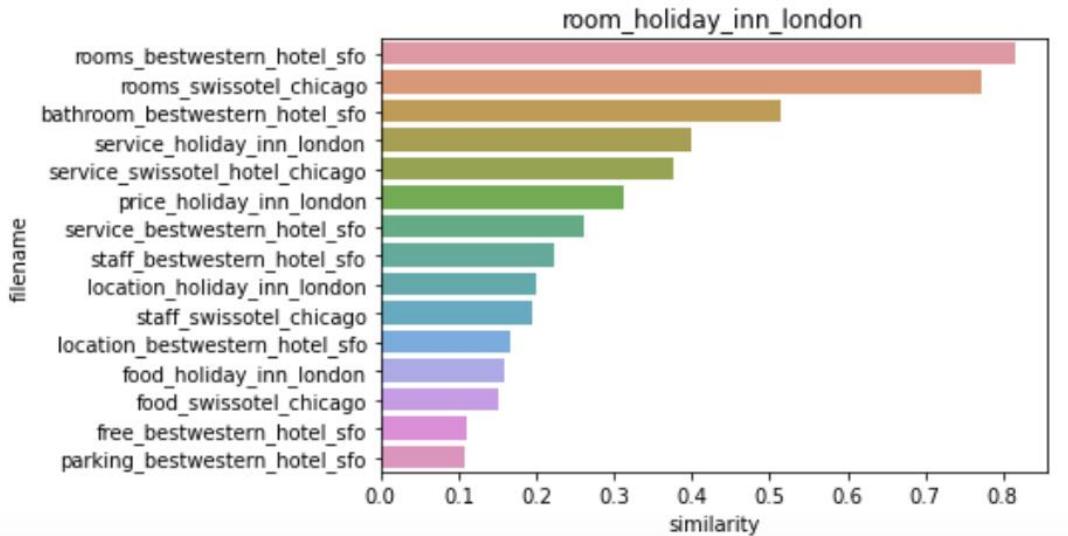
# 유사도가 큰 순으로 hotel_indexes를 추출하여 재 정렬.
hotel_sorted_indexes = hotel_indexes[sorted_index.reshape(-1)]

# 유사도가 큰 순으로 유사도 값을 재정렬하되 자기 자신은 제외
hotel_1_sim_value = np.sort(similarity_pair.reshape(-1))[:, :-1]
hotel_1_sim_value = hotel_1_sim_value[1:]

# 유사도가 큰 순으로 정렬된 Index와 유사도값을 이용하여 파일명과 유사도값을 Seaborn 막대 그래프로 시각화
hotel_1_sim_df = pd.DataFrame()
hotel_1_sim_df['filename'] = document_df.iloc[hotel_sorted_indexes]['filename']
hotel_1_sim_df['similarity'] = hotel_1_sim_value

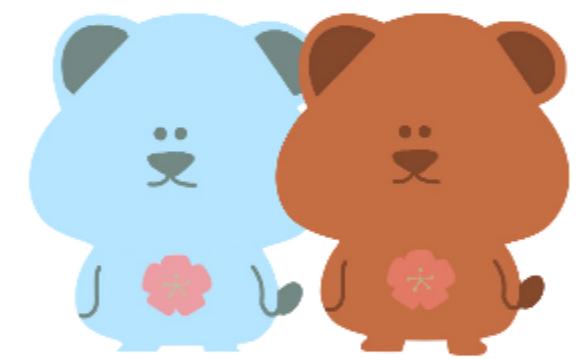
fig1 = plt.gcf()
sns.barplot(x='similarity', y='filename', data=hotel_1_sim_df)
plt.title(comparison_docname)
# fig1.savefig('p553_hotel.tif', format='tif', dpi=300, bbox_inches='tight')

Text(0.5, 1.0, 'room_holiday_inn_london')
```



cosine_similarity()를 통해 얻은 쌍 형태의 ndarray 문서 유사도를 첫번째 문서와 다른 문서 간의 유사도가 높은 순으로 정렬 및 시각화

03. 한글 텍스트 데이터 처리



3.1 한글 NLP 처리의 어려움

한글 언어 처리가 영어 등의 라틴어 처리보다 어려운 이유 1

- 띄어쓰기

한글은 띄어쓰기를 잘못하면 의미가 왜곡되어 전달될 수 있음.

Ex) ‘아버지가 방에 들어가신다’ ‘아버지 가방에 들어가신다’

이에 반해 영어는 띄어쓰기를 잘못하면 잘못된 혹은 없는 단어로 인식되는 것이 대부분임.

Ex) ‘My father enters a room’ ‘My fatherenters a room’

한글의 띄어쓰기는 고등교육을 받은 사람이라도 틀리는 경우가 종종 발생할 정도로 어렵다.

3.1 한글 NLP 처리의 어려움

한글 언어 처리가 영어 등의 라틴어 처리보다 어려운 이유 2

- 다양한 조사

주어나 목적어를 위해 추가되는 조사는 경우의 수가 많기 때문에 어근 추출(Stemming/Lemmatization) 등의 전처리로 제거하기가 까다로움.

Ex) ‘너희 집은 어디 있니? ’에서 ‘집은’이 ‘은’이 조사인지 명사인지 구분하기가 어렵다.

3.2 KoNLPy

형태소 : 단어로서 의미를 가지는 최소 단위

형태소 분석 : 말뭉치를 형태소 어근 단위로 쪼개고 각 형태소에 품사 태깅(POS tagging)을 부착하는 작업

KoNLPy

파이썬의 대표적인 한글 형태소 패키지

- 기존의 C/C++, Java로 잘 만들어진 한글 형태소 엔진을 파이썬 래퍼 기반으로 재작성한 패키지
- 기존의 엔진은 r대로 유지한 채 파이썬 기반에서 인터페이스를 제공하기 때문에 검증된 패키지의 안정성 유지
- **꼬꼬마(Kkma), 한나눔(hannanum), Komoran, 은전한닢 프로젝트(Mecab), Twitter** 5개의 형태소 분석

모듈을 KoNLPy에서 모두 사용 가능 (현재는 윈도우에서도 Mecab 구동 가능)

* Java, JPyte10 설치되어 있어야 함 -> pip install konlpy로 설치

3.3 Mecab

Mecab (은전한닢 프로젝트)

오픈소스 한국어 형태소 분석기

한나눔이나 꼬꼬마(kkma) 등의 기존 형태소 분석기의 띄어쓰기 구분의 오류나 공개 소스를 구하기 어렵다는 문제들을 극복하기 위해 시작됨

원하는 단어가 형태소 분석으로 tagging이 안 될 때 형태소 분석기에서 사용자 사전을 구축해 해당 단어가 형태소 분석기에 tagging 될 수 있도록 할 수 있음

(참고) <https://lsjsj92.tistory.com/612>

3.4 네이버 영화리뷰 데이터 실습

데이터 로딩

<https://github.com/e9t/nsmc>에서 다운로드

ratings.txt – 전체 데이터 세트

ratings_train.txt – 학습 데이터 세트

ratings_text.txt – 테스트 데이터 세트

Naver sentiment movie corpus v1.0

This is a movie review dataset in the Korean language. Reviews were scraped from [Naver Movies](#).

The dataset construction is based on the method noted in [Large movie review dataset](#) from Maas et al., 2011.

Data description

- Each file is consisted of three columns: `id`, `document`, `label`
 - `id` : The review id, provided by Naver
 - `document` : The actual review
 - `label` : The sentiment class of the review. (0: negative, 1: positive)
 - Columns are delimited with tabs (i.e., `.tsv` format; but the file extension is `.txt` for easy access for novices)
- 200K reviews in total
 - `ratings.txt` : All 200K reviews
 - `ratings_test.txt` : 50K reviews held out for testing
 - `ratings_train.txt` : 150K reviews for training

Characteristics

- All reviews are shorter than 140 characters
- Each sentiment class is sampled equally (i.e., random guess yields 50% accuracy)
 - 100K negative reviews (originally reviews of ratings 1-4)
 - 100K positive reviews (originally reviews of ratings 9-10)
 - Neutral reviews (originally reviews of ratings 5-8) are excluded

3.4 네이버 영화리뷰 데이터 실습

```
1 import pandas as pd  
2  
3 train_df = pd.read_csv('ratings_train.txt', sep='\t')  
4 train_df.head(3)
```

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	홈...포스터보고 초딩영화풀....오버연기조차 가볍지 않구나	1
2	10265843	너무재밌었다그래서보는것을추천한다	0

학습 데이터는 탭(\t)으로 칼럼이 분리돼 있음 -> `read_csv()`에서 `sep= '\t'` 으로 설정해 데이터프레임 생성

```
1 train_df['label'].value_counts( )
```



```
0    75173  
1    74827  
Name: label, dtype: int64
```

Label 값은 0과 1이 거의 균등한 분포를 나타내고 있음.

1 - 긍정, 0 - 부정

3.4 네이버 영화리뷰 데이터 실습

```
1 import re  
2  
3 train_df = train_df.fillna(' ')  
4 # 정규 표현식을 이용하여 숫자를 공백으로 변경(정규 표현식으로 '#d'는 숫자를 의미함.)  
5 train_df['document'] = train_df['document'].apply(lambda x : re.sub(r"\d+", " ", x) )  
6  
7 # 테스트 데이터셋을 로딩하고 동일하게 Null 및 숫자를 공백으로 변환  
8 test_df = pd.read_csv('ratings_test.txt', sep='\t')  
9 test_df = test_df.fillna(' ')  
10 test_df['document'] = test_df['document'].apply(lambda x : re.sub(r"\d+", " ", x) )  
11
```

데이터 전처리

train_df에 존재하는 Null을 공백으로 변환 -> fillna() 이용

숫자의 경우 단어적인 의미가 부족하므로 파이썬의 정규 표현식 모듈 re를 이용해 공백으로 변환

-> 테스트 데이터 세트도 파일로딩 후 동일한 데이터 가공을 수행

3.4 네이버 영화리뷰 데이터 실습

```
1 from konlpy.tag import Twitter  
2  
3 twitter = Twitter()  
4 def tw_tokenizer(text):  
5     # 입력 인자로 들어온 text 를 형태소 단어로 토큰화 하여 list 객체 반환  
6     tokens_ko = twitter.morphs(text)  
7     return tokens_ko
```

문장을 형태소 형태로 반환하는 별도의 tokenizer 함수 `tw_tokenizer()` 생성

- SNS 분석에 적합한 **Twitter 클래스**를 한글 형태소 엔진으로 이용
 - Twitter 객체의 **morphs()** 메서드를 이용하면 입력 인자로 들어온 문장을 형태소 단어 형태로 토큰화해 list 객체로 반환
- > 이 함수는 뒤에서 사이킷런의 *TfidfVectorizer* 클래스의 *tokenizer*로 사용됨

3.4 네이버 영화리뷰 데이터 실습

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.model_selection import GridSearchCV
4
5 # Twitter 형태의 morphs() 객체를 이용한 tokenizer를 사용. ngram_range는 (1,2)
6 tfidf_vect = TfidfVectorizer(tokenizer=tw_tokenizer, ngram_range=(1,2), min_df=3, max_df=0.9)
7 tfidf_vect.fit(train_df['document'])
8 tfidf_matrix_train = tfidf_vect.transform(train_df['document'])
```

피처 벡터화

사이킷런의 **TfidfVectorizer**를 이용해 TF-IDF 피처 모델을 생성

tokenizer = tw_tokenizer, ngram_range = (1,2), min_df = 3, max_df = 0.9

3.4 네이버 영화리뷰 데이터 실습

```
1 # Logistic Regression 을 이용하여 감성 분석 Classification 수행.
2 lg_clf = LogisticRegression(random_state=0)
3
4 # Parameter C 최적화를 위해 GridSearchCV 를 이용.
5 params = { 'C': [1 ,3.5, 4.5, 5.5, 10 ] }
6 grid_cv = GridSearchCV(lg_clf , param_grid=params , cv=3 ,scoring='accuracy' , verbose=1 )
7 grid_cv.fit(tfidf_matrix_train , train_df['label'] )
8 print(grid_cv.best_params_ , round(grid_cv.best_score_,4))
```

감성 분석 – 로지스틱 회귀

로지스틱 회귀를 이용해 분류 기반의 감성 분석을 수행

로지스틱 회귀의 하이퍼 파라미터 C의 최적화를 위해 GridSearchCV 이용

Fitting 3 folds for each of 5 candidates, totalling 15 fits

[Parallel(n_jobs=1)]: Done 15 out of 15 | elapsed: 43.6s finished

{'C': 3.5} 0.8593

C = 3.5일 때 최고 0.8593의 정확도

3.4 네이버 영화리뷰 데이터 실습

```
1 from sklearn.metrics import accuracy_score
2
3 # 학습 데이터를 적용한 TfidfVectorizer를 이용하여 테스트 데이터를 TF-IDF 값으로 Feature 벡터화
4 tfidf_matrix_test = tfidf_vect.transform(test_df['document'])
5
6 # classifier는 GridSearchCV에서 최적 파라미터로 학습된 classifier를 그대로 이용
7 best_estimator = grid_cv.best_estimator_
8 preds = best_estimator.predict(tfidf_matrix_test)
9
10 print('Logistic Regression 정확도: ',accuracy_score(test_df['label'],preds))
```

테스트 데이터 세트로 최종 감성 분석 예측

학습할 때 적용한 TfidfVectorizer를 그대로 사용해야 함.

(학습 시 설정된 TfidfVectorizer의 피처 개수와 테스트 데이터를 TfidfVectorizer로 변환할 피처 개수를 같게 하기 위함)

최적 파라미터로 학습된 classifier 이용.

Logistic Regression 정확도: 0.86172

04. 캐글 : 대형 온라인 쇼핑몰 제품 가격 예측



4.1 대회 소개

일본의 대형 온라인 쇼핑몰 Mercari사의 제품에 대해 가격을 예측하는 과제

- 제품에 대한 여러 속성 및 제품 설명 등의 텍스트 데이터로 구성
- 판매자는 제품명, 브랜드명, 카테고리, 제품 설명 등 다양한 속성 정보를 입력하고, ML 모델은 이 속성에 따라 제품 예측 가격을 판매자에게 자동으로 제공

train_id : 데이터 id

name : 제품명

item_condition_id : 판매자가 제공하는 제품 상태

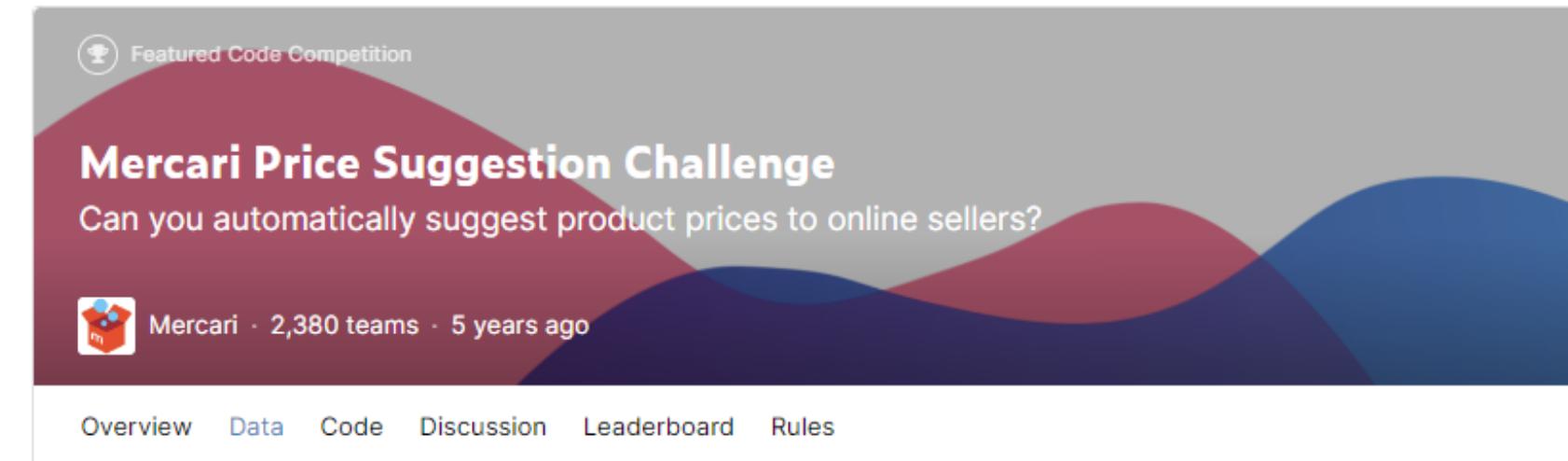
category_name : 카테고리 명

brand_name : 브랜드 이름

price : 제품 가격 (target 값)

shipping : 배송비 무료 여부. 10이면 무료, 00이면 유료

item_description : 제품에 대한 설명



Dataset Description

In this competition, you will predict the sale price of a listing based on information a user provides for this listing. This is a Kernels-only competition, the files in this Data section are downloadable just for your reference in Stage 1. Stage 2 files will only be available in Kernels and not available for download here.

4.2 데이터 전처리

```
1 from sklearn.linear_model import Ridge, LogisticRegression  
2 from sklearn.model_selection import train_test_split, cross_val_score  
3 from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer  
4 import pandas as pd  
5  
6 mercari_df = pd.read_csv('mercari_train.tsv', sep='\t')  
7 print(mercari_df.shape)  
8 mercari_df.head(3)
```

데이터 로드

(1482535, 8)

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	1	No description yet
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0	This keyboard is in great condition and works ...
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a key hol...

4.2 데이터 전처리

```
1 | print(mercari_df.info())
```

피처 type과 Null 여부 확인

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1482535 entries, 0 to 1482534
Data columns (total 8 columns):
train_id          1482535 non-null int64
name              1482535 non-null object
item_condition_id 1482535 non-null int64
category_name     1476208 non-null object
brand_name        849853 non-null object
price             1482535 non-null float64
shipping          1482535 non-null int64
item_description   1482531 non-null object
dtypes: float64(1), int64(3), object(4)
memory usage: 90.5+ MB
None
```

가격에 영향을 미치는 중요 요인인 brand_name이 많은 Null을 가지고 있음.

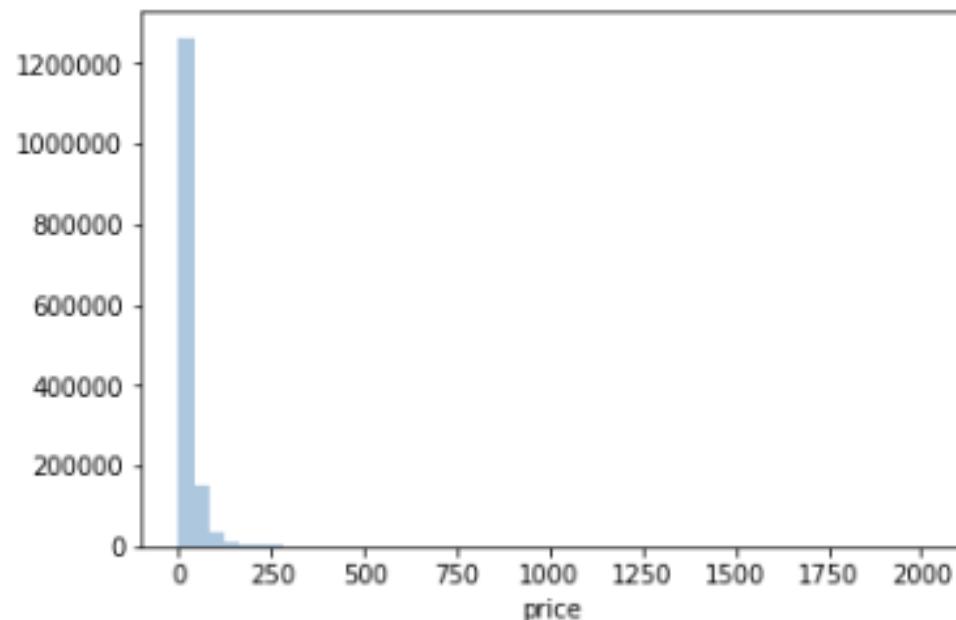
-> Null 데이터는 이후 적절한 문자열로 치환 예정

4.2 데이터 전처리

Target 값의 데이터 분포도

```
1 import matplotlib.pyplot as plt  
2 import seaborn as sns  
3 %matplotlib inline  
4  
5 y_train_df = mercari_df['price']  
6 plt.figure(figsize=(6,4))  
7 sns.distplot(y_train_df,kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b19c73d438>
```



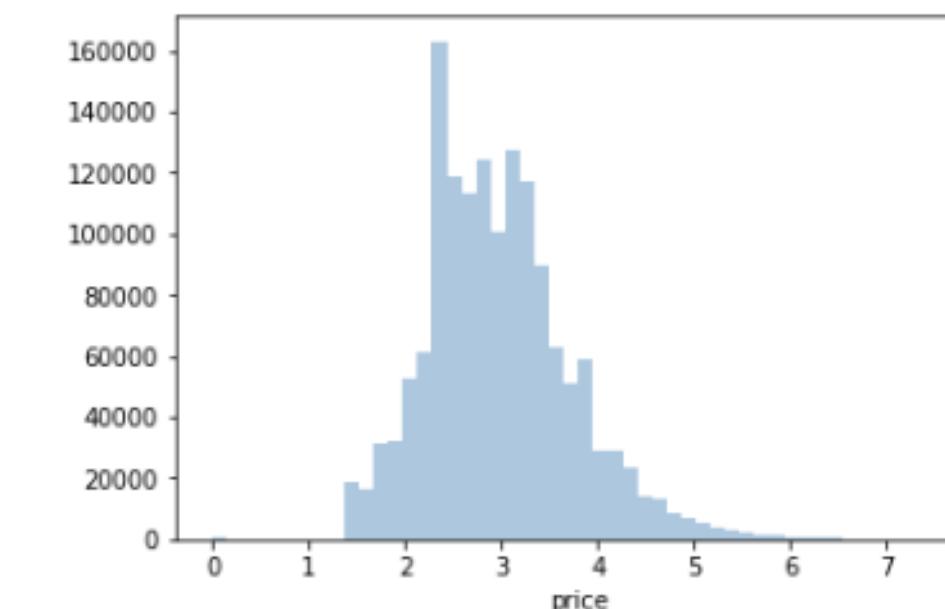
price 값이 비교적 적은 가격의 데이터 값에 왜곡돼 분포

-> 로그 값으로 변환

로그 변환 후 분포도

```
1 import numpy as np  
2  
3 y_train_df = np.log1p(y_train_df)  
4 sns.distplot(y_train_df,kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b19c720da0>
```



비교적 정규 분포에 가까워짐

-> price 칼럼을 로그 변환된 값으로 변경

```
1 mercari_df['price'] = np.log1p(mercari_df['price'])  
2 mercari_df['price'].head(3)
```

```
0    2.397895  
1    3.970292  
2    2.397895  
Name: price, dtype: float64
```

4.2 데이터 전처리

Target 값의 데이터 분포도

```
1 print('Shipping 값 유형:', mercari_df['shipping'].value_counts())
2 print('item_condition_id 값 유형:', mercari_df['item_condition_id'].value_counts())
3
```

Shipping 값 유형:

0 819435

1 663100

Name: shipping, dtype: int64

item_condition_id 값 유형:

1 640549

3 432161

2 375479

4 31962

5 2384

Name: item_condition_id, dtype: int64

```
1 boolean_cond= mercari_df['item_description']=='No description yet'
2 mercari_df[boolean_cond]['item_description'].count()
```

82489

Shipping 칼럼은 비교적 균일

Item_condition_id 칼럼은 1, 2, 3 값이 주를 이룸

Item_description 칼럼은 별도 설명이 없는 'No description yet' 0이 82489건 -> 적절한 값으로 변경해야 함

4.2 데이터 전처리

category_name의 '/'를 기준으로 단어 토큰화 후 각각 별도 피처로 저장

```
1 # apply lambda에서 호출되는 대, 중, 소 분할 할수 생성. 대, 중, 소 값들 리스트 반환
2 def split_cat(category_name):
3     try:
4         return category_name.split('/')
5     except:
6         return ['Other_Null', 'Other_Null', 'Other_Null']
7
8 # 위의 split_cat()을 apply lambda에서 호출하여 대, 중, 소 칼럼들 mercari_df에 생성.
9 mercari_df['cat_dae'], mercari_df['cat_jung'], mercari_df['cat_so'] = [
10     zip(*mercari_df['category_name'].apply(lambda x : split_cat(x)))
11
12 # 대분류만 값의 유형과 갯수를 살펴보고, 중분류, 소분류는 값의 유형이 많으므로 분류 갯수만 추출
13 print('대분류 유형 :', mercari_df['cat_dae'].value_counts())
14 print('중분류 갯수 :', mercari_df['cat_jung'].nunique())
15 print('소분류 갯수 :', mercari_df['cat_so'].nunique())
```

대부분 유형 :	
Women	664385
Beauty	207828
Kids	171689
Electronics	122690
Men	93680
Home	67871
Vintage & Collectibles	46530
Other	45351
Handmade	30842
Sports & Outdoors	25342
Other_Null	6327
Name: cat_dae, dtype: int64	
중분류 갯수 :	114
소분류 갯수 :	871

별도의 split_cat() 함수 생성

- Null이 아닌 경우 split(' ')를 이용해 대, 중, 소분류를 분리 → 리스트로 반환
- Null일 경우 except catch하여 대, 중, 소 분류 모두 'Other Null' 값을 부여

split_cat()을 apply lambda에 호출하면서 zip과 *를 적용해 대, 중, 소 칼럼을 생성

4.2 데이터 전처리

fillna()로 다른 칼럼들의 Null 값 처리

```
1 mercari_df['brand_name'] = mercari_df['brand_name'].fillna(value='Other_Null')
2 mercari_df['category_name'] = mercari_df['category_name'].fillna(value='Other_Null')
3 mercari_df['item_description'] = mercari_df['item_description'].fillna(value='Other_Null')
4
5 # 각 칼럼별로 Null값 건수 확인. 모두 0이 나와야 합니다.
6 mercari_df.isnull().sum()
```

```
train_id      0
name          0
item_condition_id 0
category_name 0
brand_name    0
price         0
shipping      0
item_description 0
cat_dae       0
cat_jung     0
cat_so        0
dtype: int64
```

Brand_name, category_name, item_description 칼럼의 Null 값을 fillna()를 이용해 일괄적으로 ‘Other_Null’로 동일하게 변경

-> 모든 칼럼에서 Null 건수가 0이 됨

4.3 피처 인코딩과 피처 벡터화

문자열 칼럼은 원-핫 인코딩을 수행하거나 피처 벡터화로 변환해야 함

본 대회에서 예측 모델은 상품 가격을 예측해야 하므로 회귀 모델을 기반으로 함

-> 선형 회귀에서는 원-핫 인코딩 선호

피처 벡터화의 경우 짧은 텍스트 – Count 기반 벡터화, 긴 텍스트 – TF-IDF 기반 벡터화 적용

4.3 피처 인코딩과 피처 벡터화

인코딩, 벡터화 여부 검토

```
1 print('brand name 의 유형 개수 :', mercari_df['brand_name'].nunique())
2 print('brand name sample 5건 :#n', mercari_df['brand_name'].value_counts()[:5])
```

brand name 의 유형 개수 : 4810
brand name sample 5건 :
Other_Null 632682
PINK 54088
Nike 54043
Victoria's Secret 48036
LuLaRoe 31024
Name: brand_name, dtype: int64

```
1 print('name 의 종류 갯수 :', mercari_df['name'].nunique())
2 print('name sample 7건 :#n', mercari_df['name'][:7])
```

name 의 종류 갯수 : 1225273
name sample 7건 :
0 MLB Cincinnati Reds T Shirt Size XL
1 Razer BlackWidow Chroma Keyboard
2 AVA-VIV Blouse
3 Leather Horse Statues
4 24K GOLD plated rose
5 Bundled items requested for Ruie
6 Acacia pacific tides santorini top
Name: name, dtype: object

```
1 pd.set_option('max_colwidth', 200)
2
3 # item_description의 평균 문자열 개수
4 print('item_description 평균 문자열 개수:', mercari_df['item_description'].str.len().mean())
5
6 mercari_df['item_description'][:2]
```

item_description 평균 문자열 개수: 145.7113889385411
0
No description yet
1 This keyboard is in great condition and works like it came out of the box. All of the ports are tested and work perfectl
y. The lights are customizable via the Razer Synapse app on your PC.
Name: item_description, dtype: object

brand_name

대부분 명료한 문자열로 돼 있음

-> 원-핫 인코딩

category_name

cat_dae, cat_jung, cat_so 칼럼도 원-핫 인코딩 수행

name

유형이 매우 많고, 적은 단어 위주의 텍스트

형태

-> Count 기반 피처 벡터화

shipping

0과 1 두 가지 유형의 값

-> 원-핫 인코딩

item_condition

1, 2, 3, 4, 5 다섯 가지 유형의 값

-> 원-핫 인코딩

item_description

데이터 세트에서 가장 긴 텍스트

-> TF-IDF 기반 피처 벡터화

4.3 피처 인코딩과 피처 벡터화

피처 벡터화 - name, item_description

```
1 # name 속성에 대한 feature vectorization 블록
2 cnt_vec = CountVectorizer()
3 X_name = cnt_vec.fit_transform(mercari_df.name)
4
5 # item_description에 대한 feature vectorization 블록
6 tfidf_descp = TfidfVectorizer(max_features = 50000, ngram_range=(1,3), stop_words='english')
7 X_descp = tfidf_descp.fit_transform(mercari_df['item_description'])
8
9 print('name vectorization shape:', X_name.shape)
10 print('item_description vectorization shape:', X_descp.shape)
11
```

```
name vectorization shape: (1482535, 105757)
item_description vectorization shape: (1482535, 50000)
```

이때 CountVectorizer, TfidfVectorizer가 fit_transform()으로 반환하는 데이터는 **희소 행렬 데이터**

- > 반환된 희소 행렬 객체 변수를 결합해 새로운 데이터 세트로 구성해야 함
- > 인코딩 대상 칼럼도 희소 행렬 형태로 인코딩을 적용한 후 결합해야 함

4.3 피처 인코딩과 피처 벡터화

원-핫 인코딩 - brand_name, item_condition_id, shipping, cat_dae, cat_jung, cat_so

```
1 from sklearn.preprocessing import LabelBinarizer  
2  
3 # brand_name, item_condition_id, shipping 각 피처들을 회소 행렬 원-핫 인코딩 변환  
4 lb_brand_name = LabelBinarizer(sparse_output=True)  
5 X_brand = lb_brand_name.fit_transform(mercari_df['brand_name'])  
6  
7 lb_item_cond_id = LabelBinarizer(sparse_output=True)  
8 X_item_cond_id = lb_item_cond_id.fit_transform(mercari_df['item_condition_id'])  
9  
10 lb_shipping = LabelBinarizer(sparse_output=True)  
11 X_shipping = lb_shipping.fit_transform(mercari_df['shipping'])  
12  
13 # cat_dae, cat_jung, cat_so 각 피처들을 회소 행렬 원-핫 인코딩 변환  
14 lb_cat_dae = LabelBinarizer(sparse_output=True)  
15 X_cat_dae = lb_cat_dae.fit_transform(mercari_df['cat_dae'])  
16  
17 lb_cat_jung = LabelBinarizer(sparse_output=True)  
18 X_cat_jung = lb_cat_jung.fit_transform(mercari_df['cat_jung'])  
19  
20 lb_cat_so = LabelBinarizer(sparse_output=True)  
21 X_cat_so = lb_cat_so.fit_transform(mercari_df['cat_so'])
```

```
1 print(type(X_brand), type(X_item_cond_id), type(X_shipping))  
2 print('X_brand shape:{0}, X_item_cond_id shape:{1}'.format(X_brand.shape, X_item_cond_id.shape))  
3 print('X_shipping shape:{0}, X_cat_dae shape:{1}'.format(X_shipping.shape, X_cat_dae.shape))  
4 print('X_cat_jung shape:{0}, X_cat_so shape:{1}'.format(X_cat_jung.shape, X_cat_so.shape))  
  
<class 'scipy.sparse.csr.csr_matrix'> <class 'scipy.sparse.csr.csr_matrix'> <class 'scipy.sparse.csr.csr_matrix'>  
X_brand shape:(1482535, 4810), X_item_cond_id shape:(1482535, 5)  
X_shipping shape:(1482535, 1), X_cat_dae shape:(1482535, 11)  
X_cat_jung shape:(1482535, 114), X_cat_so shape:(1482535, 871)
```

사이킷런에서 원-핫 인코딩을 위해 지원하는 클래스 - OneHotEncoder와 LabelBinarizer

LabelBinarizer 클래스는 회소 행렬 형태의 원-핫 인코딩 변환을 지원 -> sparse_out = True 설정

-> 인코딩 변환된 데이터 세트는 CSR 형태로 변환된 csr_matrix 타입

4.3 피처 인코딩과 피처 벡터화

피처 벡터화 변환된 데이터 세트와 희소 인코딩 변환된 데이터 세트 결합

```
1 from scipy.sparse import hstack
2 import gc
3
4 sparse_matrix_list = (X_name, X_descp, X_brand, X_item_cond_id,
5                       X_shipping, X_cat_dae, X_cat_jung, X_cat_so)
6
7 # 사이파이 sparse 모듈의 hstack 함수를 이용하여 앞에서 인코딩과 Vectorization을 수행한 데이터 셋을 모두 결합.
8 X_features_sparse = hstack(sparse_matrix_list).tocsr()
9 print(type(X_features_sparse), X_features_sparse.shape)
10
11 # 데이터 셋이 메모리를 많이 차지하므로 사용 용도가 끝났으면 바로 메모리에서 삭제.
12 del X_features_sparse
13 gc.collect()
```

<class 'scipy.sparse.csr.csr_matrix'> (1482535, 161569)

사이파이 sparse의 hstack()을 이용해 결합

4.4 릿지 회귀 모델 구축 및 평가

RMSLE

오류값에 로그를 취해 RMSE를 구하는 방식.

-> 높은 값에서 오류가 발생할 경우 오류값이 더 커지는 것을 억제하기 위함

$$RMSLE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(\hat{y}_i + 1) - \log(y_i + 1))^2}$$

$\log(0)$ 이 발생하는 것을 방지하기 위해 +1을 해줌

4.4 릿지 회귀 모델 구축 및 평가

모델 평가 로직을 함수화

```
1 def rmsle(y , y_pred):  
2     # underflow, overflow를 막기 위해 log가 아닐 log1p로 rmsle 계산  
3     return np.sqrt(np.mean(np.power(np.log1p(y) - np.log1p(y_pred), 2)))  
4  
5 def evaluate_org_price(y_test , preds):  
6  
7     # 원본 데이터는 log1p로 변환되었으므로 expm1으로 원복 필요.  
8     preds_exmpm = np.expm1(preds)  
9     y_test_exmpm = np.expm1(y_test)  
10  
11    # rmsle로 RMSLE 값 추출  
12    rmsle_result = rmsle(y_test_exmpm, preds_exmpm)  
13    return rmsle_result
```

`rmsle((y, y_pred)`) – RMSLE를 구하는 함수

학습할 모델이 사용할 price 값은 로그 변환된 price 값이므로 예측값 역시 로그 변환된 값임

-> 예측된 price 값을 다시 **지수 변환**을 통해 원복해야 함

`evaluate_org_price(y_text, preds)` – 원복된 데이터를 기반으로 RMSLE 적용하는 함수

4.4 릿지 회귀 모델 구축 및 평가

모델 학습/예측하는 로직을 함수화

```
1 import gc
2 from scipy.sparse import hstack
3
4 def model_train_predict(model, matrix_list):
5     # scipy.sparse 모듈의 hstack 를 이용하여 sparse matrix 결합
6     X = hstack(matrix_list).tocsr()
7
8     X_train, X_test, y_train, y_test = train_test_split(X, mercari_df['price'],
9                                                       test_size=0.2, random_state=156)
10
11    # 모델 학습 및 예측
12    model.fit(X_train, y_train)
13    preds = model.predict(X_test)
14
15    del X, X_train, X_test, y_train
16    gc.collect()
17
18    return preds, y_test
```

Model_train_predict(model, matrix_list)

- model : 사이킷런의 회귀 estimator 객체
- matrix_list : 최종 데이터 세트로 결합할 회소 행렬 리스트

4.4 릿지 회귀 모델 구축 및 평가

Ridge 회귀 예측

```
1 linear_model = Ridge(solver = "lsqr", fit_intercept=False)
2
3 sparse_matrix_list = (X_name, X_brand, X_item_cond_id,
4                         X_shipping, X_cat_dae, X_cat_jung, X_cat_so)
5 linear_preds, y_test = model_train_predict(model=linear_model, matrix_list=sparse_matrix_list)
6 print('Item Description을 제외했을 때 rmsle 값:', evaluate_org_price(y_test, linear_preds))
7
8 sparse_matrix_list = [X_descp, X_name, X_brand, X_item_cond_id,
9                         X_shipping, X_cat_dae, X_cat_jung, X_cat_so]
10 linear_preds, y_test = model_train_predict(model=linear_model, matrix_list=sparse_matrix_list)
11 print('Item Description을 포함한 rmsle 값:', evaluate_org_price(y_test, linear_preds))
12
```

Item Description을 제외했을 때 rmsle 값: 0.5021632139113013
Item Description을 포함한 rmsle 값: 0.47122043277496645

예측에 item_description과 같은 텍스트 형태의 속성이 포함되었을 때 rmsle 값이 많이 감소

-> Item description의 영향이 중요하다.

4.5 LightGBM 회귀 모델 구축과 양상불을 이용한 최종 예측 평가

LightGBM을 이용해 회귀 수행한 뒤,

앞서 구한 릿지 모델 예측값과 LightGBM 모델 예측값을 간단한 양상을 방식으로 섞어서 최종 회귀 예측값을 평가

LightGBM 회귀 예측

```
1 from lightgbm import LGBMRegressor  
2  
3 sparse_matrix_list = (X_descp, X_name, X_brand, X_item_cond_id,  
4                         X_shipping, X_cat_dae, X_cat_jung, X_cat_so)  
5  
6 lgbm_model = LGBMRegressor(n_estimators=200, learning_rate=0.5, num_leaves=125, random_state=156)  
7 lgbm_preds, y_test = model_train_predict(model = lgbm_model, matrix_list=sparse_matrix_list)  
8 print('LightGBM rmsle 값:', evaluate_org_price(y_test, lgbm_preds))
```

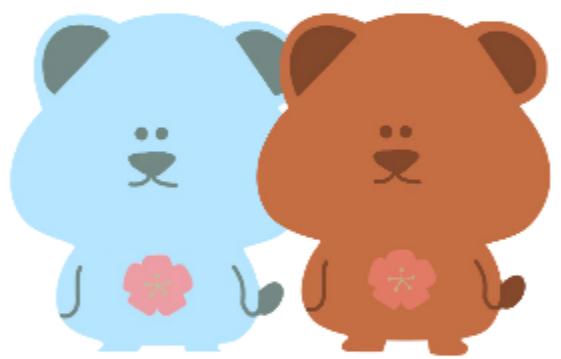
LightGBM rmsle 값: 0.4558577120744113

LightGBM 예측값과 Ridge 예측값을 양상불해 최종 예측 결괏값 도출, 평가

```
1 preds = lgbm_preds * 0.45 + linear_preds * 0.55  
2 print('LightGBM과 Ridge를 ensemble한 최종 rmsle 값:', evaluate_org_price(y_test, preds))
```

LightGBM과 Ridge를 ensemble한 최종 rmsle 값: 0.4501738147377442

05. 로그 분석을 통한 보안 위험도 예측 AI 경진대회



5.1 대회 소개

[로그 분석을 통한 보안 위험도 예측 AI 경진대회]

목적

- 새롭게 설치된 서버에 대해 인공지능으로 침해 위험도를 8단계로 분류하고자 함
- 로그 데이터를 통해 시스템의 보안 위험도 등급 예측
- 기존에 없던 패턴의 공격 탐지

train.csv

- Id : 데이터 식별자
- level : 보안 위험 등급(0~6)
- full_log : 학습 데이터 전체 로그

test.csv

- Id : 데이터 식별자
- full_log : 테스트 데이터의 전체 로그
- Train.csv에 존재하지 않는 level 존재 (0~7)

5.2 EDA

[데이터 & 라이브러리 Import]

```
import pandas as pd
import numpy as np
import os
import pandas as pds
from dask import dataframe
import re
import numpy as np
import seaborn as sbn
from matplotlib import pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import wordnet
```

```
train = pd.read_csv("train.csv")
test= pd.read_csv("test.csv")
```

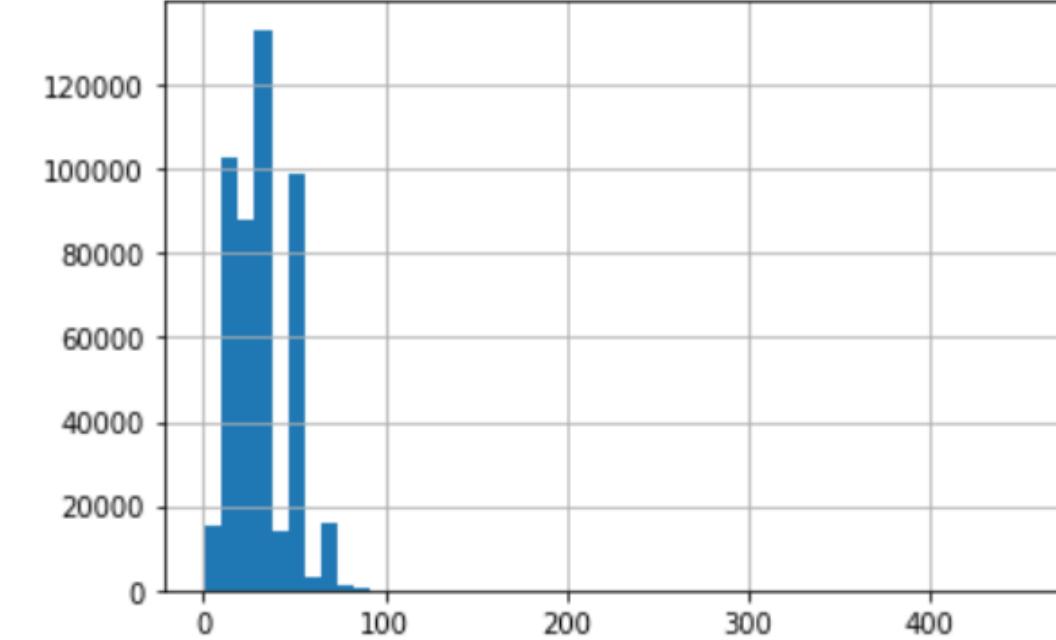
```
train.head()
```

	id	level	full_log
0	0	0	Sep 24 10:02:22 localhost kibana: {"type":"err...
1	1	0	Feb 8 16:21:00 localhost logstash: [2021-02-0...
2	2	0	Jan 13 01:50:40 localhost kibana: {"type":"err...
3	3	0	Jan 4 10:18:31 localhost kibana: {"type":"err...
4	4	1	type=SYSCALL msg=audit(1603094402.016:52981): ...

[간단한 EDA]

```
train['full_log'].str.split(' ').str.len().hist(bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff3f56413d0>
```



```
train['level'].value_counts()
```

```
0    334065
1    132517
3     4141
5     2219
2      12
4      10
6       8
Name: level, dtype: int64
```

5.3 전처리

[다양한 기호 및 숫자 제거]

- 의미없는 숫자와 기호들이 분류를 방해
 - 정규표현식 모듈 re 이용
 - re.sub(): 정규표현식과 일치하는 부분을 다른 문자열로 대체

```
# 다양한 기호 및 숫자 제거
lit = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
def mask(tt):
    tt=tt.apply(lambda x: re.sub(r'#[\n\r]', ' ',x))
    tt=tt.apply(lambda x: re.sub(r'^a-zA-Zㄱ-ㅣ가-힣0-9:#$@*~&%+!_|#[\n\r]+|[#\n\r][#\n\r]>`#[\n\r]…`]', ' ',x))
    #tt=tt.apply(lambda x: re.sub(r' ?(?P<note>[:#=][#\n\r]./,#[<#>]) ?', ' #[<note> ', x))
    tt=tt.apply(lambda x: re.sub(r'[0-9]+', ' ',x))
    tt=tt.apply(lambda x: re.sub(r'"':/()', ' ',x))
    tt=tt.apply(lambda x: re.sub(r':', ' ',x))
    tt=tt.apply(lambda x: re.sub(r',', ' ',x))
    # = tt.apply(lambda x: re.sub(r'(',')',x))
    #t = tt.apply(lambda x: re.sub(r')', ' ',x))
    tt=tt.apply(lambda x: re.sub(r'[-+=#/?:^$.@*~&%+!_|#[\n\r]+|[#\n\r][#\n\r]>`#[\n\r]…`]', ' ',x))
    for st in lit:
        st = " "+st + " "
        tt=tt.apply(lambda x: re.sub(st, ' ',x))
    tt=tt.apply(lambda x: re.sub(r'#[\n\r]+', ' ',x))

return tt
```

메타 문자	의미
[]	대괄호 안에 포함된 문자들 중 하나와 매치 [^a-zA-Z] : 모든 알파벳 제외
.	줄바꿈 문자를 제외한 모든 문자와 매치
*	앞에 오는 문자가 몇 개가 오든 모두 매치 lo*I : ll, lol, loool, looooooooooo
+	앞에 오는 문자가 최소 한 번 이상 반복되면 매치 lo+I : lol, loool, looooooooooo
?	앞에 있는 문자가 없거나 하나 있을 때 매치 lo?I : ll, loi
{m, n}	앞에 있는 문자가 m번에서 n번까지 반복될 때 매치
	or의 의미로 정규표현식들 중 어느 하나와 매치
^, \$	각각 시작과 끝에 앞에 있는 문자가 있으면 매치

5.3 전처리

[영어 단어가 아니거나 3글자 미만인 경우 삭제]

```
#영어 단어 + 3글자 이상인 경우만 True
from nltk.corpus import words
def check_words(word_list: list):
    re = [False] * len(word_list)
    for i, word in enumerate(word_list):
        if len(word) < 3:
            continue
        word = word.lower()
        if word in words.words():
            re[i] = True
    return re

word_list = ['Promiscuous', 'ab', 'nihongo', 'abstract', 'pedo', 'gid']
```

[카운트 벡터화]

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier

#남은 단어가 없는 경우에는 'missing' 으로 대체해주었습니다.
train['cut'] = train['cut'].replace('', 'missing', regex=True)
test['cut'] = test['cut'].replace('', 'missing', regex=True)

train = train.fillna("missing")
test = test.fillna("missing")

vectorizer=CountVectorizer(analyzer="word", max_features=20000)
train_features=vectorizer.fit_transform(train_text)
test_features=vectorizer.transform(test_text)
```

5.4 모델링 & 결론

[RandomForestClassifier]

```
forest=RandomForestClassifier(n_estimators=100,random_state = 1 )  
forest.fit(train_features,train_level)
```

```
results=forest.predict(test_features)  
results_proba=forest.predict_proba(test_features)
```

- test.csv에만 존재하는 level 7을 잘 구별하는 것이 관건
- 로그 데이터 전처리를 진행한 후 validation을 통해 등급별 threshold를 상세 설정한 코드가 높은 성능을 보임

[Threshold 조절]

```
#Validation을 통한 threshold 조절  
results[np.where((np.max(results_proba, axis=1)<0.5) & (results == 0))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.5) & (results == 1))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.58) & (results == 2))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.95) & (results == 3))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.58) & (results == 4))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.58) & (results == 5))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.58) & (results == 6))[0]]=7
```

```
#Validation을 통한 threshold 조절  
results[np.where((np.max(results_proba, axis=1)<0.94001) & (np.max(results_proba, axis=1) > 0.93999) & (results == 1))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.611657) & (np.max(results_proba, axis=1) > 0.6116568) & (results == 0))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.571657) & (np.max(results_proba, axis=1) > 0.5716568) & (results == 0))[0]]=7  
results[np.where((np.max(results_proba, axis=1)<0.68001) & (np.max(results_proba, axis=1) > 0.67999) & (results == 5))[0]]=7
```

06. 금융문자 분석 경진대회



6.1 대회 소개

[금융 문자 분석 경진대회]

배경

- 교묘하고 지능적인 스미싱 문자 패턴으로 고객들의 피해 사례가 증가
- 이를 방지하기 위해 문자데이터를 활용한 스미싱 탐지 모델 개발

train.csv

- Id : 고유 구분 번호
- year_month : 문자를 전송 받은 년도/월
- text : 문자 내용
- Smishing : 해당 문자의 스미싱 여부 (0-스미싱 아님 / 1-스미싱)

6.2 데이터 로드

[라이브러리 임포트]

```
import pandas as pd # 데이터 전처리
import numpy as np # 데이터 전처리
import random #데이터 전처리
from pandas import DataFrame #데이터 전처리
from collections import Counter #데이터 전처리

from tqdm import tqdm #시간 측정용

from sklearn.feature_extraction.text import CountVectorizer # model setting
from sklearn.model_selection import train_test_split # model setting

from sklearn.naive_bayes import MultinomialNB # model 관련
from sklearn.metrics import roc_auc_score # model 성능 확인
```

[데이터 확인]

```
train = pd.read_csv("train.csv") #해당 14th data의 csv 파일 중 train.csv 불러오기
```

```
train.head(2)
```

```
test = pd.read_csv("public_test.csv")
```

```
test.head(2)
```

	id	smishing
0	340000	1.000000e-09
1	340001	1.000000e-09

```
train.shape, test.shape,
```

((295945, 4), (1626, 3))

- train data : 29만 개의 행
- test data : 1626개의 행

6.3 전처리

[mecab]

- 한글 형태소 분석기 -> 토큰화 진행
- 토큰화
 - 문장 토큰화 : 문장의 마지막을 뜻하는 기호에 따라 문장 분리
 - 단어 토큰화 : 문장을 단어로 분리

```
from konlpy.tag import Mecab  
tokenizer = Mecab()
```

[mecab 전처리 진행]

```
import konlpy  
from konlpy.tag import Mecab  
  
tokenizer = Mecab() # setting tokenizer using Mecab()
```

```
train_doc = [ ( tokenizer.pos(x), y ) for x, y in tqdm( zip( train_xx['text'], train_yyy['smishing'] ) )  
test_doc = [ ( tokenizer.pos(x), y ) for x, y in tqdm( zip( test_xx['text'], test_yyy['smishing'] ) ) ]
```

6.3 전처리

[stopword 제거]

- 스톱 워드 : 분석에 큰 의미가 없는 단어

```
stopwords = ['XXX', '.', '을', '를', '이', '가', '-', '(', ')', ':', '!', '?', ')-', '.-', '—', 'XXXXXX', '..', '.(', '은', '는'] #필요없는 단어 리스트

def get_couple(_words): #필요없는 단어들 없애는 함수
    global stopwords
    _words = [x for x in _words if x[0] not in stopwords]
    l = len(_words)
    for i in range(l-1):
        yield _words[i][0], _words[i+1][0]
```

```
X_train, Y_train = [], []
for lwords in train_doc:
    Y_train.append(lwords[1])

    temp = []
    for x, y in get_couple(lwords[0]):
        temp.append("{}{}".format(x, y))

    X_train.append(" ".join(temp))
```

```
X_test = []
for lwords in test_doc:

    temp = []
    for x, y in get_couple(lwords[0]):
        temp.append("{}{}".format(x, y))

    X_test.append(" ".join(temp))
```

- 필요없는 단어를 제거함과 동시에 모형에 사용하기 위한 X_train, X_test, y_train, y_test 생성

6.3 전처리

[stopword 제거]

- 스톱 워드 : 분석에 큰 의미가 없는 단어

```
stopwords = ['XXX', '.', '을', '를', '이', '가', '-', '(', ')', ':', '!', '?', ')-', '.-', '—', 'XXXXXX', '..', '.(', '은', '는'] #필요없는 단어 리스트

def get_couple(_words): #필요없는 단어들 없애는 함수
    global stopwords
    _words = [x for x in _words if x[0] not in stopwords]
    l = len(_words)
    for i in range(l-1):
        yield _words[i][0], _words[i+1][0]
```

```
X_train, Y_train = [], []
for lwords in train_doc:
    Y_train.append(lwords[1])

    temp = []
    for x, y in get_couple(lwords[0]):
        temp.append("{}{}".format(x, y))

    X_train.append(" ".join(temp))
```

```
X_test = []
for lwords in test_doc:

    temp = []
    for x, y in get_couple(lwords[0]):
        temp.append("{}{}".format(x, y))

    X_test.append(" ".join(temp))
```

- 필요없는 단어를 제거함과 동시에 모형에 사용하기 위한 X_train, X_test, y_train, y_test 생성

6.4 벡터화 & 모델링

[카운트 기반 벡터화]

- CountVectorizer을 이용해 카운트 기반 벡터화 구현
- 피처값을 count로 부여하여 count값이 높을수록 중요한 단어로 고려

```
v=CountVectorizer()
```

```
v.fit(X_train)
```

```
vec_x_train=v.transform(X_train).toarray()
```

```
vec_x_test=v.transform(X_test).toarray()
```

[나이브 베이즈 분류기] <https://zephyrus1111.tistory.com/230>

- 클래스가 주어졌을 때 독립 변수의 조건부 확률에
독립 가정을 추가한 베이즈 분류기
- MultinomialNB
빈도수를 나타내는 설명변수가 있을 경우 사용하는 NB 분류기

```
m1=MultinomialNB()
```

```
m1.fit(vec_x_train,Y_train)
```

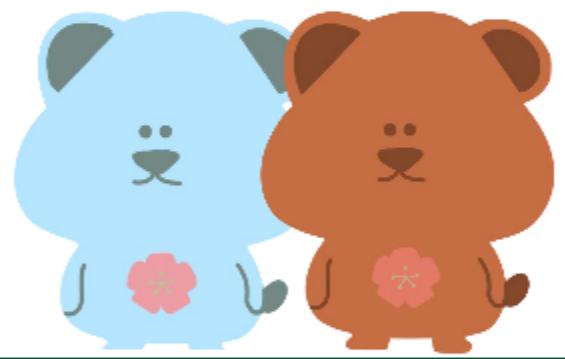
```
y_train_pred1=m1.predict_proba(vec_x_train)
```

```
y_train_pred1_one=[ i[1] for i in y_train_pred1]
```

```
y_test_pred1=m1.predict_proba(vec_x_test)
```

```
y_test_pred1_one=[ i[1] for i in y_test_pred1]
```

07. 자연어 기반 기후기술분류 AI 경진대회



7.1 대회 소개

[자연어 기반 기후기술분류 AI 경진대회]

배경

- 탄소중립 실현을 위한 기후기술 ➔ 매우 광범위
- 국가 연구개발과제를 ‘기후기술분류체계’에 맞추어 라벨링하는 알고리즘 개발

train.csv

- Index
- 제출년도
- 사업명
- 사업부처명
- 계속과제여부
- 내역사업명
- 과제명
- 요약문_연구목표
- 요약문_연구내용
- 요약문_기대효과
- 요약문_한글키워드
- 요약문_영문키워드

labels_mapping.csv

- label
- label과 기후기술분류체계를 mapping한 meta data

7.2 데이터 EDA

[데이터 및 라이브러리 import]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import json
import os
import tqdm

from konlpy.tag import Okt

import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import log_loss, accuracy_score, f1_score
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier

train=pd.read_csv('train.csv')
test=pd.read_csv('test.csv')
```

```
train.head(2)
```

index	제출년도	사업명	사업부 과제명	계속 과제 여부	내역사업명	과제명	요약문_연구목표	요약문_연구내용	요약문_기대효과	요약문_한글키워드	요약문_영문키워드	label
0	0	2016 농업기초기반 연구	농촌 진흥청	신규	농산물안전성 연구	유전정보를 활용한 새로운 해충 분류군 동정기술 개발	○ 새로운 해충분류군의 동정기술 개발 및 유입확산 추적	(가) 외래 및 돌발해충의 발생 조사 및 종 동정 WnWnWn ○ 대상해충 : 최...	○ 새로운 돌발 및 외래해충의 신속, 정확한 동정법 향상 WnWnWn ○ 돌발 및 외래...	뉴클레오티드 염기서열, 문자마커, 종 동정, 침샘, 전사체	nucleotide sequence, molecular marker, species...	24
1	1	2019 이공학학술연구기반구축 (R&D)	교육부	신규	지역대학우수 과학자지원사업(1년~5년)	대장암의 TRAIL 내성 표적 인자 발굴 및 TRAIL 반응 예측 유전자 지도 구축...	최종목표: TRAIL 감수성 표적 유전자를 발굴하고 내성 제어 기전을 연구. 발굴된...	1차년도 Wn1) Microarray를 통한 선천적 TRAIL 내성 표적 후보 유전자...	1) TRAIL 내성 특이적 표적분자를 발굴하고, 이를 이용한 TRAIL 효과 증진...	대장암, 항암제 내성, 세포사멸, 유전자발굴	TRAIL, Colorectal cancer, TRAIL resistance, Apopt...	0

7.2 데이터 EDA

[시각화]

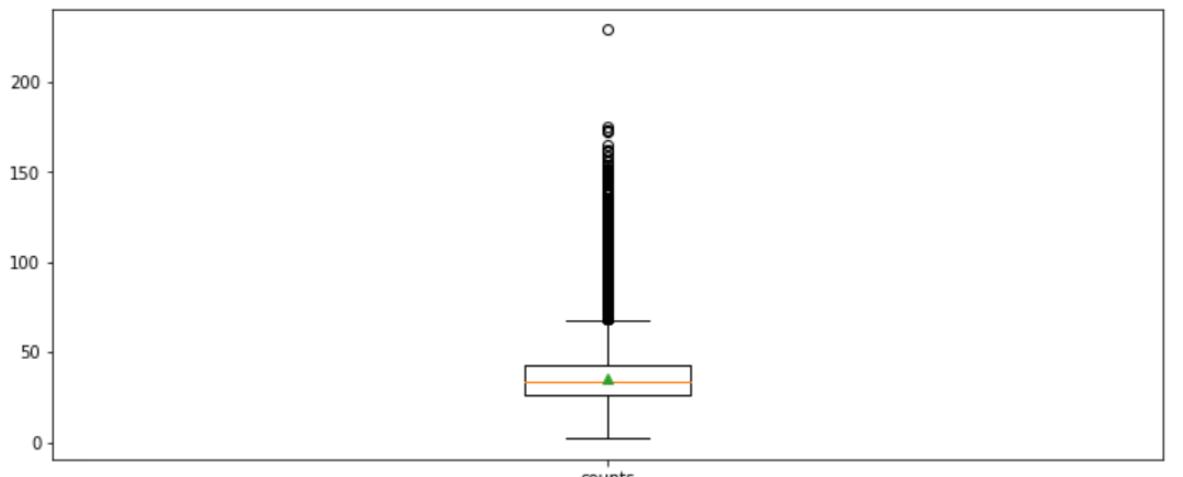
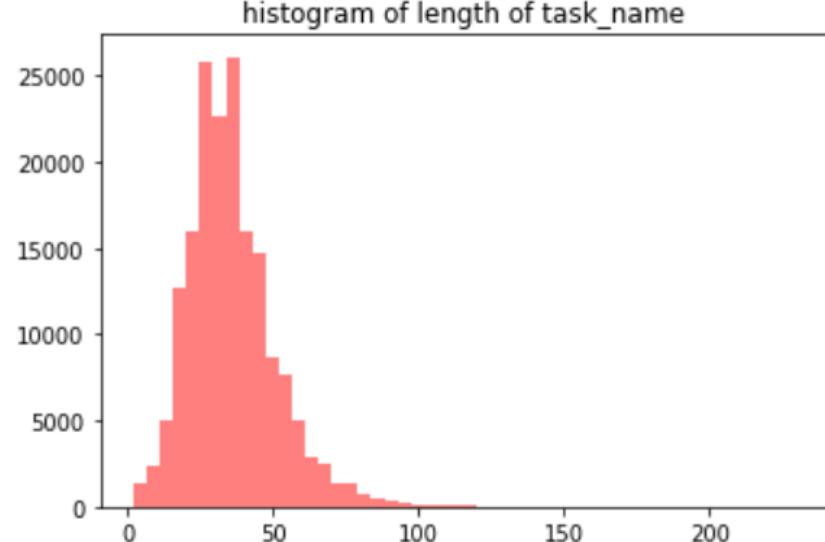
```
length=train['과제명'].astype(str).apply(len)
plt.hist(length, bins=50, alpha=0.5, color='r', label='word')
plt.title('histogram of length of task_name')
plt.figure(figsize=(12, 5))
plt.boxplot(length, labels=['counts'], showmeans=True)
print('과제명 길이 최댓값: {}'.format(np.max(length)))
print('과제명 길이 최솟값: {}'.format(np.min(length)))
print('과제명 길이 평균값: {}'.format(np.mean(length)))
print('과제명 길이 중간값: {}'.format(np.median(length)))
```

과제명 길이 최댓값: 229

과제명 길이 최솟값: 2

과제명 길이 평균값: 35.84252225995961

과제명 길이 중간값: 34.0



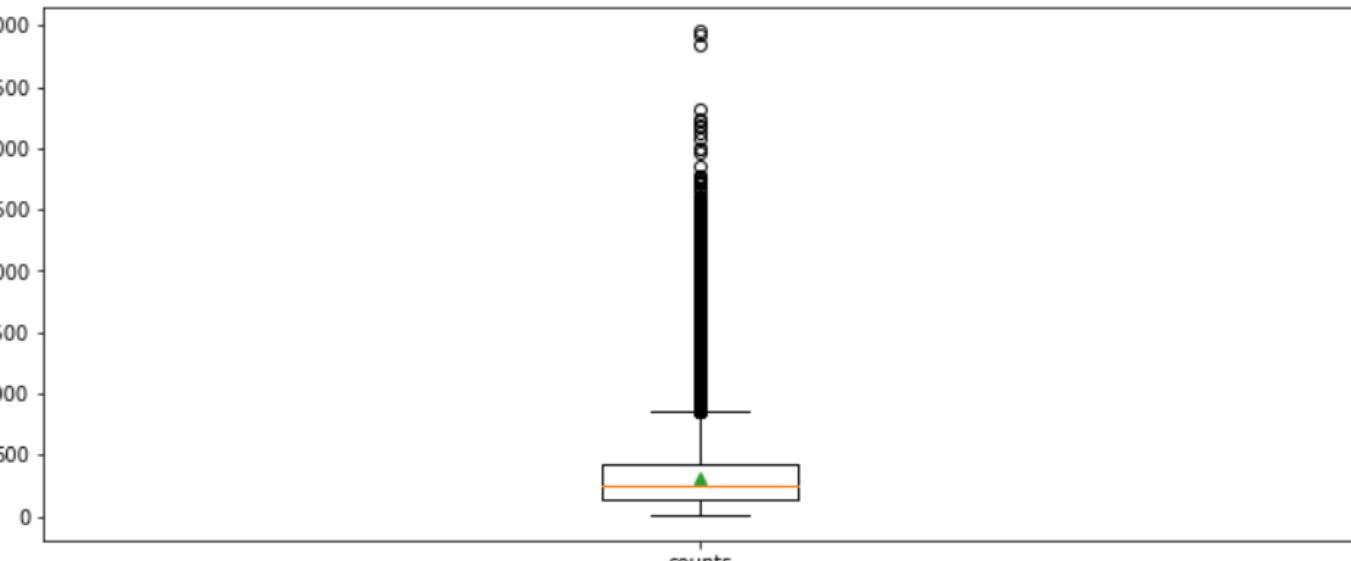
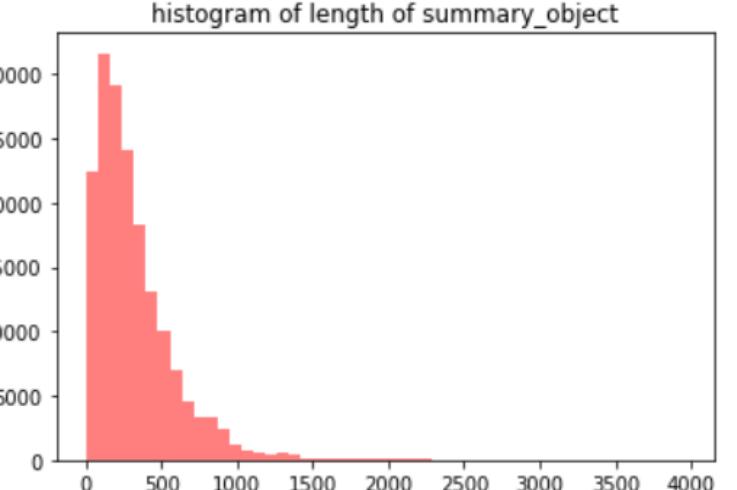
```
length=train['요약문_연구목표'].astype(str).apply(len)
plt.hist(length, bins=50, alpha=0.5, color='r', label='word')
plt.title('histogram of length of summary_object')
plt.figure(figsize=(12, 5))
plt.boxplot(length, labels=['counts'], showmeans=True)
print('요약문_연구목표 길이 최댓값: {}'.format(np.max(length)))
print('요약문_연구목표 길이 최솟값: {}'.format(np.min(length)))
print('요약문_연구목표 길이 평균값: {}'.format(np.mean(length)))
print('요약문_연구목표 길이 중간값: {}'.format(np.median(length)))
```

요약문_연구목표 길이 최댓값: 3951

요약문_연구목표 길이 최솟값: 1

요약문_연구목표 길이 평균값: 318.1008066366807

요약문_연구목표 길이 중간값: 249.0

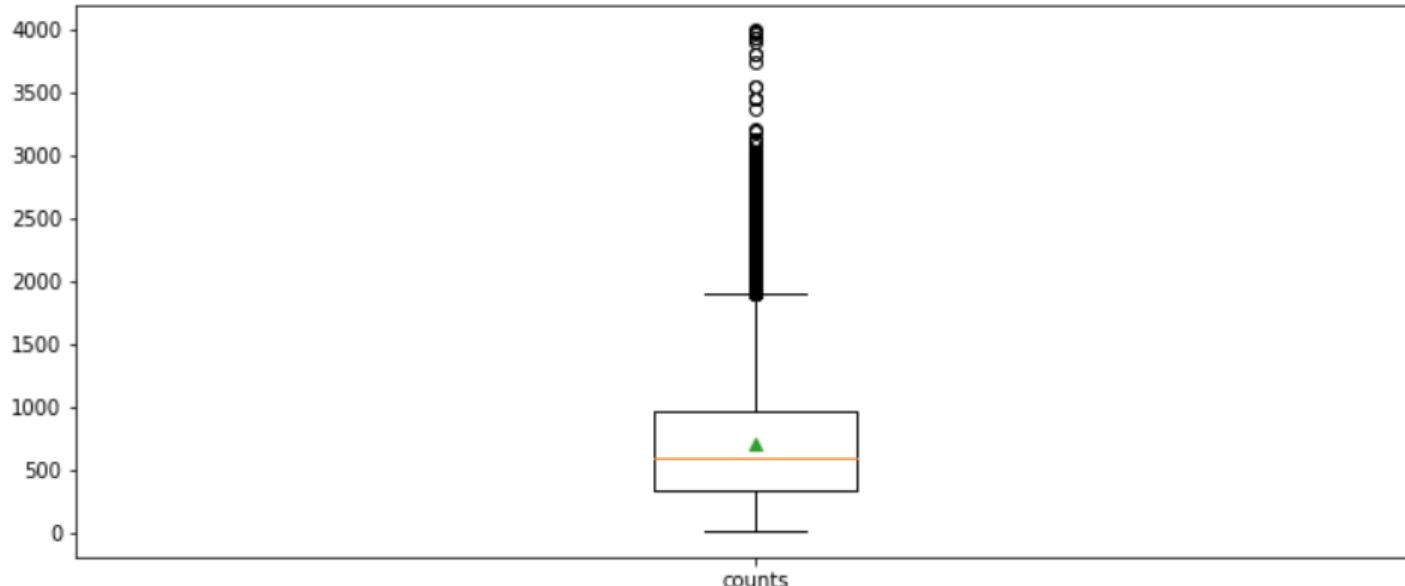
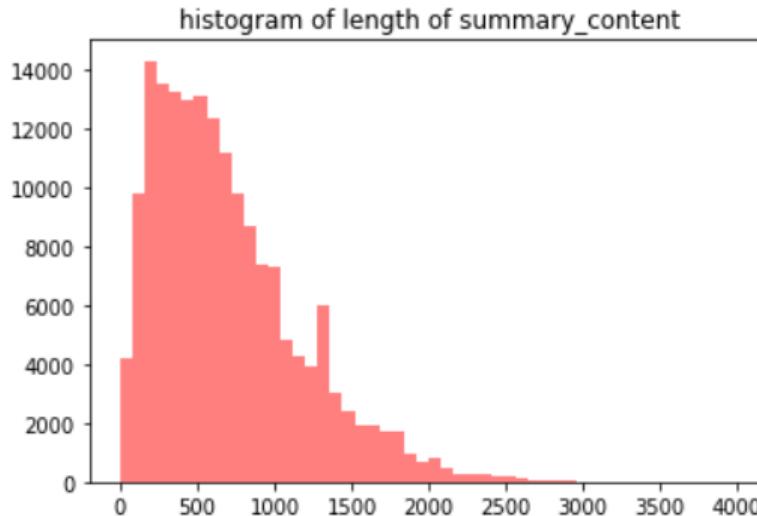


7.2 데이터 EDA

[시각화] 모두 왼쪽으로 치우친 그래프

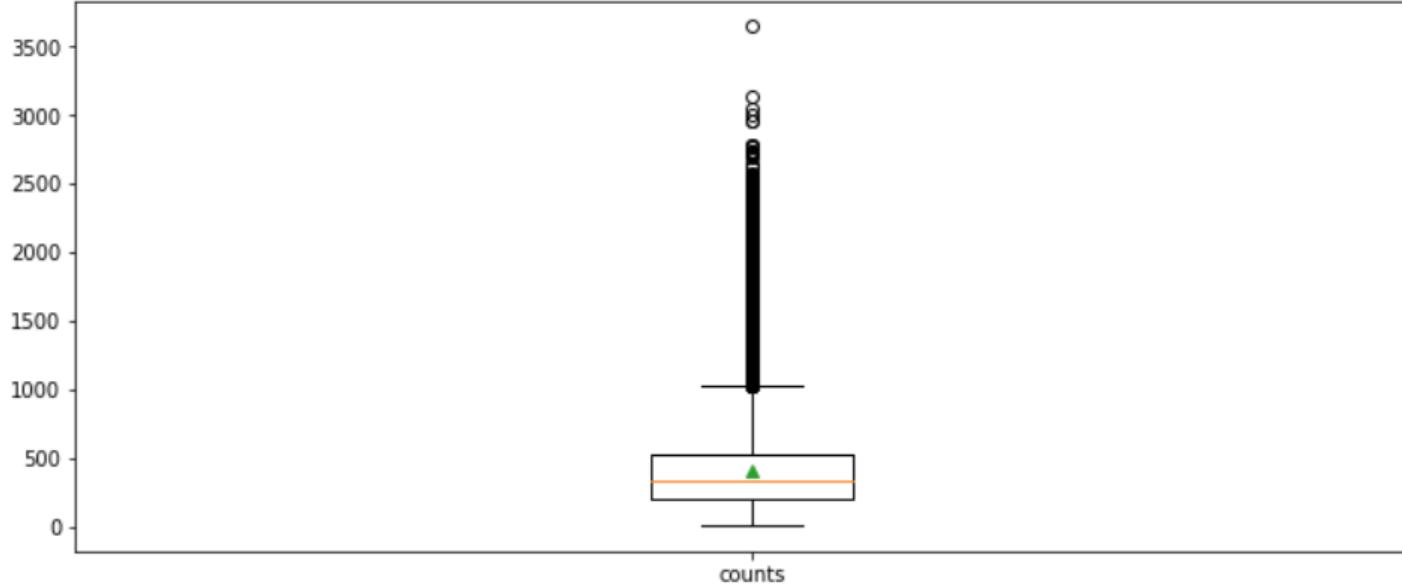
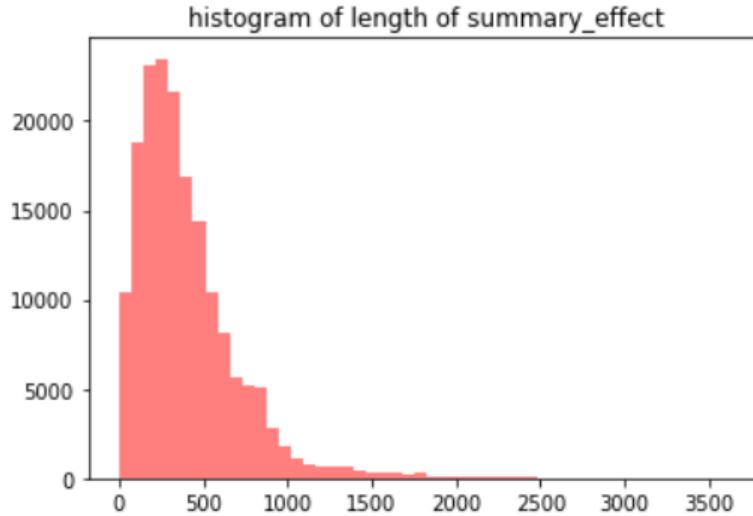
```
length=train['요약문_연구내용'].astype(str).apply(len)
plt.hist(length, bins=50, alpha=0.5, color='r', label='word')
plt.title('histogram of length of summary_content')
plt.figure(figsize=(12, 5))
plt.boxplot(length, labels=['counts'], showmeans=True)
print('요약문_연구내용 길이 최댓값: {}'.format(np.max(length)))
print('요약문_연구내용 길이 최솟값: {}'.format(np.min(length)))
print('요약문_연구내용 길이 평균값: {}'.format(np.mean(length)))
print('요약문_연구내용 길이 중간값: {}'.format(np.median(length)))
```

요약문_연구내용 길이 최댓값: 3999
요약문_연구내용 길이 최솟값: 1
요약문_연구내용 길이 평균값: 699.2930282724435
요약문_연구내용 길이 중간값: 597.0



```
length=train['요약문_기대효과'].astype(str).apply(len)
plt.hist(length, bins=50, alpha=0.5, color='r', label='word')
plt.title('histogram of length of summary_effect')
plt.figure(figsize=(12, 5))
plt.boxplot(length, labels=['counts'], showmeans=True)
print('요약문_기대효과 길이 최댓값: {}'.format(np.max(length)))
print('요약문_기대효과 길이 최솟값: {}'.format(np.min(length)))
print('요약문_기대효과 길이 평균값: {}'.format(np.mean(length)))
print('요약문_기대효과 길이 중간값: {}'.format(np.median(length)))
```

요약문_기대효과 길이 최댓값: 3649
요약문_기대효과 길이 최솟값: 1
요약문_기대효과 길이 평균값: 400.4864374885258
요약문_기대효과 길이 중간값: 329.0



7.3 데이터 전처리

[okt 형태소 분석기]

- okt = Okt() : 객체 생성
- Okt.morphs() : 텍스트를 형태소 단위로 나눔
 - norm : 문장 정규화 옵션
 - stem : 각 단어에서 어간 추출 옵션
- okt.nouns() : 텍스트에서 명사만 추출
- okt.phrases() : 텍스트에서 어절 추출
- okt.pos() : 각 품사를 태깅하는 역할
 - 품사 태깅? 주어진 텍스트를 형태소 단위로 나누고,
나눠진 각 형태소를 그에 해당하는 품사와 함께 리스트화 하는 것

7.3 데이터 전처리

[토큰화 & 스톱워드 제거]

```
def preprocessing(text, okt, remove_stopwords=False, stop_words=[]):
    text=re.sub("[^가-힣ㄱ-ㅎㅏ-ㅣ]","", text)
    word_text=okt.morphs(text, stem=True)
    if remove_stopwords:
        word_review=[token for token in word_text if not token in stop_words]
    return word_review

stop_words=['은','는','이','가','하','아','것','들','의','있','되','수','보','주','등','한']
okt=Okta()
clean_train_text=[]
clean_test_text=[]

for text in tqdm.tqdm(train['과제명']):
    try:
        clean_train_text.append(preprocessing(text, okt, remove_stopwords=True, stop_words=stop_words))
    except:
        clean_train_text.append([])
```

100% |██████████| 174304/174304 [44:25<00:00, 65.39it/s]

```
for text in tqdm.tqdm(test['과제명']):
    if type(text) == str:
        clean_test_text.append(preprocessing(text, okt, remove_stopwords=True, stop_words=stop_words))
    else:
        clean_test_text.append([])
```

100% |██████████| 43576/43576 [12:25<00:00, 58.44it/s]

[카운트 벡터화]

```
from sklearn.feature_extraction.text import CountVectorizer

#tokenizer 인자에는 list를 받아서 그대로 내보내는 함수를 넣어줍니다.
#또한 소문자화를 하지 않도록 설정해야 에러가 나지 않습니다.
vectorizer = CountVectorizer(tokenizer = lambda x: x, lowercase=False)
train_features=vectorizer.fit_transform(clean_train_text)
test_features=vectorizer.transform(clean_test_text)
#test데이터에 fit_transform을 할 경우 data leakage에 해당합니다
```

7.4 모델링

[랜덤포레스트 모델링]

```
#훈련 데이터 셋과 검증 데이터 셋으로 분리  
TEST_SIZE=0.2  
RANDOM_SEED=42  
  
train_x, eval_x, train_y, eval_y=train_test_split(train_features, train['label'], test_size=TEST_SIZE, random_state=RANDOM_SEED)
```

```
#랜덤포레스트로 모델링  
from sklearn.ensemble import RandomForestClassifier  
  
forest=RandomForestClassifier(n_estimators=100)  
  
forest.fit(train_x, train_y)  
  
RandomForestClassifier()
```

```
#모델 검증  
forest.score(eval_x, eval_y)
```

```
0.9208571182696996
```

[예측 및 제출]

```
forest.predict(test_features)  
  
array([0, 0, 0, ..., 2, 0, 0])
```

```
sample_submission['label']=forest.predict(test_features)
```

```
sample_submission
```

	index	label
0	174304	0
1	174305	0
2	174306	0
3	174307	0
4	174308	0
...
43571	217875	0
43572	217876	0
43573	217877	2
43574	217878	0
43575	217879	0

```
43576 rows × 2 columns
```

8. Covid Literature Clustering



8.1 대회 소개

[COVID-19 Literature Clustering]

배경

- 코로나 바이러스 문헌 급증에 따라 문서 접근성을 위한 텍스트 마이닝 도구 개발
- 유사한 연구 기사를 함께 클러스터링하여 관련 출판물 검색 단순화

접근

1. NLP를 사용하여 각 문서 본문의 텍스트 구문 분석
2. TF-IDF를 사용하여 용어 빈도로 각 문서 변환
3. T-SNE를 사용하여 유사한 연구 논문 클러스터링하기 전 차원 축소 적용
4. PCA를 사용하여 predictor의 차수를 노이즈와 이상치를 제거하는 여러 차원으로 투영
5. k-means 군집 분석을 적용하여 군집화 진행
6. 확률적 경사 하강을 사용한 분류를 통해 시각화

8.2 데이터 전처리

[데이터 확인]

```
df_covid.head()
```

paper_id	doi	abstract	body_text	authors	title	journal	abstract
c9d6539c963b68f8f605afaf5a9cff7df1251f7d	10.1016/j.mbs.2020.108441	Since its emergence late in 2019, the COVID-19... • Introduction of a starting time parameter fo...	Garba, Salisu M.. Lubuma, Jean M-S.. Tsanou...	Modeling the transmission dynamics of the ...	Math Biosci	Since its emerger 2019, th 1...	
234abf43ec014de9616407bcd48f671360195905	10.1007/978-3-030-52683-2_11	N-Variant Execution (NVX) systems utilize arti...	Memory errors are a continuous source of softw...	Voulimeneas, Alexios. Song, Dokyung. Par...	Distributed Heterogeneous N-Variant Execution	Detection of Intrusions and Malware, and Vulne...	N-Variar Executio systems utilize ...
07da062617f1f8841c4b7b87bbca7ccb14b05d01	10.1007/978-3-030-55571-9_7	The construction of discourse blaming EU migra...	Eurosceptic inflection often added by London-b...	Rowinski, Paul	Britain First: A Journey into Emotive Rhetoric	Post-Truth, Post-Press, Post-Europe	The con of discou blaming EU r
56df3c03a94c4a16a2b03c6a9d4f7c9dac4bbf4b	10.1007/s10461-020-02822-4	Adolescents and youth living with HIV have poo...	Adolescents and youth, 10 to 24 years of age, ...	Reif, Lindsey K.. Abrams, Elaine J.. Arpadi...	Interventions to Improve Antiretroviral Th...	AIDS Behav	Adolesce youth liv HIV have
4d54f710a365c24cc897968fca43d3672b9acecf	10.3390/ijerph18031067	Housing is an essential component of human lif...	Kim, Jisun. Yoo, Seunghyun	Perceived Health Problems of Young Single-...	Int J Environ Res Public Health	Not prov	

```
df = df_covid.sample(10000, random_state=42)  
del df_covid
```

[결측치 제거]

```
df.dropna(inplace=True)  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 8879 entries, 6113 to 5443  
Data columns (total 8 columns):  
paper_id          8879 non-null object  
doi               8879 non-null object  
abstract          8879 non-null object  
body_text         8879 non-null object  
authors           8879 non-null object  
title             8879 non-null object  
journal            8879 non-null object  
abstract_summary  8879 non-null object  
dtypes: object(8)  
memory usage: 624.3+ KB
```

```
df = df[df['language'] == 'en']  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 8587 entries, 6113 to 5443  
Data columns (total 9 columns):  
paper_id          8587 non-null object  
doi               8587 non-null object  
abstract          8587 non-null object  
body_text         8587 non-null object  
authors           8587 non-null object  
title             8587 non-null object  
journal            8587 non-null object  
abstract_summary  8587 non-null object  
language          8587 non-null object  
dtypes: object(9)  
memory usage: 670.9+ KB
```

8.2 데이터 전처리

[스톱워드 제거]

```
custom_stop_words = [  
    'doi', 'preprint', 'copyright', 'peer', 'reviewed', 'org', 'https', 'et', 'al', 'author', 'figure',  
    'rights', 'reserved', 'permission', 'used', 'using', 'biorxiv', 'medrxiv', 'license', 'fig', 'fig.',  
    'al.', 'Elsevier', 'PMC', 'CZI'  
]  
  
for w in custom_stop_words:  
    if w not in stopwords:  
        stopwords.append(w)
```

[벡터화]

```
from sklearn.feature_extraction.text import TfidfVectorizer  
def vectorize(text, maxx_features):  
  
    vectorizer = TfidfVectorizer(max_features=maxx_features)  
    X = vectorizer.fit_transform(text)  
    return X
```

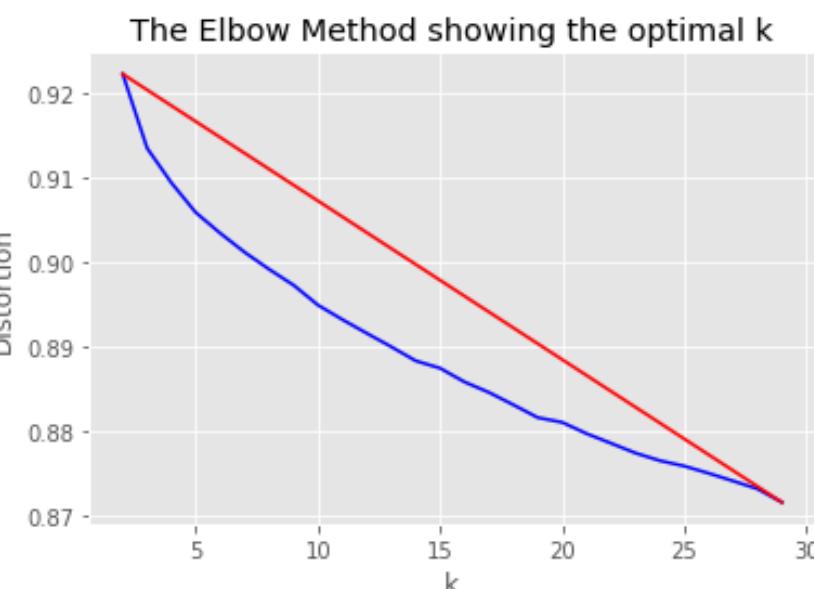
8.3 PCA & 클러스터링

[PCA]

```
from sklearn.decomposition import PCA  
  
pca = PCA(n_components=0.95, random_state=42)  
X_reduced= pca.fit_transform(X.toarray())  
X_reduced.shape  
  
(8587, 2305)
```

[K-means 군집화]

```
from sklearn import metrics  
from scipy.spatial.distance import cdist  
  
# run kmeans with many different k  
distortions = []  
K = range(2, 30)  
for k in K:  
    k_means = KMeans(n_clusters=k, random_state=42).fit(X_reduced)  
    k_means.fit(X_reduced)  
    distortions.append(sum(np.min(cdist(X_reduced, k_means.cluster_centers_, 'euclidean'), axis=1)) / X.shape[0])  
    #print('Found distortion for {} clusters'.format(k))  
  
X_line = [K[0], K[-1]]  
Y_line = [distortions[0], distortions[-1]]  
  
# Plot the elbow  
plt.plot(K, distortions, 'b-')  
plt.plot(X_line, Y_line, 'r')  
plt.xlabel('k')  
plt.ylabel('Distortion')  
plt.title('The Elbow Method showing the optimal k')  
plt.show()
```



```
k = 20  
kmeans = KMeans(n_clusters=k, random_state=42)  
y_pred = kmeans.fit_predict(X_reduced)  
df['y'] = y_pred
```

8.3 PCA & 클러스터링

[t-SNE]

```
from sklearn.manifold import TSNE

tsne = TSNE(verbose=1, perplexity=50) # Changed perplexity from 100 to
# per FAQ
X_embedded = tsne.fit_transform(X.toarray())
```

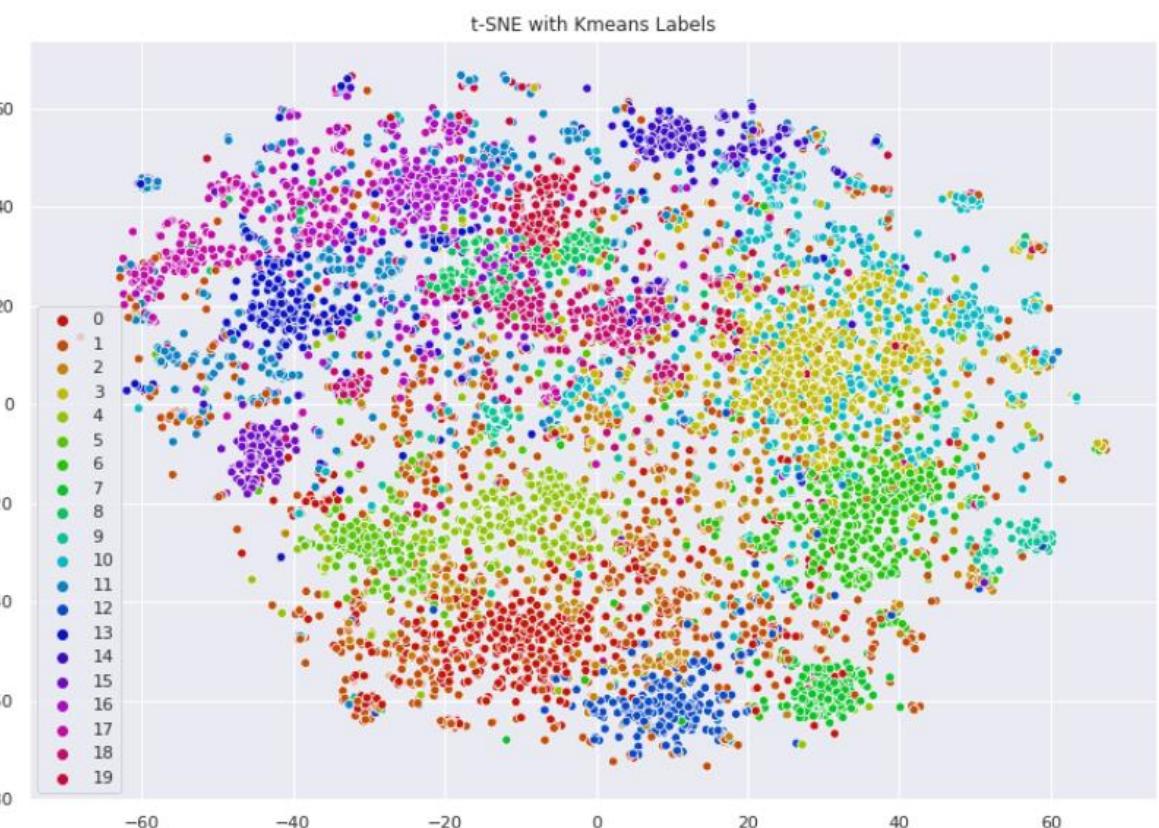
[t-SNE] Computing 151 nearest neighbors...
[t-SNE] Indexed 8587 samples in 14.753s...
[t-SNE] Computed neighbors for 8587 samples in 885.490s...
[t-SNE] Computed conditional probabilities for sample 1000 / 8587
[t-SNE] Computed conditional probabilities for sample 2000 / 8587
[t-SNE] Computed conditional probabilities for sample 3000 / 8587
[t-SNE] Computed conditional probabilities for sample 4000 / 8587
[t-SNE] Computed conditional probabilities for sample 5000 / 8587
[t-SNE] Computed conditional probabilities for sample 6000 / 8587
[t-SNE] Computed conditional probabilities for sample 7000 / 8587
[t-SNE] Computed conditional probabilities for sample 8000 / 8587
[t-SNE] Computed conditional probabilities for sample 8587 / 8587
[t-SNE] Mean sigma: 0.332463
[t-SNE] KL divergence after 250 iterations with early exaggeration: 8
8.342064
[t-SNE] KL divergence after 1000 iterations: 2.103220

```
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns

# sns settings
sns.set(rc={'figure.figsize':(13,9)})

# colors
palette = sns.hls_palette(20, l=.4, s=.9)

# plot
sns.scatterplot(X_embedded[:,0], X_embedded[:,1], hue=y_pred, legend='full',
                 palette=palette)
plt.title('t-SNE with Kmeans Labels')
plt.savefig("improved_cluster_tsne.png")
plt.show()
```



8.4 토픽 모델링

[토픽 모델링]

```
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import CountVectorizer

vectorizers = []

for ii in range(0, 20):
    # Creating a vectorizer
    vectorizers.append(CountVectorizer(min_df=5, max_df=0.9, stop_words
='english', lowercase=True, token_pattern='[a-zA-Z\-\'][a-zA-Z\-\']{2,}'))

vectorized_data = []

for current_cluster, cvec in enumerate(vectorizers):
    try:
        vectorized_data.append(cvec.fit_transform(df.loc[df['y'] == current_cluster, 'processed_text']))
    except Exception as e:
        print("Not enough instances in cluster: " + str(current_cluster))
        vectorized_data.append(None)

clusters_lda_data = []

for current_cluster, lda in enumerate(lda_models):
    print("Current Cluster: " + str(current_cluster))

    if vectorized_data[current_cluster] != None:
        clusters_lda_data.append((lda.fit_transform(vectorized_data[current_cluster])))
```

```
all_keywords[0][:10]
```

```
['disease',
 'covid-',
 'research',
 'food',
 'community',
 'datum',
 'company',
 'case',
 'social',
 'technology']
```

THANK YOU

