

Text Mining with Youtube

유튜브 크리에이터 비교 및 분석

정은지

Contents

01 주제 소개 및 전처리

- 분석할 데이터 및 목표
- 전처리 과정

02 데이터 시각화

- 제목 및 댓글 분석 및 시각화
- 워드 클라우드 등

03 Machine Learning

- 1) NMF(Non-negative Matrix Factorization)
 - 2) Logistic Regression, Linear SVC, Random Forest, KNN, Naïve Bayes, Perceptron, SGDClassifier, DecisionTree, SVM .. **이건 상황보고..**
- 최종 Score 정렬 및 파일 생성

1-1. 분석할 데이터 및 목표 - 1

유튜브의 인기채널인 '펭수'와 '워크맨'을 비교해봄으로써 최근의 트렌드 변화와 인기에 영향을 미치는 요인에 대해 분석

* 목표: 어떤 영상이 가장 인기가 많은지, 왜 인기가 많은지 게시글 및 댓글 분석을 통해 알아보기

페

하

“펭수-하하”

교육방송 역사 D+137

5DC 치트 여스새 펭수의



2주 전

선 넘는 인간 : 장성규

선 넘는 펭귄 : 펭수

 1.6천

 답글



워크맨-Workman

구독자 353만명

홈

동영상

재생목록

커뮤니티

채널

정보



1-2. 분석할 데이터 및 목표 - 2

1) DataSet: 팽수 및 워크맨 동영상 및 동영상 별 댓글

```
intro_paw.describe()
```

	Unnamed: 0	영상수
count	2.000000	2.000000
mean	1.500000	84.000000
std	0.707107	69.296465
min	1.000000	35.000000
25%	1.250000	59.500000
50%	1.500000	84.000000
75%	1.750000	108.500000
max	2.000000	133.000000

```
paw.describe()
```

	Unnamed: 0	싫어요
count	158.000000	158.000000
mean	51.253165	467.43038
std	36.693070	685.10947
min	0.000000	8.000000
25%	19.250000	70.000000
50%	43.500000	163.500000
75%	80.750000	404.750000
max	157.000000	157.000000

```
paw_reply.describe() # 통합한 동영상별 댓글 list DF 확인
```

	Unnamed: 0	Like	Title_number
count	3160.000000	3160.000000	3160.0
mean	9.500000	1430.986392	0.0
std	5.767194	2652.356452	0.0
min	0.000000	0.000000	0.0
25%	4.750000	193.000000	0.0
50%	9.500000	546.500000	0.0
75%	14.250000	1500.000000	0.0
max	19.000000	38000.000000	0.0

2) Column Count

- Intro_paw: 2개 # 비교 채널 수
- Paw: 168개 # 영상 수
- Paw_reply: 3160개 # 영상에서 추출한 댓글 수

실제 댓글은 1영상 당 약 5천개가 존재하나,
부하로 인해 동영상당 20개 한정(동영상158개)으로 수집함.
* 코드상 pagedowns 숫자만 변경하면 전체 댓글 수집 가능

1-2. 데이터 전처리 과정:

1) 패키지 import -1 (data처리용 - pandas, crawling, konlpy 등)

```
# 펭수 패키지 import
import datetime as dt
import pandas as pd
import requests
import time
import urllib.request #
import re
import konlpy

from bs4 import BeautifulSoup
from pandas import DataFrame
from selenium.webdriver import Chrome
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
```

```
#워크맨 패키지 import
import datetime as dt
import pandas as pd
import requests
import time
import urllib.request #
import re
import konlpy

from bs4 import BeautifulSoup
from pandas import DataFrame
from selenium.webdriver import Chrome
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
```

펭수와 워크맨 코드 개별 구현, 전처리 단계에서 병합 예정.
(사용한 패키지는 두 파일 모두 동일)

-
- └ Konlpy : 한국어 분석을 위한 패키지
형태소 분석, 명사 분석 등 한국어에 특화된 분석 가능
 - └ selenium : 웹 브라우저를 제어하는 패키지
유튜브 화면 클릭, 스크롤 제어 등에 사용

1-2. 데이터 전처리 과정:

1) 패키지 import -2 (시각화 및 ML)

```
# 시각화 패키지
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image #워드클라우드 패키지
from wordcloud import WordCloud, STOPWORDS #워드클라우드 패키지
fpath = "NotoSansCJKkr-Bold.otf" # 국문 지원을 위한 폰트 path
%matplotlib inline

# Machine Learning 패키지
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
```

추출한 데이터를 이용하여 시각화 및 기계 학습을 위한 패키지 import

- * 시각화: wordcloud 및 matplotlib 사용

- * Machine Learning Model:

LogisticRegression, Linear SVC, Random Forest, KNN,

Naïve Bayes, Perceptron, SGDClassifier, DecisionTree, SVM

1-2. 데이터 전처리 과정:

2) 데이터 입력 (Chrome Driver, Crawling) - 공통

```
#크롬 드라이버 연결
delay=0.1
browser = Chrome()
browser.implicitly_wait(delay)

start_url = 'https://www.youtube.com/channel/UCtckgmUcpzqGnzcs7xEqMzQ/videos'
browser.get(start_url) #browser로 위의 url 실행(펍수 videos directory)
browser.maximize_window()

body = browser.find_element_by_tag_name('body') #스크롤 위한 소스 추출
num_of_pagedowns = 20 # 페이지 down 수

#스크롤 다운
while num_of_pagedowns:
    body.send_keys(Keys.PAGE_DOWN)
    time.sleep(0.1)
    num_of_pagedowns -= 1
```

Chrome Driver를 통해 별도의 창을 Open 후, 해당 창에서 data를 crawling 진행함
스크롤 다운이 필요한 페이지므로 스크롤 제어를 사용함

1-2. 데이터 전처리 과정:

2) 데이터 입력 (Chrome Driver, Crawling) - 공통

```
# 페이지 소스 받아오기
html0 = browser.page_source
html = BeautifulSoup(html0, 'html.parser')
```

```
# 동영상 directory에 있는 각 영상들의 key값 생성을 위한 리스트 선언
# title(제목), href(링크), viewcount(조회수)
```

```
title_list = [] #제목 리스트 생성
href_list = [] #주소 리스트 생성
viewcountmake_list = [] #조회수 크롤링 위한 리스트 생성
viewcount_list = [] #조회수 리스트 생성
```

```
# 구독자 수 저장
subsc = html.find(id="subscriber-count").text
```

```
# title 저장
for tr in html.find_all(id="video-title"):
    title = tr.get('title')
    title_list.append(title)
```

```
#href 태그 내용 저장
for tr in html.find_all(id="video-title"):
    href = tr.get('href')
    href_list.append(href)
```

```
# 조회수 저장
for tr in html.find_all('span', class_="style-scope ytd-grid-video-renderer"):
    viewcount = tr.get_text('span')
    viewcountmake_list.append(viewcount)
```

```
#구독자수, 동영상 수
print(subsc, ", 영상 개수:", len(title_list))
```

구독자 133만명 , 영상 개수: 133

1-2. 데이터 전처리 과정:

2) 데이터 입력 (Chrome Driver, Crawling) - 공통

```
# 각 array별 개수 확인
print(len(title_list), len(href_list), len(viewcountmake_list))

#title, 주소, 좋아요수 데이터프레임 생성
peng_list = pd.DataFrame({'title':title_list, 'href':href_list, 'viewcount':viewcountmake_list})
```

구독자 133만명 , 영상 개수: 133

133 133 266

```
c:\users\백송0\appdata\local\programs\python\python36\lib\site-packages\pandas\core\internals
#construction.py in extract_index(data)
    315         lengths = list(set(raw_lengths))
    316         if len(lengths) > 1:
--> 317             raise ValueError('arrays must all be same length')
    318
    319         if have_dicts:
```

ValueError: arrays must all be same length

DF 생성 시, 조회수(viewcount)는 조회수/등록일이 Set로 포함되어
arrays length 불일치(title_list:133, href_list:133, viewcountmake_list:266)로 인해
에러 발생. 따라서, 불필요한 데이터인 '등록일' 제거하는 전처리 진행

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (데이터 클리닝 작업) - 공통

```
# 데이터 클리닝 작업(조회수과 등록일에서 등록일 제거)
```

```
for tr in range(0, len(viewcountmake_list), 2):
```

```
    a = viewcountmake_list[tr]
```

```
    viewcount_list.append(a)
```

```
# 조회수 데이터 전처리
```

```
clean_viewcount = []
```

```
for i in viewcount_list:
```

```
    a = i[4:-2]
```

```
    amul = float(a)*10000
```

```
    clean_viewcount.append(int(amul))
```

```
print(clean_viewcount)
```

• 데이터 클리닝 작업

1) 등록일 제거

- 기존: X개월 전 -> 삭제

2) 조회수 데이터 전처리 작업

- 기존: '조회수 86회' -> '860000'

```
[860000, 1120000, 610000, 1480000, 660000, 1130000, 1350000, 890000, 1300000, 10
10000, 1370000, 1550000, 600000, 1940000, 640000, 1400000, 1390000, 880000, 7400
00, 1040000, 680000, 320000, 1430000, 580000, 900000, 800000, 710000, 410000, 22
30000, 1170000, 180000, 1530000, 200000, 510000, 320000, 800000, 530000, 810000,
430000, 1730000, 480000, 540000, 560000, 350000, 3150000, 300000, 230000, 66000
0, 340000, 480000, 300000, 540000, 1360000, 100000, 2100000, 1550000, 380000, 84
0000, 1230000, 770000, 630000, 600000, 360000, 280000, 890000, 500000, 430000, 3
20000, 630000, 1380000, 290000, 300000, 330000, 92000, 1070000, 360000, 440000,
220000, 1520000, 170000, 290000, 580000, 180000, 230000, 96000, 790000, 1500000,
380000, 830000, 720000, 660000, 99000, 500000, 820000, 360000, 230000, 440000, 4
70000, 510000, 150000, 130000, 210000, 650000, 210000, 550000, 190000, 100000, 8
20000, 910000, 1020000, 930000, 230000, 490000, 680000, 680000, 480000, 930000,
220000, 390000, 330000, 470000, 550000, 430000, 280000, 300000, 290000, 460000,
480000, 590000, 960000, 380000, 460000, 610000]
```

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (df 생성 및 확인) - 공통

#title, 주소, 좋아요수 데이터프레임 생성

```
peng_list = pd.DataFrame({'title':title_list, 'href':href_list, 'viewcount':clean_viewcount})  
peng_list.sort_values(by=['viewcount'], axis=0, ascending=False)
```

	title	href	viewcount
44	EBS 최초 연습생 펑수의 오디션 합격 TIP *최초공개*	/watch?v=K_5lal40ICk	3150000
28	[단독] 펑권 의혹 전격 해부! 독점 인터뷰: 김민교, 양치승 [Ep.57]	/watch?v=UD-WQvgjsng	2230000
54	[Ep.44] 펑수, 드디어 크로스를 만났다	/watch?v=TckHOovKE2I	2100000
13	[Ep.65] 세상에 나쁜 펑권은 없다.	/watch?v=wedLGh2jxkQ	1940000
39	EBS 옥상에서 똑딱이 선배님을 만났다 (feat. 역대급 깜짝손님)	/watch?v=yN7wrzXtIWM	1730000
11	[Ep.66] 화보 모델 펑수	/watch?v=XUM3sH1kBtw	1560000
55	[Ep.43]EBS 복지 클래스 전격 공개! [힐링 선물 3종세트]	/watch?v=80cyxKBcYXA	1550000
31	[Ep.56] '펑TV 야유회'라고 쓰고 '지옥'이라 읽는다	/watch?v=pv84Qcu9DOQ	1530000
78	'자꾸 교육방송 선 넘는' 이말년(침착맨)과 펑수(EBS 연습생)의 짝방 폭격! [...	/watch?v=S9q4BorGgOI	1520000
86	예술천재 펑수, 고양예고 테스트 도전! 음악과 미술과 뿌셔! [EP.27]	/watch?v=NjS52IUjH DU	1500000
3	[Ep.71] 내 헤드셋 누가 가져갔어?	/watch?v=rRZtVrh6kgl	1480000
22	[Ep.60] 펑수, EBS 퇴사??? SBS 정복기(2)	/watch?v=ze1wbV20qAU	1430000
15	1회 남극 유치원 동창회 시작합니다(Feat. 돌리 선배님 등장)	/watch?v=E5f4pRlycsU	1400000
16			
69	(납량특집) EBS 펑권		
10			

- 가장 인기있는 동영상 찾기 위해
동영상별 df를 생성 후 조회수(viewcount)기준 정렬

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (1. 동영상 별 속성값 list추출) - 공통

```
# 영상 본문 key(제목, 조회수, 좋아요, 싫어요, 본문내용, 전체 댓글 수) 추출 - 1

title_num = 0;
n_title_list = [] # 영상별 제목 리스트 생성
story_sub_list = [] # 영상 본문내용 리스트
n_viewcount_list = [] #영상별 조회수 리스트 생성
n_like_list = [] #영상별 좋아요 리스트 생성
n_unlike_list = [] #영상별 싫어요 리스트 생성
n_href_list = [] # 영상별 주소 리스트 생성
str_n_comments = [] # 영상별 댓글 총 수 리스트 생성
start_url = 'https://www.youtube.com/'

# 데이터
for i in href_list[:2]: # :2 빼면 전체 데이터, 시간 단축을 위해 2개 샘플만 시행
    url = start_url + i
    browser.get(url)
    print(i) # 추출 주소값
    n_href_list.append(i)

    source = browser.page_source
    bs = BeautifulSoup(source, 'html.parser')
    bsstory = BeautifulSoup(source, "html.parser")
```

영상별 '제목, 본문, 조회수, 좋아요, 싫어요, 댓글수' 가 포함된 DataSet 생성 예정
(펍수 동영상 디렉토리에 있는 모든 동영상을 조회하여 정보 추출)

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (1. 동영상 별 속성값 list 추출) - 공통

```
# 영상에 대한 기본 정보 수집
info1 = bs.find('div',{ 'id':'info-contents'})

#n_title = "" #초기화 계속
n_title = info1.find('h1',{ 'class':'title style-scope ytd-video-primary-info-renderer'}).text
n_title_list.append(n_title)
n_viewcount = info1.find('yt-view-count-renderer',{ 'class':'style-scope ytd-video-primary-info-renderer'}).find_all('span')[0].text
n_viewcount_list.append(n_viewcount)
n_like = info1.find('div',{ 'id':'top-level-buttons'}).find_all('yt-formatted-string')[0].text #좋아요수
n_like_list.append(n_like)
n_unlike = info1.find('div',{ 'id':'top-level-buttons'}).find_all('yt-formatted-string')[1].text
n_unlike_list.append(n_unlike)

# 영상 본문 내용 추출
story_sub = bsstory.find('yt-formatted-string', class_='content style-scope ytd-video-secondary-info-renderer').text
story_sub_list.append(story_sub)
title_num = title_num + 1; # 동영상 번호

# 영상별 댓글수 추출

for i in range(len(n_comments)):
    str_tmp = str(n_comments[i].text)
    str_tmp = str_tmp.replace('댓글 ', '')
    str_tmp = str_tmp.replace('개', '')
    str_n_comments.append(str_tmp)
```

각 영상마다 ‘제목, 본문, 조회수, 좋아요, 싫어요, 댓글수’를 추출하여
각 항목(title_list, viewcount_list, like_list, unlike_list, story_sub_list, str_n_comments)의 list에 추가

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (1. 동영상 별 속성값 list 추출) - 공통

조회수 데이터 전처리

```
clean_n_viewcount = []
for i in n_viewcount_list:
    a = i[4:-1]
    clean_n_viewcount.append(a)
print(clean_n_viewcount)
```

좋아요 데이터 전처리

```
clean_n_like_list = []
for i in n_like_list:
    if i[-1] == '천' or '만':
        a = i[:-1]
        if i[-1] == '천': amul = float(a)*1000
        if i[-1] == '만': amul = float(a)*10000
        clean_n_like_list.append(int(amul))
    else:
        clean_n_like_list.append(i)
print(clean_n_like_list)
```

싫어요 데이터 전처리

```
clean_n_unlike_list = []
for i in n_unlike_list:
    clean_n_unlike_list.append(i)
print(clean_n_unlike_list)
```

```
['1,374,702', '1,562,005', '600,930', '1,944,586', '648,478', '1,409,749',
'1,399,059', '887,330', '745,696', '1,048,624', '685,071', '329,770', '1,440,
539', '585,841', '906,534', '805,053', '719,986', '416,550', '2,239,768', '1,
180,868', '184,352', '1,541,343', '204,786', '517,546', '325,203', '801,839',
'540,834', '816,829', '430,905', '1,732,931', '485,258', '549,376', '568,06
4', '354,701', '3,159,281', '310,126', '232,295', '668,482', '347,716', '482,
736', '305,721', '541,394', '1,366,438', '101,847', '2,109,712', '1,556,128',
'384,418', '845,470', '1,236,459', '780,503', '633,499', '605,853', '364,52
9', '281,998', '900,953', '506,483', '433,951', '325,786', '631,440', '1,392,
969', '300,465', '305,365', '332,093', '92,308', '1,078,602', '366,121', '44
4,650', '229,006', '1,530,687', '171,710', '295,701', '589,027', '188,131',
'230,547', '96,789', '799,954', '1,506,724', '388,573', '832,745', '727,670',
'671,355', '99,810', '510,981', '830,523', '361,678', '230,559', '449,038',
'472,802', '514,906', '151,038', '130,488', '211,608', '655,297', '211,119',
'552,914', '196,212', '107,366', '830,121', '919,272', '1,021,378', '938,69
0', '233,913', '499,892', '681,557', '689,878', '487,521', '934,838', '226,65
1', '393,587', '334,374', '471,266', '555,697', '438,033', '286,852', '304,25
2', '300,175', '461,337', '481,318', '593,577', '964,559', '386,790', '463,77
4', '611,418']
[41000, 45000, 28000, 50000, 20000, 37000, 38000, 24000, 27000, 35000, 21000,
13000, 26000, 21000, 21000, 25000, 17000, 14000, 44000, 30000, 6500, 28000, 6
200, 13000, 12000, 25000, 13000, 19000, 14000, 36000, 12000, 14000, 13000, 90
00, 49000, 7800, 6200, 18000, 10000, 14000, 9600, 12000, 23000, 3300, 35000,
28000, 17000, 14000, 18000, 21000, 18000, 17000, 8300, 8200, 18000, 13000, 94
00, 9600, 12000, 23000, 12000, 24000, 7900, 3300, 15000, 7500, 23000, 5500, 2
2000, 9700, 9100, 11000, 3300, 4400, 2800, 15000, 24000, 10000, 18000, 23000,
15000, 2300, 8500, 13000, 6500, 6700, 12000, 11000, 21000, 3400, 4400, 6700,
21000, 5700, 13000, 5800, 4000, 12000, 18000, 19000, 18000, 7600, 12000, 1600
0, 12000, 7500, 14000, 5400, 11000, 8700, 11000, 12000, 9300, 4900, 6900, 700
0, 10000, 18000, 14000, 22000, 8100, 8700, 12000]
['748', '453', '133', '539', '135', '421', '373', '193', '130', '275', '126',
'57', '347', '125', '242', '157', '159', '84', '779', '261', '23', '387', '2
5', '128', '47', '136', '93', '178', '68', '360', '83', '102', '137', '39',
'808', '45', '23', '94', '62', '119', '55', '227', '294', '27', '442', '374',
'40', '131', '462', '132', '165', '148', '67', '51', '120', '112', '102', '14
```

조회수/좋아요/싫어요의 경우 '조회수 X회', 'X.X만(ex. 3.6만)' 으로 추출되어,
이를 'X', 텍스트 삭제 후, 정수형(ex. 36000)으로 전처리 진행

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (1. 동영상 별 속성값 list 추출) - 공통

```
n_peng_list = pd.DataFrame({'제목':n_title_list, '주소':n_href_list, '조회수':clean_n_viewcount,  
                             '좋아요':clean_n_like_list, '싫어요':clean_n_unlike_list, '댓글수':str_n_comments,  
                             '본문내용':story_sub_list})
```

```
n_peng_list.head()
```

	제목	주소	조회수	좋아 요	싫어 요	댓글 수	본문내용
0	[Ep.73] 10살 펭귄 벌써 집 장만	/watch?v=0jQ6W6BPUFc	1,102,406	37000	403	5,730	드디어 소품실을 탈출한 펭수, 새로운 펭숙소에서 집들이를 하는데... 과연 어떤 모...
1	영화 '천문: 하늘에 묻는다' 오디션 보러 간 펭수(* 쿠기영상 있음!)	/watch?v=8qYmLV0IYXI	1,175,032	38000	437	5,166	첫 영화 오디션을 본 펭수\n라면 먹고 돌아오다?\n\n[MUSIC INFO]...

각 영상에 대한 속성값의 DataFrame 생성 후 head를 통해 생성 값 확인

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (2. 각 영상 별 댓글list 추출) - 공통

```
# 영상별 댓글 추출
title_num = 0;
youtube_pd = pd.DataFrame() # 영상 당 댓글 df 생성용
total_youtube_pd = pd.DataFrame() # 최종 df 생성용
str_youtube_comments_len = [] # 영상별 댓글수 리스트 생성

start_url = 'https://www.youtube.com/'
for i in href_list[:2]: # :2 빼면 전체 데이터, 시간 단축을 위해 2개 샘플만 시행
    url = start_url + i
    browser.get(url)
    print(i) # 추출 주소값
    source = browser.page_source
    bs = BeautifulSoup(source, "html.parser")
```

동영상 별 상위 댓글 20개씩을 추출하여 df 생성 예정.

- 댓글 20개 * 131개 동영상 = 2620개의 행(Rows)

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (2. 각 영상 별 댓글list 추출) - 공통

```
# 댓글 추출 코드
reply_list = [] #댓글 리스트 생성
body = browser.find_element_by_tag_name('body') #스크롤하기 위해 소스 추출
num_of_pagedowns = 20

#스크롤 다운
while num_of_pagedowns:
    body.send_keys(Keys.PAGE_DOWN)
    num_of_pagedowns -= 1

    html0 = browser.page_source
    html = BeautifulSoup(html0, 'html.parser')

    youtube_user_IDs = html.select('div#header-author > a > span')
    youtube_comments = html.select('yt-formatted-string#content-text')
    youtube_likes = html.select('div#toolbar > span')

    str_youtube_userIDs = []
    str_youtube_comments = []
    str_youtube_likes = []
    str_youtube_title_number = [] #영상 타이틀 넘버
```

각 영상의 댓글 추출을 위한 pagedown 셋팅 및
댓글의 속성값(User_ID, comments, likes) 생성

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (2. 각 영상 별 댓글list 추출) - 공통

```
for i in range(len(youtube_user_ids)):
    str_tmp = str(youtube_user_ids[i].text)
    str_tmp = str_tmp.replace('\\\\n', '')
    str_tmp = str_tmp.replace('\\\\t', '')
    str_tmp = str_tmp.replace('\\\\', '')
    str_youtube_user_ids.append(str_tmp)

    str_tmp = str(youtube_comments[i].text)
    str_tmp = str_tmp.replace('\\\\n', '')
    str_tmp = str_tmp.replace('\\\\t', '')
    str_tmp = str_tmp.replace('\\\\', '')
    str_youtube_comments.append(str_tmp)

    str_tmp = str(youtube_likes[i].text)
    str_tmp = str_tmp.replace('\\\\n', '')
    str_tmp = str_tmp.replace('\\\\t', '')
    str_tmp = str_tmp.replace('\\\\', '')
    str_youtube_likes.append(str_tmp)

    # 어떤 영상인지 넘버링으로 확인
    str_youtube_title_number.append(title_num)
```

User Id, comments, like에 포함된 공백, 탭, 과도한 띄어쓰기 제거(replace)

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (2. 각 영상 별 댓글list 추출) - 공통

```
# 좋아요 수 전처리, 좋아요수가 str+공백으로 들어가 있어 공백 제거 후 계산
clean_str_youtube_likes = []
for i in str_youtube_likes:
    i = i.replace(" ", '')
    if i.find('만') != -1:
        a = i.replace('만', '')
        amul = float(a)*10000
        clean_str_youtube_likes.append(int(amul))
    elif i.find('천') != -1:
        a = i.replace('천', '')
        amul = float(a)*1000
        clean_str_youtube_likes.append(int(amul))
    else:
        clean_str_youtube_likes.append(int(i))

# 각 영상별 댓글 데이터 프레임 생성
youtube_pd = pd.DataFrame({"ID":str_youtube_userIDs, "Comment":str_youtube_comments,
                           "Like":clean_str_youtube_likes, "Title_number":str_youtube_title_number})
total_youtube_pd = pd.concat([total_youtube_pd, youtube_pd])
print(len(youtube_pd))
```

* 좋아요 수 전처리

좋아요 수에 str+공백이 포함되어 공백 제거 후 계산 (ex. 3.6만 → 36000)

* 댓글 데이터 프레임 생성

유저ID, 댓글 본문, 좋아요 수, 영상 번호 로 구성.

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (2. 각 영상 별 댓글list 추출) - 공통

```
# 게시글당 댓글 추출개수를 구하고, 이를 모두 합해 전체 동영상의 댓글 전체 개수를 구함.
print("* 본문 추출 개수: ", len(story_sub_list)) #추출한 본문 개수 확인
print("* 댓글 추출 개수: ", len(total_youtube_pd)) #추출한 댓글 개수 확인

/watch?v=U34Z1Ky11CU
20

/watch?v=2gg1QKk7fjw
20

/watch?v=M-52pdhp-ao
20

/watch?v=Er016Y9sE8Q
20

/watch?v=4vkzT47_Vdg
20

/watch?v=XeH_8w4AWS4
20

* 본문 추출 개수: 123
* 댓글 추출 개수: 2460
```

* 본문 및 댓글 추출 개수 확인

123개의 동영상을 확인하였으며, 각 영상 별 상위 20개의 댓글을 추출함.

123개 * 20개 = 2460개 댓글을 추출하였음.

이는 상단의 pagedowns 값을 조절하여 영상 별 댓글 추출 수를 늘릴 수 있으나,
컴퓨터 부하 및 엄청난 시간 소요로 인해 금번 과제에서는 pagedowns=10으로 설정.

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (2. 각 영상 별 댓글list 추출) - 공통

```
print(str_youtube_comments_len)
```

[illegible]

#추출한 본문 예시 1개 확인

```
print("예시 1개", print(story_sub_list[1]))
```

평수가 화보 모델로서 촬영을 하게 되었다!
 화보 촬영을 위한 평수의 눈물(?) 나는 노력!
 그리고 현직 모델에게 직접 받는 모델 포즈까지!
 예시 1개 None

* 실제 추출된 값 확인

각 본문 당 20개의 댓글을 추출하였으며, 영상 본문 또한 정상 추출되었음.

1-2. 데이터 전처리 과정:


2) 데이터 입력 및 처리 (본문내용 데이터 전처리) - 공통

```
# 본문내용 데이터 전처리
clean_story_sub_list = []
for i in story_sub_list:
    a = i.replace('\n', '')
    # 첨부 음악 리스트가 본문내용에 포함된 경우 찾아서 삭제
    delresult = a.find("[MUSIC INFO]")
    delresult1 = a.find("음원 정보")
    delresult2 = a.find("1.")
    if delresult != -1:
        a = a[:delresult]
        clean_story_sub_list.append(a)
    elif delresult1 != -1:
        a = a[:delresult1]
        clean_story_sub_list.append(a)
    elif delresult2 != -1:
        a = a[:delresult2]
        clean_story_sub_list.append(a)
    else:
        clean_story_sub_list.append(a)
print(clean_story_sub_list)
```

['K-펍권 한다 해외진출', '펍수가 화보 모델로서 촬영을 하게 직접 받는 모델 포즈까지!', '힙합펍수의 커버영상이 드. 디. 이번 신상뮤비 공개챌린지에 참여해준 모든 분들 펍러뷰~♡잇기 두', "어느 날 갑자기 펍수에게 이상한 행동들이 보인다!? 걱정 ===== 칠판연 '꿈을 이루기 위한 7가지 충' 을 앞둔여러분들을 위해 이 영상을 바칩니다 펍펍', '펍수가 워 창회에서 떠나는 즐거운 시간들 시작합니다', 'SBS에 내레이션 매니저스펍수 사관학교를 통해 진정한 매니저로 거듭나보자펍!' 이 근질근질했는데 이제 대놓고 자랑해도 돼?', '고3들이 기다려라! 펍수가 간다!', '이거 보고 내일 화보 보면 너 꿀잼', '펍수를 5 분우 로 초대할 셸럽의 정체가 드러났다..!S 본부의 특급 셸럽 배성재가 직접 펍수의 자질을 확인하고 모종의 거래까지..?', '우리 친구들을 위 해일요일 밤에 올림니당 펍펍', 'SBS의 셸럽을 자초한 누군가가 초대장을 보내왔다! 펍수의 SBS 정복기 1편 (feat. 재재)', '펍수는 구독자 여러분들이 있어서하나도 외롭지 않습니다!', '펍수가 드디어 해외진출을?!???펍수 친구 엘로디의 프랑스 진출 꿀팁 대방출!!펍수는 프랑 스 진출에 성공할 수 있을까??', '이 영광을구독자 여러분들께 돌립니다!', '펍수가 펍권이 아니다?!?!? 펍수를 둘러싼 정체에 대한 의혹 날날이 파헤쳐 보겠습니다 펍펍!!', '펍수 당근 10살 맞거든요?(당근을 흔든다)', '목요일 20시30분 전주재방금요일 20시 30분 본방 연속 2편일요일 12시30분 전주재방, 본방재방 총 4편"생방송 톡!톡! 보니하니" 인서트 18시 10분경 전주재방 2편자주 만나도 자주 사랑해주세요

영화 '전문: 하늘에 묻는다' 오디션 보러 간 펍수(*쿠키영상 있음!)

조회수 1,136,170회 · 2019. 12. 10. 👍 3.8만 💬 416 🔗 공유 📌 저장 ...

 **자이언트 펍TV**
구독자 133만명 구독

첫 영화 오디션을 본 펍수!
라면 먹고 돌아오다?

불필요한 음원 정보 삭제

[MUSIC INFO]

1. What's Your Name (If You Want The Part, Earn It)_J.K. Simmons_Whiplash OST
2. The Avengers (From _Avengers Assemble_)_London Music Works
3. ㅋ_장기하와 얼굴들
4. All You Need Is Love (With Orchestra)_The Beatles Revival Band
5. 로망_장미여관_미생 OST
6. under pressure / Rhythm Nation (Feat. Happy Feet Two Chorus) _Pink
7. 명랑_김태성_명랑 OST
8. 광해 왕이 된 남자 Opening_모그
9. 급할수록 서둘러라_Unknown_미생 OST
10. Slow Slow Quick Quick_김종천_경숙이, 경숙아버지 OST

그 외 음원, 모두كم 제공

카테고리 [영화/애니메이션](#)

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (최종 DataFrame) - 공통

```
# title, 주소, 조회수, 본문내용 추가한 데이터 프레임
```

```
peng_list_update = pd.DataFrame({'제목':n_title_list, '주소':n_href_list, '조회수':clean_n_viewcount,  
                                '좋아요':clean_n_like_list, '싫어요':clean_n_unlike_list, '댓글수':str_n_comments, '본문내용':clean_s
```

```
peng_list_update.head()
```

	제목	주소	조회수	좋아 요	싫어 요	댓글 수	본문내용
0	[Ep.67] 전 세계 게 샀거라 (feat. 외교부)	/watch?v=yUJAvW2Rykc	1,380,758	42000	749	6,838	K-펑권 한다 해외진출
1	[Ep.66] 화보 모델 펑수	/watch?v=XUM3sH1kBtw	1,580,523	46000	455	7,678	펑수가 화보 모델로서 촬영을 하게 되었다!화보 촬영을 위한 펑수의 눈물(?) 나는 ...
2	펑수와 팬들의最強 콜라베이션 신상유비 (feat. 찰린지♡)	/watch?v=LPmyxMH96S8	603,876	29000	135	3,921	힙합펑수의 커버영상이 드. 디. 어. 찾아왔다!월미도에서 촬영했던 미공개 영상과 여...
3	[Ep.65] 세상에 나쁜 펑권은 없다.	/watch?v=wedLGh2jxkQ	1,971,318	50000	548	6,974	어느 날 갑자기 펑수에게 이상한 행동들이 보인다? 걱정된 제작진들이 긴급 솔루션 ...
4	수험생은 지금 당장 이 영상을 봅니다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	650,703	20000	135	2,760	내일 모레 엄청 큰 시험을 앞둔여러분들을 위해 이 영상을 바칩니다!당 펑펑

댓글 수까지 최종 업데이트된 펑수 영상의 DataFrame

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (예시 확인) - 공통

0번째 영상에서 추출한 댓글 DataFrame
youtube_pd.head()

* 0번째 영상에서 추출한 댓글 DataFrame

	ID	Comment	Like	Title_number
0	Sol Sol	평수전용 빨대 제작해서 물 좀 주세요염 !!!..	1100	0
1	윤하슬	방송국놈들 평수 인기 많다고 스케줄 많이 잡지마 ;	1100	0
2	문성수	각 방송사 아나운서 조차 이름 잘 모르는 지금 이시대....EBS 김명중이 누군지 ...	3000	0
3	내삶 유포리아	제작진이 평수를 너무좋아하는거같음ㅋㅋㅋㅋㅋㅋ	3000	0
4	Music of The Night Korea	평수야 안녕! 뮤지컬 <오페라의 유령> 월드투어 팀인데 우리 공연 보러 오지 않을래...	1200	0

전체 영상에서 추출한 댓글 DataFrame
total_youtube_pd

* 전체 영상에서 추출한 댓글 DataFrame

	ID	Comment	Like	Title_number
0	대한민국외교부	평수 와줘서 고마워요! 해외진출의 꿈을 응원할게요~ 2019 한·아세안 특별정상회의...	3600	0
1	평랑단1호	악플러들 고소하고 EBS 빛 갓자	3600	0
2	Isabel la	아니 무슨 교육방송이 ㅋㅋㅋ 웬만한 예능프로그램보다 더 잘만들어 자막이고 편집이고 ...	1300	0
3	월드곰탕이	조반 영상부터 다 봐온 사람으로서열심히 한 죄밖에 없는데너무 힘들게 하는듯나쁜 댓글...	1300	0
4	gravity	위험한 물품 가지고 있어여? "제 자신" ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ	1000	0
5	Henry Kang	평수 마트료시가 굿즈 나오면 구매하실 분👉	1000	0
6	Zaya Kim	평수야 나는 덴마크 사는 한국사람이야 그런데 여기 친구들한테 너의 동영상을 전도하고...	908	0
7	Bak Shin	외교부편이 진짜 레전드네요. 어른이들과의 합도 좋고. 꿀벌평수, 판다평수, 취권평수...	908	0
8	열공열공	외교부 10살한테 다산의 상징이라뇨 ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ	635	0

1-2. 데이터 전처리 과정:

2) 데이터 입력 및 처리 (csv 파일 저장) - 공통

```
# 펭수 intro csv 저장
intro_df.to_csv("peng_intro.csv", mode='w', encoding="utf-8-sig")

# 펭수 동영상 csv 저장
peng_list_update.to_csv("peng.csv", mode='w', encoding="utf-8-sig")

# 펭수 댓글 csv 저장
total_youtube_pd.to_csv("peng_reply.csv", mode='w', encoding="utf-8-sig")
```

최종 추출된 3개의 DataFrame을 워크맨 자료와 비교하기 위해 csv파일로 변환.

이후 워크맨에서도 동일한 작업 진행하여 csv 파일 변환 후,
데이터 시각화 및 머신러닝 진행

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 (새 파일 생성하여 df 통합) – 패키지 import

```
#펍수/워크맨 csv 불러와서 시각화 및 머신러닝
# 패키지 import
import datetime as dt
import pandas as pd #df 분석용
import requests
import time
import urllib.request #
import re
import konlpy
#import konlpy.tag import Okt

from bs4 import BeautifulSoup
from pandas import DataFrame
from selenium.webdriver import Chrome
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
from nltk import FreqDist
```

새로운 파일 생성하여
펍수의 csv + 워크맨의 csv를 통합 후,
시각화 및 머신러닝 진행 예정

```
# 시각화
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from PIL import Image #워드클라우드용
from wordcloud import WordCloud #워드클라우드용
fpath = "NotoSansCJKkr-Bold.otf" #워드클라우드 국문지원을 위한 별도 폰트 path 설정
%matplotlib inline
```

```
# 기계 학습
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
```

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 (새 파일 생성하여 df 통합) – 파일 Read 및 통합

```
##### Read #####
peng_intro = pd.read_csv('peng_intro.csv') # 펭수의 intro DF
work_intro = pd.read_csv('work_intro.csv') # 워크맨의 intro DF

peng = pd.read_csv('peng.csv')           #펭수의 동영상list DF
peng_reply = pd.read_csv('peng_reply.csv') #펭수의 동영상별 댓글list DF

work = pd.read_csv('work.csv')           #워크맨의 동영상list DF
work_reply = pd.read_csv('work_reply.csv') #워크맨의 동영상별 댓글list DF
```

```
##### DF 통합 #####
# 1) 펭수 + 워크맨 intro DF 통합
intro_paw = pd.DataFrame(peng_intro)
intro_paw = intro_paw.append(work_intro)

# 2) 펭수 + 워크맨 동영상list DF 통합
paw = pd.DataFrame(peng)
paw = paw.append(work)

# 3) 펭수 + 워크맨 동영상별 댓글list DF 통합
paw_reply = pd.DataFrame(peng_reply)
paw_reply = paw_reply.append(work_reply)
```

intro_paw.head()

	Unnamed: 0	구분	구독자	영상수
0	1	펭수	135만명	133
0	2	워크맨	353만명	35

각 dataset에서 추출한 csv를
Read하여 새 파일로 통합

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 (새 파일 생성하여 df 통합) – 새 속성 추가

```
##### 구분자 추가 #####
```

```
for i in range(len(peng)): peng['구분'] = '펭수'
for i in range(len(work)): work['구분'] = '워크맨'
for i in range(len(peng_reply)): peng_reply['구분'] = '펭수'
for i in range(len(work_reply)): work_reply['구분'] = '워크맨'
```

통합 후 항목 구분을 위해
'구분' 열 추가

```
paw.head() # 통합한 동영상 list DF 확인
```

Unnamed: 0		제목	주소	조회수	좋아요	싫어요	댓글수	본문내용	구분
0	0	[Ep 67] 전 세계 게 샀거라 (feat. 외교부)	/watch?v=yUJAvW2Rykc	1,374,702	41000	748	20	K-펭귄 한다 해외진출	펭수
1	1	[Ep.66] 화보 모델 펭수	/watch?v=XUM3sH1kBtw	1,562,005	45000	453	20	펭수가 화보 모델로서 촬영을 하게 되었다! 화보 촬영을 위한 펭수의 눈물(?) 나는 ...	펭수
2	2	펭수와 팬들의 최강 콜라베이션 신상뮤비 (feat. 챌린지♡)	/watch?v=LPmyxMH96S8	600,930	28000	133	20	힙합펭수의 커버영상이 드. 디. 어. 찾아왔다! 월미도에서 촬영했던 미공개 영상과 여...	펭수
3	3	[Ep.65] 세상에 나쁜 펭귄은 없다.	/watch?v=wedLGh2jxkQ	1,944,586	50000	539	20	어느 날 갑자기 펭수에게 이상한 행동들이 보인다!? 걱정된 제작진들이 긴급 솔루션 ...	펭수
4	4	수험생은 지금 당장 이 영상을 보니다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	648,478	20000	135	20	내일 모레 엄청 큰 시험을 앞둔 여러분들을 위해 이 영상을 바칩니다! 펭펭	펭수

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 (새 파일 생성하여 df 통합) – DF별 describe

```
intro_paw.describe()
```

	Unnamed: 0	영상수
count	2.000000	2.000000
mean	1.500000	84.000000
std	0.707107	69.296465
min	1.000000	35.000000
25%	1.250000	59.500000
50%	1.500000	84.000000
75%	1.750000	108.500000
max	2.000000	133.000000

```
paw_reply.describe() # 통합한 동영상별 댓글 list DF 확인
```

	Unnamed: 0	Like	Title_number
count	3160.000000	3160.000000	3160.0
mean	9.500000	1430.986392	0.0
std	5.767194	2652.356452	0.0
min	0.000000	0.000000	0.0
25%	4.750000	193.000000	0.0
50%	9.500000	546.500000	0.0
75%	14.250000	1500.000000	0.0
max	19.000000	38000.000000	0.0

```
paw.describe()
```

	Unnamed: 0	싫어요
count	158.000000	158.000000
mean	51.253165	467.43038
std	36.693070	685.10947
min	0.000000	8.000000
25%	19.250000	70.000000
50%	43.500000	163.500000
75%	67.750000	357.000000
max	99.000000	1000.000000

DataSet Count

- Intro_paw: 2개 # 비교 채널 수
- Paw: 168개 # 영상 수
- Paw_reply: 3160개 # 영상에서 추출한 댓글 수

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 (새 파일 생성하여 df 통합) – Read 및 통합

```
peng_and_work.describe() # 통합한 동영상 list DF 확인
```

	Unnamed: 0	좋아요	싫어요	댓글수
count	158.000000	158.000000	158.000000	158.0
mean	51.253165	29871.518987	463.455696	20.0
std	36.693070	32547.930539	683.857777	0.0
min	0.000000	2300.000000	7.000000	20.0
25%	19.250000	9775.000000	67.250000	20.0
50%	43.500000	17500.000000	162.000000	20.0
75%	82.750000	35750.000000	459.750000	20.0
max	122.000000	180000.000000	3600.000000	20.0

```
peng_and_work_reply.describe() # 통합한 동영상별 댓글 list DF 확인
```

	Unnamed: 0	Like	Title_number
count	2460.000000	2460.000000	2460.0
mean	9.500000	1504.704878	0.0
std	5.767454	1275.499039	0.0
min	0.000000	10.000000	0.0
25%	4.750000	780.000000	0.0
50%	9.500000	1100.000000	0.0
75%	14.250000	1500.000000	0.0
max	19.000000	15000.000000	0.0

통합한 df list 정보 확인

- 동영상 list
(peng_and_work)
: 158개의 영상
- 동영상 별 댓글 list
(peng_and_work_reply)
: 2460개의 댓글

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 (새 파일 생성하여 df 통합) – Read 및 통합

```
peng_and_work.head()
```

Unnamed: 0		제목	주소	조회수	좋아요	싫어요	댓글수	본문내용
0	0	[Ep.67] 전 세계 게 섰거라 (feat. 외교부)	/watch?v=yUJAvW2Rykc	1,374,702	41000	748	20	K-펑권 한다 해외진출
1	1	[Ep.66] 화보 모델 펑수	/watch?v=XUM3sH1kBtw	1,562,005	45000	453	20	펑수가 화보 모델로서 촬영을 하게 되었다! 화보 촬영을 위한 펑수의 눈물(?) 나는 ...
2	2	펑수와 팬들의 최강 컬래버레이션 신상유비 (feat. 챌린지♡)	/watch?v=LPmyxMH96S8	600,930	28000	133	20	힙합펑수의 커버영상이 드. 디. 어. 찾아왔다! 월미도에서 촬영했던 미공개 영상과 여...
3	3	[Ep.65] 세상에 나쁜 펑권은 없다.	/watch?v=wedLGh2jxkQ	1,944,586	50000	539	20	어느 날 갑자기 펑수에게 이상한 행동들이 보인다!? 걱정된 제작진들이 긴급 솔루션 ...
4	4	수험생은 지금 당장 이 영상을 봅니다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	648,478	20000	135	20	내일 모레 엄청 큰 시험을 앞둔 여러분들을 위해 이 영상을 바칩니다! 펑펑

```
peng_and_work.tail()
```

Unnamed: 0		제목	주소	조회수	좋아요	싫어요	댓글수	본문내용
30	30	독립 채널 ※충격※ 반도의 안 흔한 시급 17,000원 짜리 알바..?! 공짜치킨...	/watch?v=A0as_0ObgU	5,171,242	38000	876	20	시급 17,000원(?)에 일하면서 공짜 치킨까지! 세상에 이런 알바가 존재한다고요...
31	31	독립 채널 ※댓글 정정 주의※ 몸과 마음이 퓨어하다 못해 푸어...해짐★ 역대급...	/watch?v=QKTEbrFDdBc	4,446,505	44000	846	20	근무중 멘탈 나감 주의! 갱 갱 갱 청구 가능한냐는 장성규 저 세상 드립 무엇ㅋㅋㅋㅋ...
32	32	독립 채널 사장님도 비추하는 극한 ★고깃집 알바 ★솔직 리뷰(ft.고기 잘 구우...	/watch?v=MnJqDkNwlsk	4,306,791	48000	848	20	손님들의 🍳솔솔한 팁까지! 기대해볼 수 있다는 고깃집 알바 리뷰.soljik03:4...
33	33	독립 채널 워크맨 사상 이외의 1등 시급 등장!!! 방학 단기 알바 끝판왕 ...	/watch?v=Das7SCjARak	3,598,478	43000	591	20	여름방학 단기 알바 찾는 사람들 주목! 워크맨 시급 순위 1등에 빛나는 🍷단기 알바 ...

각 DataFrame 확인

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 (새 파일 생성하여 df 통합) – Read 및 통합

```
peng_and_work_reply.head()
```

	Unnamed: 0	ID	Comment	Like	Title_number
0	0	대한민국외교부	평수 와줘서 고마워요! 해외진출의 꿈을 응원할게요~ 2019 한·아세안 특별정상회의...	3600	0
1	1	평랑단1호	악플러들 고소하고 EBS 빛 갓자	3600	0
2	2	Isabel la	아니 무슨 교육방송이 ㅋㅋㅋ 웬만한 예능프로그램보다 더 잘만들어 자막이고 편집이고 ...	1300	0
3	3	월드곰탱이	초반 영상부터 다 봐온 사람으로서열심히 한 죄밖에 없는데너무 힘들게 하는듯나쁜 댓글...	1300	0
4	4	gravity	위험한 물품 가지고 있어여? "제 자신" ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ	1000	0

```
peng_and_work_reply.tail()
```

	Unnamed: 0	ID	Comment	Like	Title_number
695	15	김준	해병대 나오셨어요? 저는 그냥 4년제.. ㅋㅋㅋㅋ 🤔	3100	0
696	16	이정식	01:16 회미의 취미할때 표정보소 ㅋㅋㅋ	1000	0
697	17	풍에	여직원분 매니저님인데...관리자중에 2번째로 높으신분임...	1000	0
698	18	Ebony Kim	왜.... 선배 웃기고 좋은데..... 꼭 다 웃으면서 항상 친절해야되는건 아니...	5000	0
699	19	양동호	18년도 여름성수기에 원마운트 실내워터파크 라가로 알바했었는데 매니저님 되게 착하시...	5000	0

각 DataFrame 확인

1-2. 데이터 전처리 과정:

3) 통합된 데이터 전처리 – 토큰화, 명사분석(불용어 제거, 어간추출)

명사 분석 : 한글 데이터 특성 상 유의미한 가장 작은 단위인 형태소로 변환 후
의미 없는 데이터들(조사, 연결어 등)을 확인

→ 불용어 제거 및 의미 있는 단어 추출이 가능한 명사 분석으로 토큰화

```
# 형태소 분석
```

```
from konlpy.tag import Okt  
okt=Okt()
```

```
# 제목 리스트 명사 분석
```

```
title_list=[]  
for i in range(len(peng_and_work_title_list)):  
    title_list.append(okt.nouns(peng_and_work_title_list[i]))
```

```
# 본문 내용 명사 분석
```

```
content_list=[]  
for i in range(len(peng_and_work_content_list)):  
    content_list.append(okt.nouns(peng_and_work_content_list[i]))
```

```
# 댓글 내용 명사 분석
```

```
reply_list=[]  
for i in range(len(peng_and_work_reply_list)):  
    reply_list.append(okt.nouns(peng_and_work_reply_list[i]))
```

```
# 리스트 안의 리스트 하나의 리스트로 만들기
```

```
def flatten (n):  
    org =[]  
    for i in n :  
        if (isinstance(i,list)):  
            org += flatten(i)  
        else:  
            org.append(i)  
    return org
```

1-2. 데이터 전처리 과정:

3) 통합된 데이터 전처리 – 토큰화, 명사분석(불용어 제거, 어간추출)

```
flatten(title_list)
flatten(content_list)
flatten(reply_list)
```

```
['핑수',
'해외진출',
'꿈',
'응원',
'아세안',
'특별',
'정상',
'회의',
'홍보',
'핑권',
'악플러',
'빛',
'무스',
'교육방송',
'예능',
'프로그램',
'더',
'자막',
'편집',
'프로그램']
```

명사 추출결과 출력

```
title_f = flatten(title_list)
content_f = flatten(content_list)
reply_f = flatten(reply_list)
```

3) 두 데이터 통합 – 워드클라우드 확인

[illegible]

이후 추가적인 전처리를 통해
'워크맨', '펍수' 삭제 후
워드클라우드 재 확인 예정

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 – 빈도분석 확인

```
title_f = flatten(title_list)
content_f = flatten(content_list)
reply_f = flatten(reply_list)
```

```
# 단어 빈도수 확인을 위한 패키지 import
from nltk.book import *
import operator
from nltk.corpus import brown
from nltk.corpus import stopwords
```

```
# 제목 빈도수 확인
fdist1 = FreqDist(title_f)
dict_w = {}
for w in title_f:
    dict_w[w] = fdist1[w]
resultdict = sorted(dict_w.items(), key=operator.itemgetter(1), reverse=True)
print(resultdict)
```

```
[('펍수', 87), ('워크맨', 38), ('펍권', 30), ('리뷰', 29), ('알바', 28), ('펍', 13), ('채널', 12), ('공개', 11), ('독립', 10), ('매니저', 9), ('연습생', 9), ('세상', 8), ('자이언트', 8), ('장성규', 8), ('라이브', 7), ('독자', 7), ('눈물', 7), ('역대', 7), ('최초', 7), ('선배', 5), ('직캠', 5), ('투표', 5), ('게임', 5), ('인', 5), ('것', 5), ('데뷔', 5), ('민속촌', 5), ('언박싱', 4), ('대결', 4), ('편', 4), ('방송', 4), ('마감', 4), ('선공', 4), ('현장', 4), ('도전', 4), ('거대', 4), ('이말년', 4), ('고양예고', 4), ('기념', 4), ('관종', 4), ('남매', 4), ('폭발', 4), ('세계', 3), ('린지', 3), ('영상', 3), ('남극', 3), ('등장', 3), ('우리', 3), ('남', 3), ('첫', 3), ('학교', 3), ('극장', 3), ('스타', 3), ('진짜', 3), ('펍수쇼', 3), ('안', 3), ('힐링', 3), ('선물', 3), ('본격', 3), ('알', 3), ('의', 3), ('수능', 3), ('이야기', 3), ('말', 3), ('피지', 3), ('컬', 3), ('갤러리', 3), ('빽빽이', 3), ('아저씨', 3), ('굿', 3), ('선', 3), ('대회', 3), ('전지영', 3), ('은밀', 3), ('명', 3), ('돌파', 3), ('댓글', 3), ('두', 3), ('주의', 3), ('일', 3), ('홍대', 3), ('뽕로로', 3), ('인생', 3), ('먹방', 3), ('배달', 3), ('시급', 3), ('사장', 3), ('드립', 3), ('끝판', 3), ('꿀', 3), ('극한', 3), ('게', 2), ('팬', 2), ('신상', 2), ('이', 2), ('스', 2), ('인간', 2), ('버튼', 2), ('처음', 2), ('나', 2), ('공', 2), ('비화', 2), ('개', 2), ('정복기', 2), ('추억', 2), ('친구', 2), ('한류', 2), ('축하', 2), ('전격', 2), ('나이', 2), ('살', 2), ('내', 2), ('시간', 2), ('선생님', 2), ('준비', 2), ('웁팍', 2), ('부', 2), ('당신', 2), ('눈', 2), ('잔소리', 2), ('오디션', 2), ('참치', 2), ('송', 2), ('소개', 2), ('테스트', 2), ('복지', 2), ('종', 2), ('세트', 2), ('답양', 2), ('누가', 2), ('낙시', 2), ('공
```

데이터 추이 확인을 위해 빈도분석 진행

Nltk의 FreqDist를 활용하여 빈도수 기준 정렬함(제목, 본문내용, 댓글 각각 정렬)

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 – 빈도분석 확인

본문 내용 빈도수 확인

```
fdist1 = FreqDist(content_f)
dict_w = {}
for w in content_f:
    dict_w[w] = fdist1[w]
content_dict = sorted(dict_w.items(), key=operator.itemgetter(1), reverse=True)
print(content_dict)
```

```
[('핑수', 130), ('리뷰', 74), ('알바', 69), ('장성규', 60), ('워크맨', 47), ('직업', 37), ('스튜디오', 34), ('핑', 28), ('핑권', 25), ('인력', 23), ('것', 20), ('영상', 19), ('독자', 18), ('자이언트', 18), ('매니저', 16), ('소장', 16), ('잡것', 16), ('공개', 14), ('수', 14), ('친구', 13), ('여러분', 12), ('연습생', 12), ('나', 11), ('우리', 11), ('위해', 10), ('선배', 10), ('덜', 10), ('위', 9), ('이번', 9), ('이', 9), ('남극', 9), ('더', 9), ('편', 9), ('세상', 7), ('거', 7), ('꿀팁', 7), ('앞', 7), ('크리에이터', 7), ('뽀로로', 7), ('시간', 6), ('하나', 6), ('방', 6), ('게', 6), ('내', 6), ('준비', 6), ('말', 6), ('지금', 6), ('눈', 6), ('게임', 6), ('사람', 6), ('방송', 6), ('팬', 6), ('체험', 6), ('스승', 6), ('채널', 6), ('에버랜드', 6), ('눈물', 5), ('핑핑', 5), ('워', 5), ('오', 5), ('과연', 5), ('이유', 5), ('시작', 5), ('플', 5), ('인', 5), ('알', 5), ('라이브', 5), ('도전', 5), ('만', 5), ('명', 5), ('안', 5), ('첫', 5), ('배달', 5), ('데뷔', 5), ('실화', 5), ('남매', 5), ('알바생', 5), ('술집', 5), ('촬영', 4), ('직접', 4), ('힐합', 4), ('린지', 4), ('참여', 4), ('날', 4), ('갑자기', 4), ('진', 4), ('가지', 4), ('기', 4), ('셀럽', 4), ('한국', 4), ('제일', 4), ('안전', 4), ('대결', 4), ('시민', 4), ('잔소리', 4), ('수가', 4), ('콜라보', 4), ('오늘', 4), ('성우', 4), ('고민', 4), ('달성', 4), ('한강', 4), ('꿀', 4), ('덧글', 4), ('민속촌', 4), ('버스킹', 4), ('주의', 4), ('제주도', 4), ('일', 4), ('소', 4), ('화보', 3), ('모델', 3), ('디', 3), ('보고', 3), ('확인', 3), ('의', 3), ('진출', 3), ('방출', 3), ('주재', 3), ('사랑', 3), ('건', 3), ('구독', 3), ('난', 3), ('패션', 3), ('특별', 3), ('일일', 3), ('핑수쇼', 3), ('공연', 3), ('발라드', 3), ('락', 3), ('분', 3), ('투표', 3), ('직캠', 3), ('원', 3), ('공포', 3), ('특집', 3), ('핑하', 3), ('미디어', 3), ('때문', 3), ('두', 3), ('여름', 3), ('끝', 3), ('미용실', 3), ('사실', 3), ('홈', 3), ('맨', 3), ('이말년', 3), ('손흥민', 3), ('고양예고', 3), ('예술', 3), ('주목', 3), ('덕분', 3), ('아이돌', 3), ('현장', 3), ('홍보', 3), ('광', 3), ('손배침', 3), ('동년배', 3), ('소통', 3), ('인생', 3), ('유튜브', 3), ('보기', 3), ('진짜', 3), ('보이', 3), ('계속', 3), ('전지영', 3), ('선생님', 3), ('때', 3), ('어린이집', 3), ('항상', 3), ('요즘', 3), ('울', 3), ('술', 3), ('편의점', 3), ('무엇', 3), ('등장', 3), ('도미노피자', 3), ('독립', 3), ('맥주', 3), ('뜻밖', 3), ('영화관', 3), ('워터파크', 3), ('해외진출', 2), ('노력', 2), ('수의', 2), ('커버', 2), ('드', 2), ('월미', 2), ('모범', 2), ('모든', 2), ('리크', 2), ('연', 2), ('꽃', 2), ('내외', 2), ('비행', 2), ('동차현', 2), ('장', 2), ('통해', 2)]
```

본문내용 빈도수 기준 정렬

3) 두 데이터 통합 – 빈도분석 확인

```
title_df.head() # 데이터 확인
```

	0	1
0	목소리	449
1	진	348
2	삶	338

```
title_df.head() # E/O/E 확인
```

데이터 프레임 결과

댓글 내용 빈도수 정렬까지 완료 후 각 데이터 프레임 형태로 저장

1-2. 데이터 전처리 과정:

3) 두 데이터 통합 – 추가 전처리

Peng_and_work의 '제목', '본문내용' FreqDist(?) 구해서
가장 빈도수 높은 단어가 몇프로 차지하는지 출력.

- 제목, 본문내용에 대한 명사화 선행 후,
- Freq df 하나씩 만들어서 워드클라우드 다시

Peng_and_work_reply의 'ID', 'Comment'의 FreqDist(?) 구해서
가장 빈도수 높은 단어(사람, 본문 댓글)가 몇프로 차지하는지 출력.

- 본문내용 명사화 진행 필요
- 제목, 본문내용에 대한 Freq df 하나씩 만들어서 워드클라우드 다시

peng_and_work 의 freq df에서

제목or본문 빈도수가 높은 것에 따라 조회수가 높은지 확인 (정렬 하면 되려나(?))

Peng_and_work_reply의 freq df에서

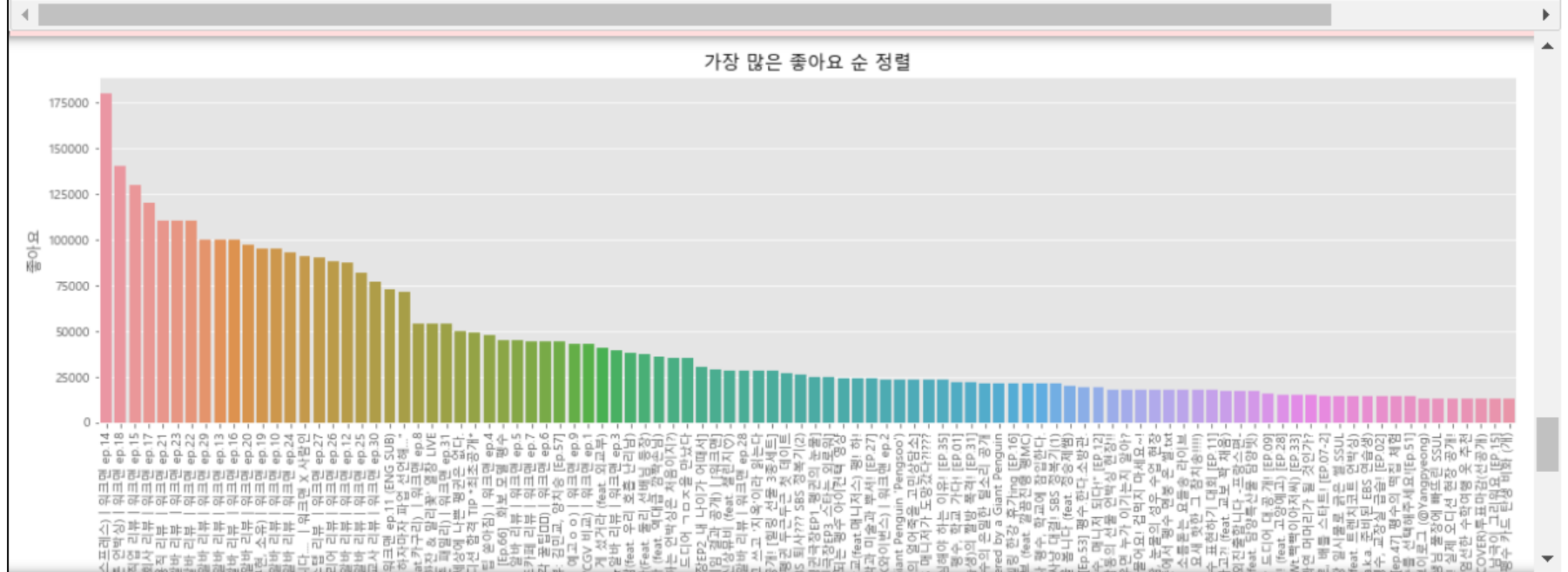
ID에 ○□○ㄹㄴㄹ○ㄴ□ㄴ○아나몰라!!!!

2. 데이터 시각화 및 분석

- 1) 좋아요 순 정렬 및 시각화
- 2) 새로운 key값 생성 (Unlike Ratio) 및 시각화
- 3) df 재분리를 통해 두 값의 차이를 분별
- 4) 워드클라우드 생성 (동영상 별 제목 이용)

2. 데이터 시각화 – 좋아요 순 정렬 및 시각화

```
#시각화
fig = plt.figure(figsize=(20,5))
plt.title('가장 많은 좋아요 순 정렬')
sns.barplot(x='제목', y='좋아요', data=paw[['제목', '좋아요']].groupby('제목', as_index=False).mean().sort_values(by='좋아요', ascending=False))
plt.xticks(rotation=90); #x축(country) 회전
```



```
# 구분에 따른 요약
paw[['좋아요', '구분']].groupby('구분', as_index=False).mean().sort_values(by='좋아요', ascending=False).head()
```

	구분	좋아요	워크맨의 좋아요가 약 5배 이상 많음. 이는 워크맨의 구독자가 3배 이상 많으며, 조회수 또한 동일함. 조회수는 많은 경우 3,000,000이 넘어 python에서 가끔 처리를 못해 '좋아요' 기준을 정렬하였음.
0	워크맨	79628.571429	
1	펑수	15737.398374	

2. 데이터 시각화 – ‘Unlike Ratio’ 추가 생성 및 시각화

: 둘 간 기존 속성값들의 수치 차이로 인해,
평균값 비교를 위한 Unlike Ratio 생성

```
##### UNLIKE RATIO 구하기 #####
# 워킹맨 구독자 수가(353만명) 펑수(134만명)의 3배이상으로,
# 비율을 구해 서로의 값을 확인 예정

unlike_ratio_list = []
for i in range(len(paw)):
    unlike_ratio = (paw.iloc[i, 5] / paw.iloc[i, 4]) * 100 # 퍼센테이지 표현
    unlike_ratio_ar = round(unlike_ratio, 2) # 소숫점 2자리까지 표현
    unlike_ratio_list.append(unlike_ratio_ar)

paw['unlike_ratio(%)'] = unlike_ratio_list # ratio 열 생성
```

```
paw.head()
```

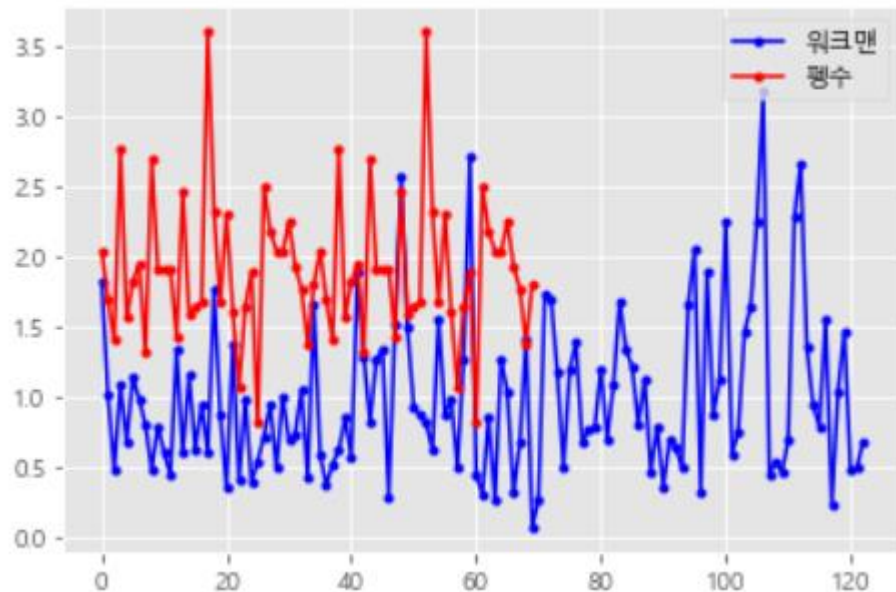
Unnamed: 0		제목	주소	조회수	좋아요	싫어요	댓글 수	본문내용	구분	unlike_ratio(%)
0	0	[Ep.67] 전 세계 게 섰거라 (feat. 외교부)	/watch?v=yUJAyW2Ryk	1,374,702	41000	748	20	K-펑권 한다 해외진출	펑수	1.82
1	1	[Ep.66] 화보 모델 펑수	/watch?v=XUM3sH1kBtw	1,562,005	45000	453	20	펑수가 화보 모델로서 촬영을 하게 되었다! 화보 촬영을 위한 펑수의 눈물(?) 나는 ...	펑수	1.01
2	2	펑수와 팬들의 최강 콜라베이션 신상유비 (feat. 챌린지♡)	/watch?v=LPmyxMH96S8	600,930	28000	133	20	힙합펑수의 커버영상이 드. 디. 어. 찾아왔다! 웰미도에서 촬영했던 미공개 영상과 여...	펑수	0.48
3	3	[Ep.65] 세상에 나쁜 펑권은 없다.	/watch?v=wedLGh2jxkQ	1,944,586	50000	539	20	어느 날 갑자기 펑수에게 이상한 행동들이 보인다? 걱정된 제작진들이 긴급 솔루션 ...	펑수	1.08
4	4	수험생은 지금 당장 이 영상을 봅시다 (feat. 정승제쌤)	/watch?v=25RhZK3HuYM	648,478	20000	135	20	내일 모레 엄청 큰 시험을 앞둔 여러분들을 위해 이 영상을 바칩니다! 펑펑	펑수	0.68

2. 데이터 시각화 – Unlike Ratio 시각화

: df가 통합되어 있어, 색상 구별을 위해 df를 재분리 후 시각화.

```
# unlike에 따른 정렬 및 시각화
peng_list=[] # 현재 df가 통합되어, 색상 분리를 위해 다시 분리
work_list=[]

for i in range(len(paw)):
    if (paw.iloc[i,8]) == '펭수': # '구분'이 펭수일 경우
        peng=paw.iloc[i,9] # 펭수의 ratio를 peng_list에 삽입
        peng_list.append(peng)
    else:
        work=paw.iloc[i,9] # '구분'이 워크맨일 경우
        work_list.append(work) # 워크맨의 ratio를 work_list에 삽입
```



```
plt.plot(peng_list, marker='.', color='b') # 펭수
plt.plot(work_list, marker='.', color='r') # 워크맨
plt.legend(['워크맨', '펭수'], loc='upper right')
```

	구분	unlike_ratio(%)
0	워크맨	1.901143
1	펭수	0.995691

상대적으로
워크맨의 비선호도가 높음 (약 2%)

2. 데이터 시각화 – 워드클라우드

:동영상df를 통해 ‘제목’ 속성으로 워드 클라우드.

‘펑수’, ‘워크맨’이 가장 많이 출력되나,

해당 채널의 성격을 보여주는 ‘알바 리뷰’, ‘feat’, ‘EBS’ 등도 보임.

워드 클라우드 설정

```
title_ls=(''.join(peng_and_work["제목"])) # peng_and_work의 제목df를 string으로
```

#사이즈 등 설정

```
plt.figure(figsize=(10,8))
```

```
plt.imshow(wc)
```

```
plt.tight_layout(pad=0)
```

```
plt.axis('off')
```

```
image_test=np.array(Image.open("ytbb.png")) #mask 이미지 설정
```

```
wc = WordCloud(max_font_size=200, font_path=fpath, stopwords=STOPWORDS, background_color='#FFFFFF', width=1200, height=800, mask=image_test)
```



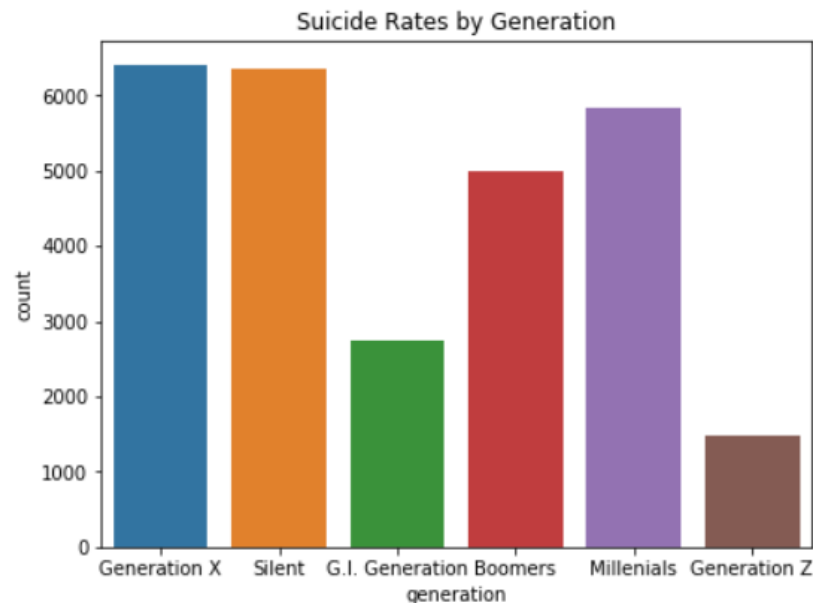
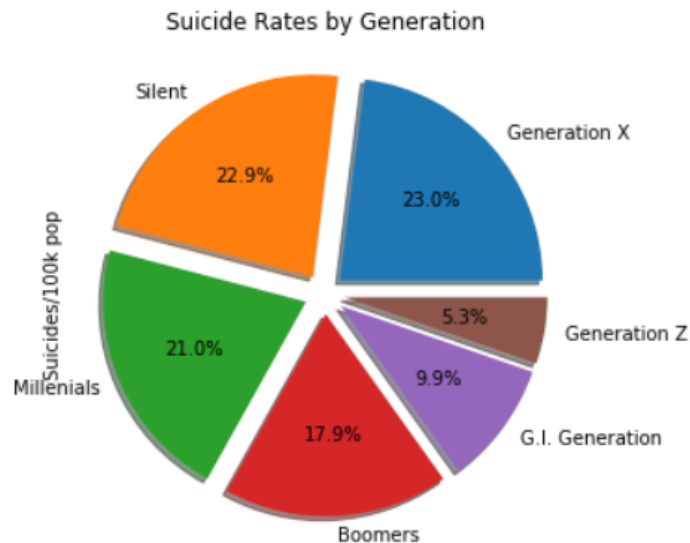
6) Generation에 따른 자살률 확인

Generation X, Silent 시대 자살률이 가장 높다.

#6) generation별 자살률 및 시각화 => Generation X세대 자살률이 가장 높음

```
f,ax=plt.subplots(1,2,figsize=(15,5))
data['generation'].value_counts().plot.pie(explode=[0.1,0.1,0.1,0.1,0.1,0.1],autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('Suicide Rates by Generation')
ax[0].set_ylabel('Suicides/100k pop')
sns.countplot('generation',data=data,ax=ax[1])
ax[1].set_title('Suicide Rates by Generation')
plt.show()

data[['generation','suicides/100k pop']].groupby('generation', as_index=False).mean().sort_values(by='suicides/100k pop', ascending=False)
```



3. Machine Learning

- 사용 알고리즘

- 1) Logistic Regression
- 2) Support Vector Machines
- 3) Decision Tree
- 4) KNN
- 5) Naive Bayes
- 6) Linear SVC
- 7) Perceptron
- 8) Random Forest
- 9) Stochastic Gradient Decent

1) 데이터 추가 전처리 – String형을 정수형으로 변환 (Age, Sex, Generation)

#1) age 의 string을 정수로 바꾸기

```
def func(dataset):  
    if dataset['age'] == '75+':          # 75+ ==> 80 치환  
        return '80'  
    elif dataset['age'] == '55-74':      # 55-74 ==> 65 치환  
        return '65'  
    elif dataset['age'] == '35-54':      # 35-54 ==> 45 치환  
        return '45'  
    elif dataset['age'] == '25-34':      # 25-34 ==> 30 치환  
        return '30'  
    elif dataset['age'] == '15-24':      # 15-24 ==> 20 치환  
        return '20'  
    else:  
        return '10'                    # 5-14 = 10 치환
```

#train data 적용

```
data_train['age'] = data_train.apply(func, axis=1)
```

```
data_train[['age', 'suicides/100k pop']].groupby('age', as_index=False).mean().sort_values(by='suicides/100k pop', ascending=False)
```

#test data도 같이 적용

```
data_test['age'] = data_test.apply(func, axis=1)
```

```
data_test[['age', 'suicides/100k pop']].groupby('age', as_index=False).mean().sort_values(by='suicides/100k pop', ascending=False)
```

Out[92]:

	age	suicides/100k pop
3	45	17.452146
4	65	16.878808
5	80	16.605894
2	30	15.256784
1	20	13.779591
0	10	6.063025

(1) 연령대(Age)부터 string값을 정수형으로 변환

(2) 성별을 정수형으로 변환 (0 = female, 1= male)

2) sex의 string을 정수로 바꾸기

```
def func(dataset):  
    if dataset['sex'] == 'female':          # 여자 = 0 으로 치환  
        return '0'  
    else:  
        return '1'                        # 남자 = 1 로 치환. spelling이 겹치므로 여자부터 치환해야함.  
  
data_train['sex'] = data_train.apply(func, axis=1)  
data_train[['sex', 'suicides/100k pop']].groupby('sex', as_index=False).mean().sort_values(by='suicides/100k pop', ascending=False)  
  
data_test['sex'] = data_test.apply(func, axis=1)  
data_test[['sex', 'suicides/100k pop']].groupby('sex', as_index=False).mean().sort_values(by='suicides/100k pop', ascending=False)
```

	sex	suicides/100k pop
1	1	22.62954
0	0	6.07116

(3) 세대를 정수형으로 변환

3) Generation 의 string을 정수로 바꾸기

```
def func(dataset):  
    if dataset['generation'] == 'G.I. Generation': # G.I generation = 1  
        return '1'  
    elif dataset['generation'] == 'Silent': # Silent = 2  
        return '2'  
    elif dataset['generation'] == 'Boomers': # Boomers = 3  
        return '3'  
    elif dataset['generation'] == 'Generation X': # Generation X = 4  
        return '4'  
    elif dataset['generation'] == 'Millenials': # Millenials = 5  
        return '5'  
    else:  
        return '6' # Generation Z = 6  
  
data_train['generation'] = data_train.apply(func, axis=1)  
data_train[['generation', 'suicides/100k pop']].groupby('generation', as_index=False).mean().sort_values(by='suicides/100k pop', ascending=True)  
  
data_test['generation'] = data_test.apply(func, axis=1)  
data_test[['generation', 'suicides/100k pop']].groupby('generation', as_index=False).mean().sort_values(by='suicides/100k pop', ascending=True)
```

	generation	suicides/100k pop
0	1	27.191793
1	2	22.021429
2	3	16.220836
3	4	11.396403
4	5	5.556242
5	6	0.654696

2) 학습집합 및 테스트 집합 준비

```
▶ X_train = data_train.drop("Over_mean_suicides", axis=1)
X_train = X_train.drop("suicides/100k pop", axis =1)
X_train = X_train.drop("country", axis=1)
Y_train = data_train["Over_mean_suicides"]

X_test = data_test.drop("Over_mean_suicides", axis=1).copy()
X_test = X_test.drop("suicides/100k pop", axis =1)
X_test = X_test.drop("country", axis =1)
X_train.shape, Y_train.shape, X_test.shape

]: ((19474, 10), (19474,), (8346, 10))
```

만들어 준 속성(Over_mean_suicides)을 예측값으로 사용.

이에 영향을 미치는 요소인 suicides/100k pop 은 제거하고,
Country의 경우 나라가 100개이므로 String제거 복잡하여 속성 제거

비율: Train data(19474, 70%), Test Data(8346, 30%)

2) 학습집합 및 테스트 집합 준비

```
▶ X_train = data_train.drop("Over_mean_suicides", axis=1)
X_train = X_train.drop("suicides/100k pop", axis =1)
X_train = X_train.drop("country", axis=1)
Y_train = data_train["Over_mean_suicides"]

X_test = data_test.drop("Over_mean_suicides", axis=1).copy()
X_test = X_test.drop("suicides/100k pop", axis =1)
X_test = X_test.drop("country", axis =1)
X_train.shape, Y_train.shape, X_test.shape

]: ((19474, 10), (19474,), (8346, 10))
```

만들어 준 속성(Over_mean_suicides)을 예측값으로 사용.

이에 영향을 미치는 요소인 suicides/100k pop 은 제거하고,
Country의 경우 나라가 100개이므로 String제거 복잡하여 속성 제거

비율: Train data(19474, 70%), Test Data(8346, 30%)

3) Machine Learning

– Logistic Regression ,SVC, Decision Tree

```
# 1. Logistic Regression
logreg = LogisticRegression(solver='lbfgs')
logreg.fit(X_train, Y_train)

Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

```
]: 70.83
```

**1. Logistic Regression
: 70.83%**

```
# 2. Support Vector Machines
svc = SVC()
solver='liblinear'
svc.fit(X_train, Y_train)

Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
print(acc_svc)
```

```
c:\users\백송이\appdata\local\programs\python\python36\lib\site-packages\skl
gamma will change from 'auto' to 'scale' in version 0.22 to account better f
cale' to avoid this warning.
"avoid this warning.", FutureWarning)
```

```
87.23
```

**2. SVM
: 87.23%**

```
# 3. Decision Tree
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
acc_decision_tree
```

```
]: 96.22
```

**3. Decision Tree
: 96.22%**

3) Machine Learning

– KNN, Gaussian Naïve Bayes, LinearSVC

```
▶ # 4. KNN
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

```
] 92.13
```

4. KNN
: 92.13%

```
▶ # 5. Gaussian Naive Bayes
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

```
] 73.49
```

5. Gaussian Naïve Bayes
: 73.49%

```
▶ # 6. Linear SVC
linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)
Y_pred = linear_svc.predict(X_test)
acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
acc_linear_svc
```

```
c:\users\백송이\appdata\local\programs\python\python36\lib\site-package
to converge, increase the number of iterations.
"the number of iterations.", ConvergenceWarning)
```

```
] 62.06
```

6. Linear SVC
: 62.06%

3) Machine Learning

– Perceptron, Random Forest, SGD

7. Perceptron

```
perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
Y_pred = perceptron.predict(X_test)
acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
acc_perceptron
```

c:\users\백송이\appdata\local\programs\python\python36\lib\site-packages\sklearn: max_iter and tol parameters have been added in Perceptron in 0.19. If both are None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000 (used by 0.19) and tol will be 1e-7 (used by 0.19). (FutureWarning)

68.78

**7. Perceptron
: 68.78%**

8. Random Forest

```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
acc_random_forest
```

96.21

**8. Random Forest
: 96.21%**

9. Stochastic Gradient Descent

```
sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
acc_sgd
```

c:\users\백송이\appdata\local\programs\python\python36\lib\site-packages\sklearn: max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000 (used by 0.19) and tol will be 1e-7 (used by 0.19). (FutureWarning)

62.35

**9. Stochastic
Gradient Descent
: 62.35%**

4) 알고리즘 별 Score 정렬

```
models = pd.DataFrame({  
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',  
             'Random Forest', 'Naive Bayes', 'Perceptron',  
             'Stochastic Gradient Decent', 'Linear SVC',  
             'Decision Tree'],  
    'Score': [acc_svc, acc_knn, acc_log,  
             acc_random_forest, acc_gaussian, acc_perceptron,  
             acc_sgd, acc_linear_svc, acc_decision_tree]})  
models.sort_values(by='Score', ascending=False)
```

	Model	Score
8	Decision Tree	96.22
3	Random Forest	96.21
1	KNN	92.13
0	Support Vector Machines	87.23
4	Naive Bayes	73.49
2	Logistic Regression	70.83
5	Perceptron	68.78
6	Stochastic Gradient Decent	62.35
7	Linear SVC	62.06

Decision Tree의 성공률이
가장 높았으며,
Linear SVC가 가장 낮았음.

5) 최종 파일 생성

국가 및 'if over mean of suicide' 에 대해
최종 파일 작성

```
In [99]: submission = pd.DataFrame({
          "Country": data_test["country"],
          "If_over_mean_of_suicide": Y_pred
        })
submission.to_csv('final_pjt_2017052588.csv', index=False)
```

```
In [100]: data_final = pd.read_csv('final_pjt_2017052588.csv')
data_final.info()
data_final.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8346 entries, 0 to 8345
Data columns (total 2 columns):
Country                8346 non-null object
If_over_mean_of_suicide 8346 non-null int64
dtypes: int64(1), object(1)
memory usage: 130.5+ KB
```

Out[100]:

If_over_mean_of_suicide	
count	8346.000000
mean	0.004553
std	0.067327
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

* If_over_mean_of_suicide
10page에서 만든 속성.
해당 년도에 따른 국가의 자살률이
전세계 평균자살률을 초과하는지 여부