

A rectangular search bar with a light blue border and a white background. Inside the bar, on the left, is a magnifying glass icon. To its right, the text "강남아이티학원_한수정" is entered.

Movie 'CATS' Analysis

- WebCrawling & MachineLearning -

Step 1. Task Setting

Step 2. Data Collecting – WebCrawling

Step 3. Data Preparation

Step 4. Model Training

Step 5. Model Qualification

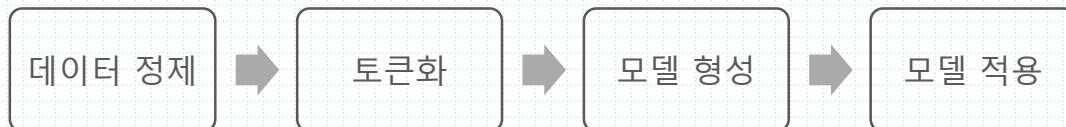
Step 6. Review Prediction

Result

Step 1. Task Setting



- Background : Want to know the rating of the movie 'Cats'
- Purpose : Analyzing Movie Rating using Audience Evaluation
- Data used for Collecting : Naver Movie – Cats
- Collecting Method : Web Scraping by using Jupyter Notebook
- Model used for Analysis : LSTM model
- Analysis method :



캣츠

Cats, 2019

관람객 6.43 기자·평론가 4.75

네티즌 4.59 내 평점 등록>

개요 뮤지컬 드라마 | 미국, 영국 | 109분 | 2019.12.24 개봉

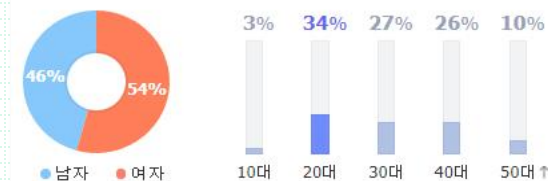
감독 톰 후퍼

출연 제니퍼 허드슨(그리자벨라), 테일러 스위프트(봄발루리나), ... 더보기

등급 [국내] 12세 관람가



성별·나이별 관람추이



다운로드

공식사이트

❤ 3,725



주요정보

배우/제작진

포토

동영상

평점

리뷰

명대사/연관영화

줄거리

오늘밤, 운명을 바꿀 마법 같은 기회가 찾아온다!

1년에 단 하루, 새로운 삶을 살 수 있는 고양이를 선택하는 운명의 밤.

기적 같은 기회를 잡기 위한 축제가 점점 무르익는 동안 뜻하지 않은 위기가 찾아오는데...

① Model used for Web Scraping

```
from bs4 import BeautifulSoup # 리뷰할 내용 긁어오기
from selenium import webdriver # 웹사이트 열기
import time # 시간(현재/멈춤 등)과 관련된 코드 실행
import math # 수학적 계산 실행
import os # 파일 생성/열기/지정
import pandas as pd # 데이터 조작/분석
import numpy as np # 수치계산
```

② Open Web Page

```
# 입력 받기
what_movie = input("영화 제목을 입력하세요. ")
```

Input : 영화 'Cats' 입력하기

```
# 네이버 영화 창 열기
from selenium import webdriver
chrome_path = 'E:\temp\WebCrawling\chromedriver.exe'
driver = webdriver.Chrome(chrome_path)
url = 'https://movie.naver.com/'
driver.get(url)
driver.maximize_window()
import time
time.sleep(2)
```

```
# 쿠키 삭제하기
driver.delete_all_cookies()
time.sleep(1)
```

"쿠키 삭제" : 간혹 사이트에서 자동 크롤링이 감지되면 접근 차단이 되기에 수시로 지우기

③ Search movie 'Cats'

```
search_bar = driver.find_element_by_id('ipt_tx_srch')
search_bar.click()
search_bar.send_keys(what_movie)
search_bar.send_keys("#n")
movie_click = driver.find_element_by_xpath('//*[@id="old_content"]/ul[2]/li[1]/dl/dt/a')
movie_click.click()
movie_grade = driver.find_element_by_link_text('평점')
movie_grade.click()
```

④ Enter to the audience rating page and Check the number of reviewers

```
full_html = driver.page_source
soup = BeautifulSoup(full_html, 'html.parser')
f_review = soup.find('iframe', id="pointAfterListIframe")['src']
driver.get(url+f_review)
review_html = driver.page_source
frame_soup = BeautifulSoup(review_html, 'html.parser')
total_r = frame_soup.find('body').find('div', class_='score_total')
total_r = total_r.find('strong', 'total').find('em').text.strip()
total_r = total_r.replace(",", "")
total_r = int(total_r)
cnt = total_r
```

⑤ Make a function of Web Scraping

```
def get():
    os.chdir(folder_path)

    f.write("-----" + "\n" + "\n")
    print("-" * 70)
    print()
    each_movie = r_list[x]

    number.append(count)
    print("%s번째 리뷰" % count)
    f.write("%s번째 리뷰\n" % count)

    some_review = each_movie.find('div', 'score_reple')
    n_and_d = some_review.find('dl').find_all('em')

    try:
        name = n_and_d[0].find('span').text.strip()
        n_review.append(name)
    except:
        n_review.append("-")
        print("1. Reviewer: -")
        f.write("1. Reviewer: -\n")
    else:
        print("1. Reviewer:", name)
        f.write("1. Reviewer: %s\n" % name)

    try:
        date = n_and_d[1].text.strip()
        d_review.append(date)
    except:
        d_review.append("-")
        print("2. Date: -")
        f.write("2. Date: -\n")
    else:
        print("2. Date:", date)
        f.write("2. Date: %s\n" % date)
```

```
try:
    star = each_movie.find('div', 'star_score').find('em').text.strip()
    s_review.append(star)
except:
    s_review.append("-")
    print("3. Reviewer Rating: -/10")
    f.write("3. Reviewer Rating: -/10\n")
else:
    print("3. Reviewer Rating: %s/10" % star)
    f.write("3. Reviewer Rating: %s/10\n" % star)

try:
    contents = some_review.find('p').find_all('span')
    contents = contents[1].text.strip()
    r_review.append(contents)
except:
    contents = some_review.find('p').find('span').text.strip()
    r_review.append(contents)
    print("4. Review: %s" % contents)
    f.write("4. Review: %s\n" % contents)
else:
    print("4. Review:", contents)
    f.write("4. Review: %s\n" % contents)

try:
    like = each_movie.find('a', '_sympathyButton').find('strong').text.strip()
    l_review.append(like)
except:
    l_review.append("-")
    print("5. Like: -")
    f.write("5. Like: -\n")
else:
    print("5. Like:", like)
    f.write("5. Like: %s\n" % like)

try:
    dislike = each_movie.find('a', '_notSympathyButton').find('strong').text.strip()
    dl_review.append(dislike)
except:
    dl_review.append("-")
    print("6. Dislike: -")
    f.write("6. Dislike: -\n")
else:
    print("6. Dislike:", dislike)
    f.write("6. Dislike: %s\n" % dislike)

print()
print("-" * 70)
f.write("\n")
```

⑥ Web Scraping

```
f = open(text_name, 'a', encoding='UTF-8') # 메모장 열기
count = 1
page = int(np.ceil(cnt/10)) # 긁어야 할 데이터의 페이지수
last_cnt = int(cnt-(page*10-10)) # 마지막 페이지에서 긁어야 할 데이터의 개수

for i in range(1, page+1):
    driver.delete_all_cookies()
    time.sleep(0.5)

    # 만일, 1000페이지 이상일 때 1000단위 ', ' 붙이기
    if i >= 1000:
        i = format(i, ',')
    else:
        pass

    page = driver.find_element_by_link_text('%s %i' % (text_name, i))
    page.click()
    time.sleep(0.5)
    print("%s 페이지 추출을 시작합니다." % i)
    page = int(np.ceil(cnt/10))

    # 데이터 긁어오기
    if i == page: # 마지막 페이지 내용 긁어오기
        review_html = driver.page_source
        soup = BeautifulSoup(review_html, 'html.parser')
        main = soup.find('div', 'score_result').find('ul')
        r_list = main.find_all('li')
        for x in range(0, last_cnt):
            get()
            count += 1
    else:
        review_html = driver.page_source
        soup = BeautifulSoup(review_html, 'html.parser')
        main = soup.find('div', 'score_result').find('ul')
        r_list = main.find_all('li')
        for x in range(0, 10):
            get()
            count += 1
```

⑦ Save data in txt/csv/xlsx

```
os.chdir(folder_path) # 파일을 저장할 경로 지정
```

```
# txt 파일에 크롤링 요약 정보 저장하기
```

```
import sys
```

```
orig_stdout = sys.stdout
```

```
f = open(text_name, 'a', encoding='UTF-8')
```

```
sys.stdout = f
```

```
sys.stdout = orig_stdout
```

```
f.close( )
```

```
# 데이터 프레임 형태로 리스트 취합하기
```

```
movie_frame = pd.DataFrame()
```

```
movie_frame['Review Number']=pd.Series(number)
```

```
movie_frame['Reviewer']=pd.Series(n_review)
```

```
movie_frame['Date']=pd.Series(d_review)
```

```
movie_frame['Reviewer Rating']=pd.Series(s_review)
```

```
movie_frame['Review']=pd.Series(r_review)
```

```
movie_frame['Like']=pd.Series(l_review)
```

```
movie_frame['Dislike']=pd.Series(dl_review)
```

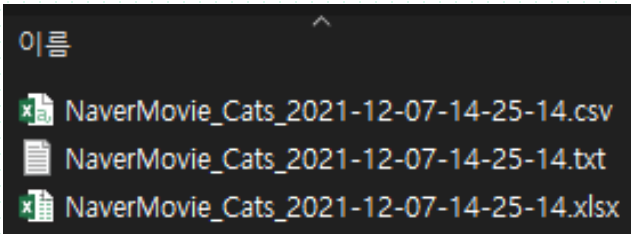
```
# csv 형태로 저장하기
```

```
movie_frame.to_csv(csv_name, encoding="utf-8-sig", index=False)
```

```
# 엑셀 형태로 저장하기
```

```
movie_frame.to_excel(xlsx_name, encoding='utf-8-sig', index=False)
```

Result



txt/csv/xlsx 파일 각각 생성

① Training-Test Data Characteristics

```
id      document      label
9976970 아 더빙.. 진짜 짜증나네요 목소리      0
3819312 흠...포스터보고 초딩영화줄....오버연기조차 가법지 않구나      1
10265843 너무재밌었다그래서보는것을추천한다      0
9045019 교도소 이야기구면...솔직히 재미는 없다..평점 조정      0
6483659 사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 뽀아보이지만 했던 커스틴 던스트가 너무나도 이뻐보였다      1
5403919 막 걸음마 댄 3세부터 초등학교 1학년생인 8살동영화..ㅋㅋㅋ...벌반개도 아까웁.      0
7797314 원작의 긴장감을 제대로 살려내지 못했다.      0
9443947 벌 반개도 아깝다 욕나온다 이웅경 김용우 연기생활이몇년인지..정말 발로해도 그것보단 낫겠다 납치.감금만반복반복..이드라마는 가족도없다
7156791 액션이 없는데도 재미 있는 몇안되는 영화      1
5912145 왜케 평점이 낮은건데? 왜 불만한테.. 할리우드식 화려함에만 너무 길들여져 있나?      1
9008700 강인피니트가장이다.진짜짱이다♥      1
10217543 볼때마다 눈물나서 죽겠다90년대의 향수자극!!허진호는 감성절제메로의 달인이다~      1
5957425 울면서 손들고 횡단보도 건널때 뛰쳐나올뻔 이범수 연기 드럽게못해      0
8628627 당백하고 깔끔해서 좋다. 신문기사로만 보다 보면 자꾸 잊어버린다. 그들도 사람이었다는 것을.      1
9864035 취향은 존중한다지만 진짜 내생애 극장에서 본 영화중 가장 노잼 노감종임 스토리도 어거지고 감동도 어거지      0
6852435 ㄱ냥 매번 긴장되고 재밌음ㅠ      1
9143163 참 사람을 웃기게 바스코가 이기면 락스코라고 까고바비가 이기면 아이돌이라고 깐다.그냥 까고싶어서 안달난것처럼 보인다      1
4891476 굿바이 레닌 표절인것은 이해하는데 왜 뒤로 갈수록 재미없어지나      0
7465483 이건 정말 깨알 캐스팅과 질척하지않은 산뜻한 내용구성이 잘 버무려진 깨알일드!!♥      1
3989148 악탈자를 위한 변명, 이라. 저놈들은 착한놈들 절대 아닌걸요.      1
4581211 나름 심오한 뜻도 있는 듯. 그냥 학생이 선생과 놀아나는 영화는 절대 아님      1
2718894 보면서 웃지 않는 건 불가능하다      1
9705777 재미없다 지루하고, 같은 음식 영화인데도 바베트의 만찬하고 넘 차이남....바베트의 만찬은 이야기도 있고 음식 보는재미도 있는데 ; 이건 볼
471131 절대 평범한 영화가 아닌 수작이라는걸 말씀드립니다.      1
8480268 주제는 좋은데 중반부터 지루하다      0
4254115 다 팔렸을꺼야. 그래서 남독할 수 없었던거야.. 그럴꺼야.. 꼭 그랬던걸꺼야..      0
7295746 k12g 고추를 털어버려야 할텐데      1
5457633 카말라벨 발연기      0
6091784 재밌는영      1
8322926 센스있는 연출력..탁월한 캐스팅..90년대의 향수..그래서 9점..      1
6331922 옴포스의 위력을 다시 한번 깨닫게 해준 적.남 꽃경사님도 연기 정말 좋았어요! 완전 명품드라마!      1
9005897 흠쓰레기 진무하고말도안됨ㅋㅋ 아..시간아까워      0
164908 재밌는데 별점이 왜이리 낮은고      1
9671124 18라도 기대했던 내가 죄인입니다 죄인입니다....      0
5712231 아직도 이 드라마는 내인생의 최고!      1
4726746 패션에 대한 열정! 안나 원투어!      1
7203432 키이라 나이를리가 연기하고자 했던건 대체 정신장애일까 틱장애일까      0
112724 허허...원작가 정신난간 유명이라... 재미있겠네요!      1
1105872 포스터는 있어보이는데 관객은 114명이네      0
4805788 이 영화가 왜 이렇게 저평가 받는지 모르겠다      1
9336496 단순하면서 은은한 매력의 영화      1
9994900 '다 알바생인가 내용도 없고 무서운거도 없고 웃긴거도 하나도 없음 완전 별싱거운 영화.ㅇ.ㅇ내ㅇ시간 넘 아까웁 ... 완전 뉘임      0
6760577 오게두어라! 서리한이 굶주렸다!      1
3440864 정말 맘에 들어요. 그래서 또 보고싶은데 또 보는 방법이 없네? >..ㅠ      1
8548510 음재문이라는 멋진 배우를 발견하게 됐어요. 소소한 일말이 잔잔한 미소를 머금게 합니다. 음악은 조금 아쉽네요ㅠ 8점 주고 싶은데 평점 올
7719892 평점에속지마시길시간낭비 돈낭비      0
```



RangeIndex: 50000 entries, 0 to 49999

Data columns (total 3 columns):

#	Column	Non-Null Count	Dtype
0	id	50000 non-null	int64
1	document	49997 non-null	object
2	label	50000 non-null	int64

dtypes: int64(2), object(1)

memory usage: 1.1+ MB

Step 3. Data Preparation



② Data Preprocessing

```
2 test_data.drop_duplicates(subset = ['document'], inplace=True) # document 열에서 중복인 내용이 있다면 중복 제거
3 test_data['document'] = test_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣]", "") # 정규 표현식 수행
4 test_data['document'] = test_data['document'].str.replace('^ +', "") # 공백은 empty 값으로 변경
5 test_data['document'].replace('', np.nan, inplace=True) # 공백은 Null 값으로 변경
6 test_data = test_data.dropna(how='any') # Null 값 제거
```



데이터 개수	전	후
train_data	150,000	145,393
test_data	50,000	48,852

③ Tokenization

```
1 X_train = []
2 stopwords = ['의', '가', '이', '은', '들', '는', '좀', '잘', '강', '과', '도', '를', '으로', '자', '에', '와', '한', '하다'] # 불용어 지정
3 for sentence in tqdm(train_data['document']):
4     tokenized_sentence = okt.morphs(sentence, stem=True) # 토큰화
5     stopwords_removed_sentence = [word for word in tokenized_sentence if not word in stopwords] # 불용어 제거
6     X_train.append(stopwords_removed_sentence)
```



```
1 tokenizer = Tokenizer()
2 tokenizer.fit_on_texts(X_train)
3 print(tokenizer.word_index)
```

```
{'영화': 1, '보다': 2, '을': 3, '없다': 4, '이다': 5, '있다': 6, '좋다': 7, '너무': 8, '다': 9, '정말':
```

④ Integer Encoding

㉠ 텍스트를 단어 집합으로 만들기 → 각 단어 집합에 고유한 정수 부여하기 # 번호가 높을 수록 단어 등장 빈도수가 높음을 의미

```
1 threshold = 3
2 total_cnt = len(tokenizer.word_index) # 단어의 수
3 rare_cnt = 0 # 등장 빈도수가 threshold보다 작은 단어의 개수를 카운트
4 total_freq = 0 # 훈련 데이터의 전체 단어 빈도수 총 합
5 rare_freq = 0 # 등장 빈도수가 threshold보다 작은 단어의 등장 빈도수의 총 합
6
7 # 단어와 빈도수의 쌍(pair)을 key와 value로 받는다.
8 for key, value in tokenizer.word_counts.items():
9     total_freq = total_freq + value
10
11     # 단어의 등장 빈도수가 threshold보다 작으면
12     if(value < threshold):
13         rare_cnt = rare_cnt + 1
14         rare_freq = rare_freq + value
15
16 print('단어 집합(vocabulary)의 크기 : ',total_cnt)
17 print('등장 빈도가 %s번 이하인 희귀 단어의 수: %s'%(threshold - 1, rare_cnt))
18 print("단어 집합에서 희귀 단어의 비율:", (rare_cnt / total_cnt)*100)
19 print("전체 등장 빈도에서 희귀 단어 등장 빈도 비율:", (rare_freq / total_freq)*100)
```

단어 집합(vocabulary)의 크기 : 43752

등장 빈도가 2번 이하인 희귀 단어의 수: 24337

단어 집합에서 희귀 단어의 비율: 55.62488571950996

전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 1.8715872104872904

④ Integer Encoding

㉞ 등장 빈도수가 낮은 단어 집합 제거

```
1 # 전체 단어 개수 중 빈도수 2 이하인 단어는 제거
2 # 0 번 패딩 토큰을 고려하여 + 1
3 vocab_size = total_cnt - rare_cnt + 1
```

㉟ 텍스트 시퀀스를 숫자 시퀀스로 변환 : 텍스트에 생성한 정수 부여하기 # 0 번은 패딩을 위한 토큰

```
1 tokenizer = Tokenizer(vocab_size)
2 tokenizer.fit_on_texts(X_train)
3 X_train = tokenizer.texts_to_sequences(X_train)
4 X_test = tokenizer.texts_to_sequences(X_test)
```

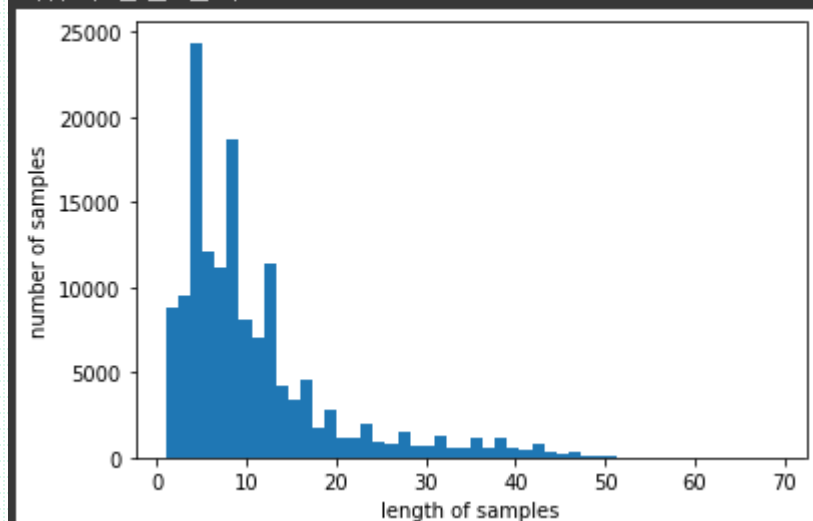
㊤ 빈 샘플 제거 : 삭제된 등장 빈도수가 낮은 단어는 빈 샘플이 됨 → 샘플의 길이를 확인하여 길이가 0 인 샘플 제거

```
1 drop_train = [index for index, sentence in enumerate(X_train) if len(sentence) < 1] # 샘플의 길이가 0 인 샘플들 가져오기
2 # 빈 샘플들을 제거
3 X_train = np.delete(X_train, drop_train, axis=0)
4 y_train = np.delete(y_train, drop_train, axis=0)
```

⑤ Padding

리뷰의 최대 길이 : 69

리뷰의 평균 길이 : 10.812485361182679



< 리뷰 데이터 길이 분포 >

```
1 def below_threshold_len(max_len, nested_list):
2     count = 0
3     for sentence in nested_list:
4         if(len(sentence) <= max_len):
5             count = count + 1
6     print('전체 샘플 중 길이가 %s 이하인 샘플의 비율: %s'%(max_len, (count / len(nested_list))*100))
```

```
1 # 적절한 샘플 길이 찾기
2 def below_threshold_len(max_len, nested_list):
3     count = 0
4     for sentence in nested_list:
5         if(len(sentence) <= max_len):
6             count = count + 1
7     print('전체 샘플 중 길이가 %s 이하인 샘플의 비율: %s'%(max_len, (count / len(nested_list))*100))
8
9 # 샘플 길이가 30인 경우
0 max_len = 30
1 below_threshold_len(max_len, X_train)
```

```
1 X_train = pad_sequences(X_train, maxlen = max_len)
2 X_test = pad_sequences(X_test, maxlen = max_len)
```

LSTM Model

① 모델 만들기

```
1 embedding_dim = 100
2 hidden_units = 128
3
4 model = Sequential()
5 model.add(Embedding(vocab_size, embedding_dim))
6 model.add(LSTM(hidden_units))
7 model.add(Dense(1, activation='sigmoid'))
8
9 es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
10 mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
11
12 model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
13 history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=64, validation_split=0.2)
```

② 모델 저장하기 : 해당 모델을 훗날 다시 사용할 수 있기에 저장해 준다.

```
2 %cd /content/
3 !pwd
4 !ls
5 model.save('best_model.h5')
```

① 정확도 측정

```
1 loaded_model = load_model('best_model.h5')
2 print("\n 테스트 정확도: %.4f" %(loaded_model.evaluate(X_test, y_test)[1]))

527/1527 [=====] - 9s 5ms/step - loss: 0.3471 - acc: 0.8536
테스트 정확도: 0.8536
```



'정확도 : 0.85'로 꽤 높음을 확인할 수 있다.

② 새로운 문장을 예측하는 함수 생성 - 함수 정확도 확인

```
2 def sentiment_predict(new_sentence):
3     new_sentence = re.sub(r'[ㄱ-ㅎㅏ-ㅣ가-힣]*', '', new_sentence)
4     new_sentence = okt.morphs(new_sentence, stem=True) # 토큰화
5     new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
6     encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
7     pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩
8     score = float(loaded_model.predict(pad_new)) # 예측
9     if(score > 0.5):
10         print("{:.2f}% 확률로 '긍정' 리뷰입니다.\n".format(score * 100))
11     else:
12         print("{:.2f}% 확률로 '부정' 리뷰입니다.\n".format((1 - score) * 100))
```

```
1 sentiment_predict('이게 영화인가')
```

```
94.61% 확률로 '부정' 리뷰입니다.
```

```
1 sentiment_predict('이것이 영화인가')
```

```
92.45% 확률로 '부정' 리뷰입니다.
```


Step 6. Review Prediction

① Data Characteristics

```
# 필요한 모듈 불러오기
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # 데이터 시각화 모듈

# 영화 이름 명시하기
print("영화 '%s' 분석을 시작합니다." % what_movie)

# 구글드라이브에서 현재 작업중인 코드로 데이터 불러오기
import pandas as pd
dataset = pd.read_csv(folder_path+'\\\\'+csv_name)

# 데이터셋의 정보를 파악한다
dataset.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5517 entries, 0 to 5516
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Review Number         5517 non-null  int64
1   Reviewer              5517 non-null  object
2   Date                  5517 non-null  object
3   Reviewer Rating       5517 non-null  int64
4   Review                5278 non-null  object
5   Like                  5517 non-null  int64
6   Dislike                5517 non-null  int64
dtypes: int64(4), object(3)
memory usage: 301.8+ KB
```

② Data Preprocessing

```
1 df.drop(['Reviewer', 'Date', 'Like', 'Dislike'], axis=1, inplace=True) # 분석에 사용하지 않는 열 제거
2 df.drop_duplicates(subset=['Review'], inplace=True) # Review 열에서 중복인 내용이 있다면 중복 제거
3 df = df.dropna(how = 'any') # Null 값이 존재하는 행 제거
```

③ Predict Result

① Cats 영화 리뷰를 best_model.h5 모델에 넣어 결과 예측

```
1 review_list = df.iloc[:,2] # Review 열만 추출
2 # 문장 예측해주는 함수
3 cat_list=[]
4 def sentiment_predict(new_sentence):
5     new_sentence = re.sub(r'[ㄱ-ㅎㅏ-ㅣ가-힣 ]',' ', new_sentence)
6     new_sentence = okt.morphs(new_sentence, stem=True) # 토큰화
7     new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
8     encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
9     pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩
10    score = float(model.predict(pad_new)) # 예측
11    if(score > 0.5):
12        cat_list.append("긍정")
13    else:
14        cat_list.append("부정")
15 # 예측 결과 for loop을 이용하여 cat_list 리스트에 담기
16 for i in review_list:
17     sentiment_predict("%s" %i)
18 # 예측 결과를 Result 열로 만들어 기존의 DataFrame에 넣기
19 df['Result'] = cat_list
```

전처리

② 생성한 새로운 DataFrame을 .csv 파일로 저장

```
1 df.drop(['Rating Result', 'Label'],axis=1)
2 df.to_csv('NaverMovie_Cats_ReviewResult_2021-12-08-10-37-20.csv',encoding='euc-kr')
```

④ Result Comparison

① Reviewer Rating을 긍정/부정으로 나누기

```
1 rating_list = df.iloc[:,0] # Reviewer Rating 열만 추출
2 # 결과 for loop을 이용하여 rating_result 리스트에 담기
3 rating_result = []
4 for i in rating_list:
5     if i > 7:
6         rating_result.append("긍정") # 평점 8 이상은 긍정
7     else:
8         rating_result.append("부정") # 평점 7 이하는 부정
9 # 결과를 Rating Result 열로 만들어 기존의 DataFrame에 넣기
10 df['Rating Result'] = rating_result
```

② 결과 예측 값과 실제 결과(평점)을 for loop을 이용하여 긍정/부정이 일치하는지 확인

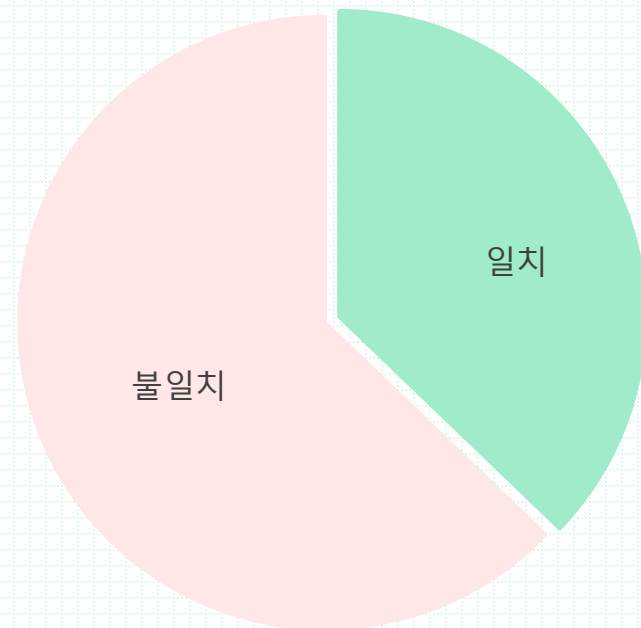
```
1 # 결과 예측 값과 실제 결과(평점)을 for loop을 이용하여 긍정/부정이 일치하는지 확인
2 how_same = []
3 for x in range(0, len(df)):
4     if cat_list[x] == rating_result[x]:
5         how_same.append(0) # 같음을 의미 = 0
6     else:
7         how_same.append(1) # 일치하지 않음을 의미 = 1
8 # 일치 여부를 Label 열로 만들어 기존 DataFrame에 넣기
9 df['Label'] = how_same
```



Result	Rating Result	Label
긍정	부정	1
부정	부정	0
긍정	부정	1
부정	부정	0
부정	부정	0

```
1 # Cats를 분석한 결과 정확도를 측정한다
2 num = (1924/len(df))*100
3 num = round(num,2)
4 print("Cats를 분석한 결과 해당 모델의 정확도는 %s입니다." %num)

Cats를 분석한 결과 해당 모델의 정확도는 37.18입니다.
```



< 일치율 >

“ THANK YOU ”