



## 1 | 주제선정

### ◇ 입소문이 영화 운명 결정…관객당 '3.7회' 정보 검색

올해는 9월과 10월의 누적 관객수가 전년 대비 90% 수준으로 꺾였다. 이는 추석 시즌 관객수가 전년의 76.2%에 지나지 않았기 때문에 나온 결과라고 볼 수 있다. 치열해진 경쟁 상황에서 특정 시즌에 유사한 장르의 영화가 집중적으로 물리면서 이목을 끌지 못했을 뿐 아니라, 관객들이 관람 전 영화정보를 꼼꼼히 검증하는 방식까지 더해진 결과다.

CGV 이승원 마케팅담당은 이런 시장 상황 속에서 '입소문'의 힘에 대해 "지난 10월 조사한 CGV 리서치센터의 '영화선택영향도 조사'에 따르면 일반적으로 관객들이 영화를 선택하기 전에 찾아보는 정보가 평균 3.7개 정도인 것으로 조사됐다"면서 "연령이 어리고, 라이트 유저(Light User) 일수록 자신이 볼 영화에 대해 정보를 탐색하려는 경향이 강해졌다. 관객들은 더 이상 단순히 배우, 감독, 예고편 등과 같은 영화 내적 요인만 가지고 영화를 보지 않는다"고 설명했다.

Google에 의해 종료된 광고입니다.

실제로 관객들이 찾아보는 정보들 중 관람평에 대한 신뢰가 매우 높아 부정적 바이럴에 의한 관람 포기율이 약 33%에 이른다. 그러나 역으로 영화 '서치' '보헤미안 랩소디' '월요일이 사라졌다' 등과 같이 입소문으로 박스오피스 순위를 역주행해 장기 흥행으로 이어지기도 했다.

## 1 | 주제선정



## 1 | 주제선정

개봉 후 2일간의 평점과 댓글이 흥행에 관련이 있을까?



평점과 댓글로 흥행 예측이 가능하지 않을까?



## 2 | 프로젝트 일정

업무	7월					8월					9월						
	3	4	5	8	9	10	2	5	6	7	8	9	23	26	27	28	29
프로젝트 계획			→														
데이터 수집				→													
데이터 전처리					→												
통계 분석						→											
1차 모델링							→										
데이터 추가 수집								→									
추가 수집 데이터 전처리									→								
2차 모델링										→							
웹 서비스 개발											→						
발표												→					



### 3 | 데이터 설명 - 출처



네이버 영화

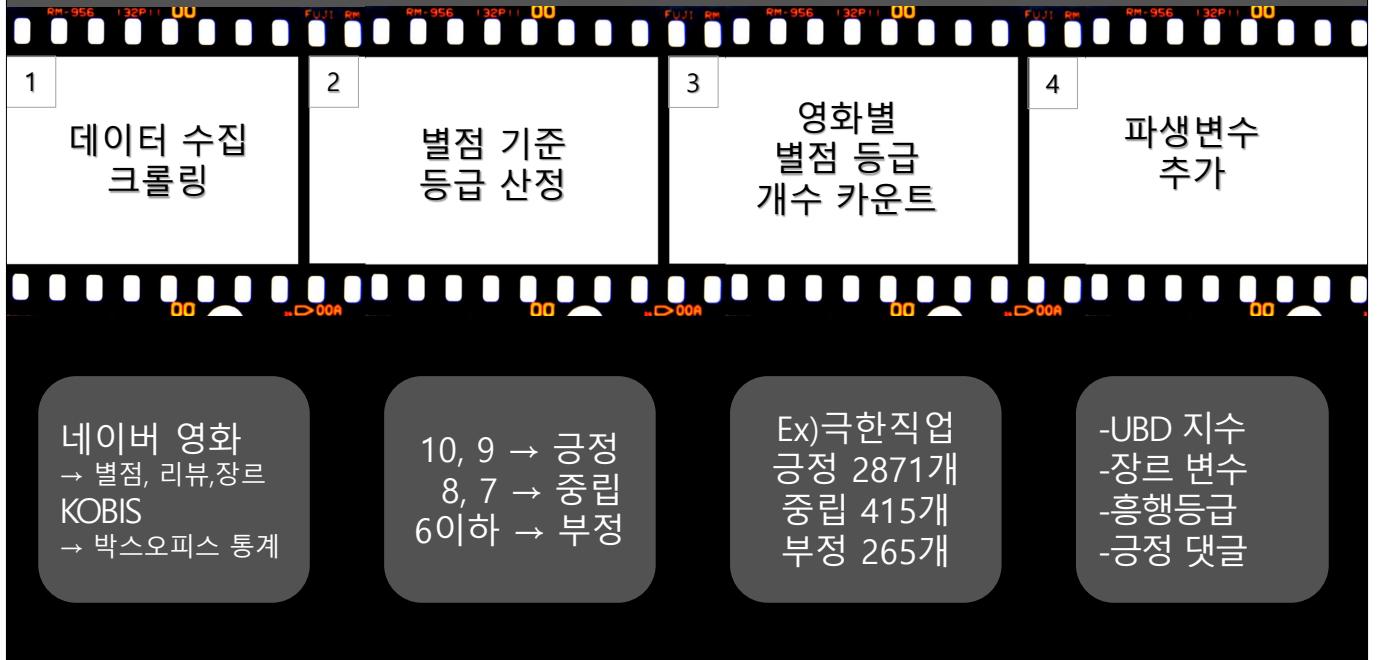


영화관 입장권 통합 전산망

영화별 데이터  
(댓글, 평점, 장르)

영화별 통계자료  
박스오피스

### 3 | 데이터 설명



### 3 | 데이터 설명 – 데이터 수집 크롤링

```
from selenium import webdriver
import time

browser = "C:\ProgramData\Anaconda3\envs\ddat\chromedriver.exe"
driver = webdriver.Chrome(browser)
review = []

driver.get('https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=101966&type')
total = driver.find_element_by_css_selector("body > div > div > div.score_total > strong")
page = int(total.replace(',', '')) / 10 + 1
body > div > div > div.score_total > strong > em
for i in range(2, 5):
    for j in range(1, 11):

        review.append(driver.find_element_by_css_selector("body > div > div > div.score_total > strong > em").text)
        driver.find_element_by_css_selector("#pagerTagAnchor" + str(i)).click()
        time.sleep(1)

with open("review.txt", "w", encoding="utf-8") as f:
    for i in review:
        f.write(i)
        f.write("\n")
f.close()
```

이름	수정한 날짜	유형	크기
포털 블라이.txt	2019-07-16 오후 4:08	텍스트 문서	13KB
할로이.txt	2019-07-16 오후 12:29	텍스트 문서	92KB
해피 데스深化改革 2 유.txt	2019-07-15 오후 6:41	텍스트 문서	1KB
해피리워드 비밀의 향.txt	2019-07-15 오후 7:01	텍스트 문서	215KB
황기.txt	2019-07-16 오후 2:37	텍스트 문서	4KB
한국 미분... 오플리아와 세 개의 결식.txt	2019-07-16 오후 4:15	텍스트 문서	18KB
파이브 러브...txt	2019-07-16 오후 12:41	텍스트 문서	120KB
포토 스코프 4.txt	2019-07-16 오후 12:40	텍스트 문서	10KB
포토 스코프 5.txt	2019-07-16 오후 12:40	텍스트 문서	9KB
포토 스코프 6.txt	2019-07-16 오후 12:40	텍스트 문서	39KB
포토 스코프 7.txt	2019-07-15 오후 7:03	텍스트 문서	1,034KB
포토 스코프 8.txt	2019-07-15 오후 8:19	텍스트 문서	9KB
포토 스코프 9.txt	2019-07-16 오후 2:27	텍스트 문서	11KB
포토 스코프 10.txt	2019-07-15 오후 6:22	텍스트 문서	151KB
포토 스코프 11.txt	2019-07-16 오후 2:26	텍스트 문서	45KB
포토 스코프 12.txt	2019-07-16 오후 2:49	텍스트 문서	50KB
포토 스코프 13.txt	2019-07-15 오후 8:59	텍스트 문서	213KB
포토 스코프 14.txt	2019-07-16 오후 7:17	텍스트 문서	1KB
포토 스코프 15.txt	2019-07-16 오후 5:02	텍스트 문서	43KB
포토 스코프 16.txt	2019-07-15 오후 8:51	텍스트 문서	86KB
포토 스코프 17.txt	2019-07-15 오후 8:57	텍스트 문서	37KB
포토 스코프 18.txt	2019-07-16 오후 12:35	텍스트 문서	19KB
포토 스코프 19.txt	2019-07-16 오후 12:44	텍스트 문서	222KB
포토 스코프 20.txt	2019-07-16 오후 2:18	텍스트 문서	9KB
포토 스코프 21.txt	2019-07-16 오후 5:40	텍스트 문서	4KB
포토 스코프 22.txt	2019-07-16 오후 5:40	텍스트 문서	3KB
포토 스코프 23.txt	2019-07-15 오후 9:03	텍스트 문서	3KB
포토 스코프 24.txt	2019-07-15 오후 8:48	텍스트 문서	373KB
포토 스코프 25.txt	2019-07-16 오후 2:07	텍스트 문서	1,739KB
포토 스코프 26.txt	2019-07-16 오후 2:39	텍스트 문서	25KB
포토 스코프 27.txt	2019-07-16 오후 4:13	텍스트 문서	10KB
포토 스코프 28.txt	2019-07-16 오후 4:55	텍스트 문서	59KB
포토 스코프 29.txt	2019-07-16 오후 10:28	텍스트 문서	367KB
포토 스코프 30.txt	2019-07-17 오후 3:10	텍스트 문서	133KB

### 3 | 데이터 설명 – 데이터 수집 크롤링

```
from selenium import webdriver
import datetime
from selenium.webdriver.common.alert import Alert
import time
browser = "C:\#ProgramData\Anaconda3\envs\ddat\chromedriver.exe"
driver = webdriver.Chrome(browser)
url = 'http://www.kobis.or.kr/business/stat/offc/searchOffcHitTotList.do?searchMode=year'
with open('영화제목3.txt', 'r', encoding='utf8') as r:
    movie_name = r.read()
    movie_title = movie_name.split('\n')
    r.close()
for i in range(0, len(movie_title)):
    # 결선이 있으면
    driver.get(url)
    time.sleep(1)
    search = driver.find_element_by_css_selector("#inp_sqrSearch")
    search.clear()
    search.send_keys(movie_title[i])
    search.submit()
    time.sleep(1)
    try:
        driver.find_element_by_css_selector('#content > div.rst_sch > table > tbody > tr:nth-child(1) > td:nth-child(1) > span > a').click()
        time.sleep(3)
        driver.find_element_by_xpath('/html/body/div[3]/div[1]/div[2]/ul/li[2]/a').click()
        time.sleep(20)

        down = driver.find_element_by_xpath('//*[@id="ui-id-1"]/div/div[2]/div[3]/div/div/div/a')
        time.sleep(1)
        down.send_keys('\n')
        time.sleep(1)
        Alert(driver).accept()
    except:
        print(movie_title[i], "error")
driver.close()
```

### 3 | 데이터 설명 – 데이터 수집 크롤링

KOBIS 박스오피스 통계정보 – 영화 스카이스크래퍼

'스카이스크래퍼' 일자별 통계정보														
- 조회일: 2019-08-09														
3 출처: 영화진흥위원회 통합전산망 ( <a href="http://www.kobis.or.kr">http://www.kobis.or.kr</a> )														
날짜	스크린수	스크린점유율	상영횟수	상영점유율	좌석수	좌석점유율	좌석판매율	매출액	매출증감(전일대비)	관객수	관객수증감(전일대비)	누적매출액	누적관객수	순위
2018-07-11	808	17.3%	4,166	개봉 2일차 스크린 수		개봉 2일차 관객 수		36,000	(6297800 -26176.6% )	78,615	78353 (- 2990.5%)	642,308,000	79,723	2
2018-07-12	831	16.6%	4,129	개봉 9일차 스크린 수		개봉 9일차 관객 수		387,500	(-50248500 -7.9%)	71,383	-7232 (- 9.2%)	#####	151,106	2
2018-07-13	812	15.9%	3,980	21.3%	627,649	21.2%	14.4%	802,736,208	220648708 (37.9%)	90,555	19172 (- 26.9%)	#####	241,661	2
2018-07-14	793	14.4%	3,959	20.0%	620,039	19.8%	31.1%	#####	927462608 (115.5%)	193,028	102473 (- 113.2%)	#####	434,689	2
2018-07-15	791	14.7%	3,872	20.2%	604,958	19.9%	30.0%	#####	-106278210 (-6.1%)	181,248	-11780 (- 6.1%)	#####	615,937	2
2018-07-16	771	15.7%	3,823	21.4%	610,916	21.4%	9.2%	455,476,100	116844450 (-72.0%)	56,277	-124971 (- 69.0%)	#####	672,214	2
2018-07-17	777	15.4%	3,791	개봉 9일차 스크린 수		개봉 9일차 관객 수		5,054,100	(-50422000 -11.1%)	50,327	-5950 (- 10.6%)	#####	722,541	3
2018-07-18	609	11.4%	2,133	11.9%	315,666	11.1%	11.2%	279,073,400	-12598070 (-31.1%)	35,234	-15093 (- 30.0%)	#####	757,775	4
2018-07-19	595	10.5%	1,943	10.8%	293,194	10.3%	10.0%	232,853,700	-46219700 (-16.6%)	29,197	-6037 (- 17.1%)	#####	786,972	4
2018-07-20	607	10.6%	1,941	10.5%	287,492	9.8%	13.3%	332,144,255	99290555 (-42.6%)	38,283	9086 (- 31.1%)	#####	825,255	4
2018-07-21	606	9.8%	1,606	8.3%	218,206	7.2%	29.8%	579,418,639	247274384 (74.4%)	65,048	26765 (- 69.9%)	#####	890,303	4

### 3 | 데이터 설명 – 데이터 수집 크롤링

## 엑셀 매크로를 통한 데이터 합치기

### 3 | 데이터 설명 – 별점 기준 등급산정, 등급 개수 카운트

 10 관광객 우울한 기분을 한방에 날려 줄 재밌고 통쾌한 영화~~~오랜만에 넘재밌는 한국 영화를 봐서 기분이 좋았다~~

카시고(yuno\*\*\*\*) | 2019.07.07 12:40 | 신고

 7 초중반에 통닭 덕분에 나름 재밌었지만 한국 범죄영화 형사영화의 느낌이 살아나면서 아쉬운 느낌의 마무리.

사자개(xyan\*\*\*\*) | 2019.07.07 22:24 | 신고

 1 이런 한심한 수준의 영화가 천만찍을정도니...참담하다...

toyo\*\*\*\* | 2019.02.06 15:08 | 신고

	NAME	긍정	중립	부정
2	극한직업	2871	415	265
3	말모이	1427	150	180
4	내안의 그놈	1279	168	117
5	주먹왕 랄프 2: 인터넷 속으로	499	145	49
6	아쿠아맨	1693	395	253
7	보해미안 랩소디	2478	124	36
8	PMC: 더 병커	1576	553	1237
9	글래스	691	227	266
10	극장판 공룡메카드: 타이니소어의 섬	373	90	12
11	범블비	1150	398	257

### 3 | 데이터 설명 – 파생변수 추가

네이버 영화 ①

극한직업 (Extreme Job, 2018)

관람객 ★★★★★ 9.20 (16,102) | 기자·평론가 ★★★★☆ 6.80 (10)  
평점주기

코미디 2019.01.23. 개봉 111분 | 한국 | 15세 관람가

감독 이병헌  
관객 수 16,264,864명  
관객 불철주야 달리고 구르자만 살적은 바닥, 급기야 해체 위기를 맞는 ... [더보기]

[다운로드] [3,157]

네이버 영화 ①

어벤져스: 엔드게임 (Avengers: Endgame, 2019)

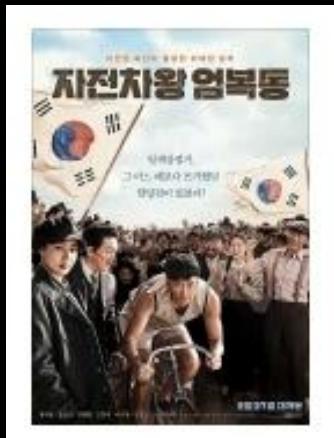
관람객 ★★★★★ 9.50 (32,115) | 기자·평론가 ★★★★☆ 7.62  
평점주기

액션, SF 2019.04.24. 개봉 181분 | 미국 | 12세 관람가  
감독 앤소니 루소, 조 루소  
관객 수 13,934,589명  
내용 인피니티 워 이후 절반만 살아남은 지구 마지막 희망이 된 어벤져스... [더보기]

[다운로드] [33,378]

NAME	코미디	...	액션	SF
극한직업	1	...	0	0
어벤져스: 엔드게임	0	...	1	1

### 3 | 데이터 설명 – 파생변수 추가



#### 영화흥행 평가 새로운 척도 'UBD'와 데이터 단위 변화의 이유

최근 인터넷 유행어 가운데 하나인 유비디(UBD)는 영화티켓 판매량의 단위를 뜻한다. 영화 <자전차왕 엄복동>이 기대와는 달리 관객 17만2천명을 기록하고 막을 내리자 그 관객수가 국내 역대 최고 흥행작인 <명량>의 1%에 해당한다고 하여 엄복동의 영문 앞 글자를 따서 만든 유비디는 17만을 뜻하는 단위처럼 쓰인다. 지금 까지 국내에서 상영된 모든 영화는 1에서 100 유비디 사이에 있으니 단위로 쓰기에도 안성맞춤이다. 영화티켓 판매량뿐만 아니라 현재 우리나라 인구수 5100만명을 17만으로 나누면 공교롭게도 현재 국회의원 정수인 300이라고 하니 쓰임새도 다양하다.

### 3 | 데이터 설명 – 파생변수 추가

NAME	9일차 관객수	UBD
극한직업	4,810,447	28.29675
말모이	1,184,919	6.970112
언니	192,942	1.134953
사탄의 인형	58,018	0.341282

NAME	부진	보통	흥행	대흥행	흥행등급
극한직업	0	0	0	1	4
말모이	0	0	1	0	3
언니	0	1	0	0	2
사탄의 인형	1	0	0	0	1

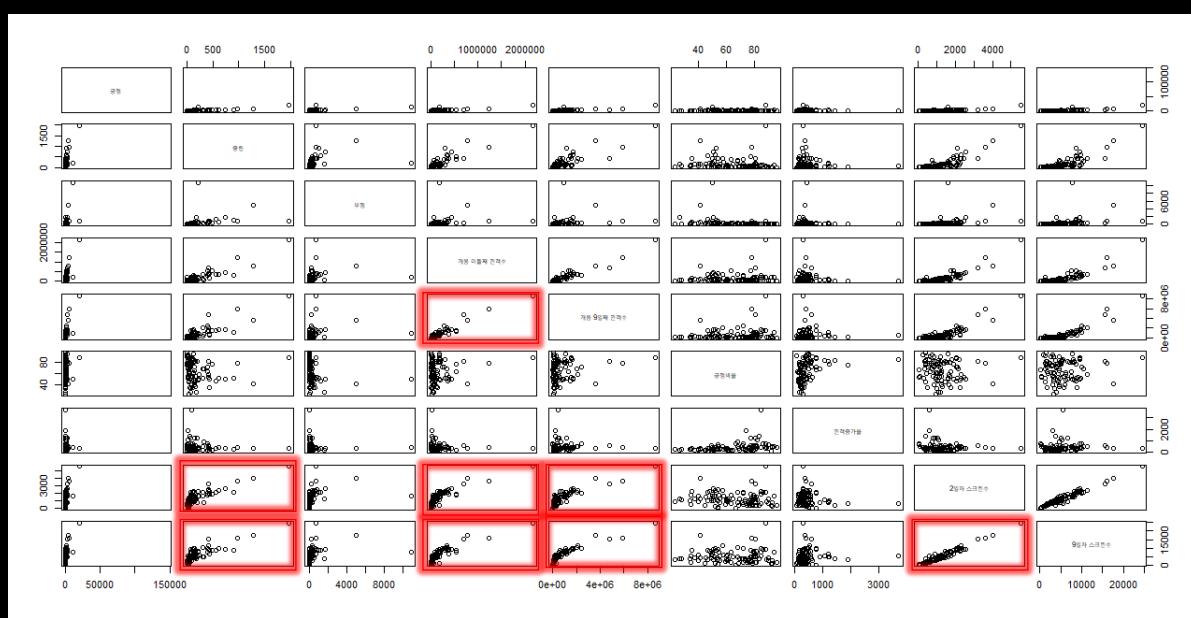
UBD	흥행등급
1 > UBD	1
2 ≥ UBD > 1	2
7 ≥ UBD > 2	3
UBD > 7	4



## 4 | 데이터 분석



## 4 | 데이터 분석 - 상관분석



## 4 | 데이터 분석 – 댓글 긍부정 판정

데이터셋으로 github의 Naver sentiment movie corpus 사용

train 15만, test 5만 총 20만개의 리뷰로 데이터셋 구성

머신러닝의 방법으로 로지스틱 회귀 사용

딥러닝의 방법으로 **LSTM**을 활용한 모델링

## 4 | 데이터 분석 – 댓글 긍부정 판정

### 1. 그리드서치를 이용한 로지스틱회귀 분류모델

```
GridSearchCV(cv='warn', error_score='raise-deprecating',
    estimator=Pipeline(memory=None,
        steps=[('tfidfvectorizer', TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
            dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
            lowercase=True, max_df=1.0, max_features=None, min_df=1,
            ngram_range=(1, 1), norm='l2', preprocessor=None, smooth...penalty='l2', random_state=None, solver='warn',
            tol=0.0001, verbose=0, warm_start=False))]),
    fit_params=None, iid='warn', n_jobs=None,
    param_grid={'tfidfvectorizer__min_df': [3, 5, 7], 'tfidfvectorizer__ngram_range': [(1, 1), (1, 2), (1, 3)], 'logisticregression__C': [0.1, 1, 10, 100]},
    pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
    scoring=None, verbose=0)
```

twit\_grid.best\_score\_  
0.8583866666666666

twit\_grid.best\_estimator\_.na  
0.86506

## 4 | 데이터 분석 – 댓글 긍부정 판정

### 1. 그리드서치를 이용한 로지스틱회귀 분류모델

A	B	C
	document	label
1	시간가는줄모르고藐네요ㅎㅎ	1
2	박누리 감독 데뷔작인데 생각보다 훨씬 탄탄하다. 이야기 자체가 엄청 새롭거나 심오하거나 하지는 않지만 가지고 있는 재료 안에	1
3	手表 3개에 이어 10개면 100억이다!	0
4	9시에 평점1점 글이 올라왔는데 어떻게 조조로보고 나왔다는거지..?	0
5	시사회로 미리봤는데 영화전체를 이끌어가는 뮤준열의 연기가 일품!!! 배우들의 연기와 함께 연출과 음악때문에 심장풀깃합니다.	1
6	속도감 느껴지고 경쾌해서 좋음. 큰 메시지 이런 거 보다 오락영화로서 가볍게 보기 좋다	1
7	영화 퀄팅타임으로 보기애 편	1
8	시사회로 미리 보고 있는데 역시 믿고 보는 뮤준열 그리고 유지진의 열연까지 빼짐없이 잘 만든 무비!	1
9	이 영화 보고 싶은데 재밌나요? 남들은 어떠셨어요?	1
10	스토리 팬찮고 스킬러인줄은 몰랐는데 스킬러 요소도 있네요 그래서 그런지 시간이 엄청 빨리감.. 주식 공부 하고싶은 영화.. 돈 벌	0
11	어제 보고왔는데 보고와서 생각할수록 더 생각이 많아진다.. 생각해보니 영화에서나 현실에서나 돈이 그렇게 무섭다. 가볍게 보고	1
12	9시 땅 하면 평점쓰는 인간들은 그거임 시사회때 보고 써제끼는거 나 역시도 그런데 별로 진짜 한국영화는 돈 주식 금융 나라팔이	0
13	간만에 재밌는 오락영화가 나왔네요	1
14	조우진 보러옴ㅋㅋㅋ	1
15	간만에 재미있게 즐기며 본 영화ㅋㅋ 10억과 100억을 한 눈에 읽을 수 있을것 같다 읽을 일이 있을진 모르겠지만ㅋㅋ 뮤준열은 첨	1
16	배우분들의 연기가 너무 좋았어요 특히 뮤준열배우의 연기가 점점 무르익어 가네요 변화되어가는 표정들이 암권입니다 연출도 입	1
17	너무 재미있요 너무 재미있요	1
18	가운데 참ㅋㅋㅋ하번 더 볼 각!	1

## 4 | 데이터 분석 – 댓글 긍부정 판정

### 2. LSTM를 활용한 모델링

Okt 자연어 처리 패키지를 활용하여 데이터의 전처리를 진행  
전처리 과정에서 숫자와 불필요한 요소를 제거

```
model = Sequential()
model.add(Embedding(max_words, 100))
model.add(LSTM(128))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=5, batch_size=60, validation_split=0.2)
```

```
Epoch 5/5
119996/119996 [=====] - 205s 2ms/step - loss: 0.2599 - acc: 0.8937 - val_loss: 0.3263 - val_acc: 0.8576
```

## 4 | 데이터 분석 – 댓글 긍부정 판정

### 2. LSTM를 활용한 모델링

5000	너무재밌어요 탄도 나왔으면 좋겠어요	0.590714
5001	너무재밌어요 다들 꼭보길	0.614871
5002	너무재밌어요 또볼거예요	0.205715
5003	너무재밌어요 역시 밀고 보는 배우들 라미란 이성경 수영 여성서사 영화의 시작을 알리네요	0.949758
5004	너무재밌어요 좋았어요	0.240248
5005	너무재밌어요 이런영화가더나왔으면좋겠습니다	0.648032
5006	너무재밌어요 전만 가자	0.610871
5007	너무재밌어요 추천합니다	0.611082
5008	너무재밌어요 ㅠㅠㅠㅠ	0.517963
5009	너무재밌어요 자뛰어야겠음	0.481603
5010	너무재밌어요 다음에도 꼭 다시 보고싶네요	0.775959
5011	너무재밌어요 이성경 너무좋아ㅠㅠ	0.252539
5012	너무재밌어요	0.379632
5013	너무재밌어요최고예요 또보러갈거예요	0.080717

테스트 셋의 정확도가  
85% 정도  
나왔음에도 불구하고  
실제 적용했을 때

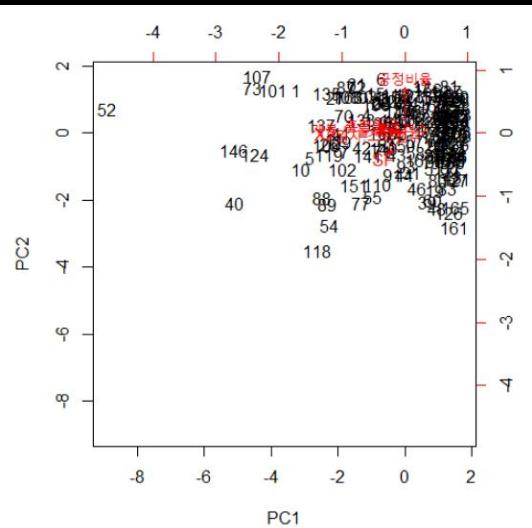
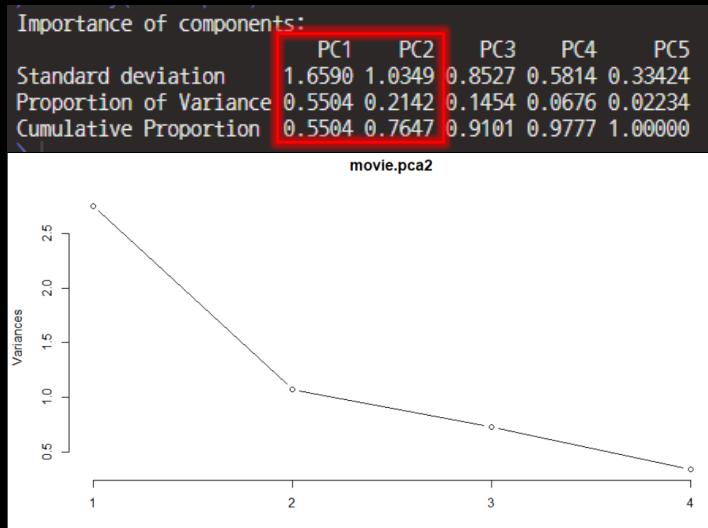
정확성이 떨어짐

## 4 | 데이터 분석 – 댓글 긍부정 판정

NAME	긍정	긍정댓글
국한직업	2871	2714
말모이	1427	1446
내안의 그늘	1279	1306
주먹왕 랄프 2: 인터넷 속으로	499	501
아쿠아맨	1693	1625
보해미안 텁소디	2478	2271
PMC: 더 병커	1576	1769
클래스	691	696
극장판 공룡메카드: 타이니소어의 섬	373	402
범블비	1150	1185
뺑반	1503	1637
러브 유어셀프 인 서울	483	449
그린 북	252	246
스윙키즈	2303	2299
드래곤 킹들이기 3	612	602
언니	332	407
언더독	340	357
점박이 한반도의 공룡2: 새로운 낙원	343	366

이후 다양한 SNS로부터  
크롤링한 리뷰의  
긍부정 판별에  
활용 가능하다고  
판단됨.

## 4 | 데이터 분석 – PCA



## 4 | 데이터 분석 – 다중회귀

활용 가능한 변수를 찾기 위해 단계적 선택법을 적용

Step: AIC=2507.83

개봉, 9일차 관객수, 개봉, 이틀째 관객수 + x2일자 스크린수 + 긍정비율 + SF

	Df	Sum of Sq	RSS	AIC
<none>		8.9775e+12	2507.8	
+ 애니메이션	1	1.1846e+11	8.8590e+12	2508.5
+ 모험	1	1.0916e+11	8.8683e+12	2508.6
+ 드라마	1	1.0532e+11	8.8721e+12	2508.7
+ 코미디	1	6.9845e+10	8.9076e+12	2509.1
+ 미스터리	1	4.9617e+10	8.9279e+12	2509.3
+ 중장편	1	3.7167e+10	8.9403e+12	2509.4
+ 앙상선	1	3.6014e+10	8.9415e+12	2509.4
+ 멜로, 로맨스	1	2.7669e+10	8.9498e+12	2509.5
+ 판타지	1	1.9778e+10	8.9577e+12	2509.6
+ 판타지	1	8.3088e+09	8.9692e+12	2509.7
+SF	1	5.6494e+09	8.9718e+12	2509.8
+부정비율	1	5.5955e+09	8.9719e+12	2509.8
+다큐멘터리	1	4.9853e+09	8.9725e+12	2509.8
+장장대사를	1	4.5525e+09	8.9729e+12	2509.8
+스릴러	1	2.7763e+08	8.9772e+12	2509.8
+극장	1	4.7993e+07	8.9774e+12	2509.8
-SF	1	4.4545e+11	9.4229e+12	2510.6
-부정비율	1	1.0988e+12	1.0076e+13	2517.3
-x2일자 스크린수	1	2.1757e+12	1.1153e+13	2527.3
-개봉, 이틀째 관객수	1	2.1042e+13	3.0019e+13	2625.3

AIC가 가장 낮은 경우를 선택

2일차 관객수  
+ 2일차 스크린 수  
+ 긍정비율  
+ SF

## 4 | 데이터 분석 – 다중회귀

```
lm(formula = 개봉.9일째.관객수 ~ 개봉.이틀째.관객수 + X2일차.스크린수 +
  긍정비율 + SF, data = movie)
```

Coefficients:

(Intercept)	개봉.이틀째.관객수	X2일차.스크린수	긍정비율	SF
-4.582e+05	3.692e+00	2.430e+02	5.090e+03	-2.023e+05

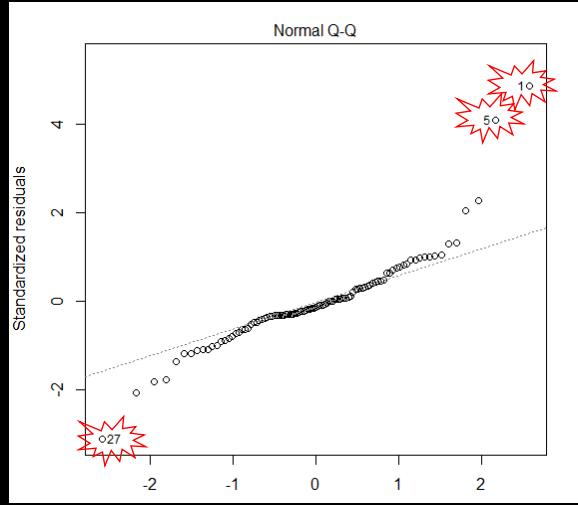
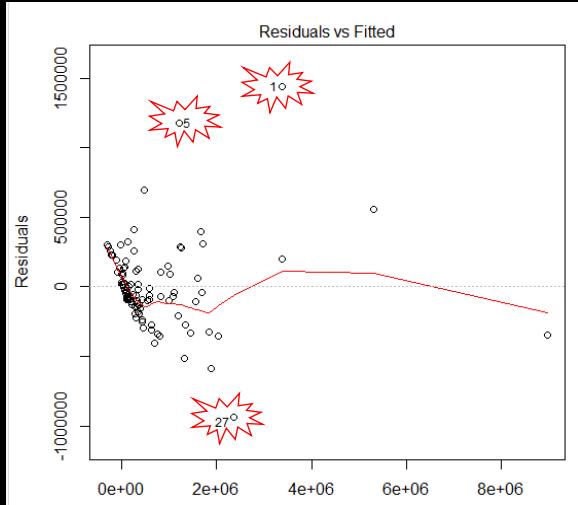
$$Y = -4.582 \times 10^5 + 3.692 \times X_1 + 2.430 \times 10^2 \times X_2 + 5.090 \times 10^3 \times X_3 + -2.023 \times 10^5 \times X_4$$

## 4 | 데이터 분석 – 다중회귀

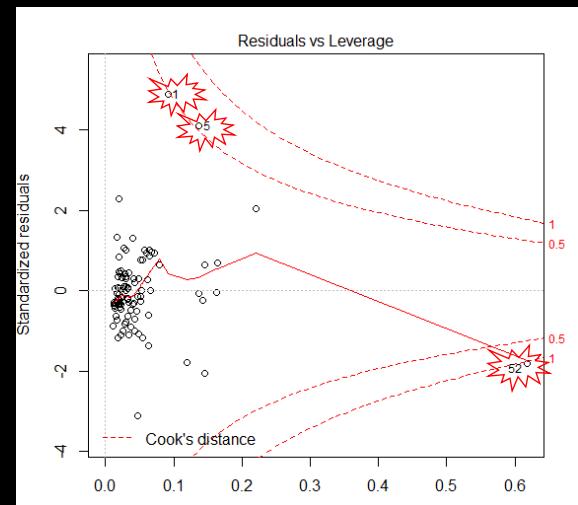
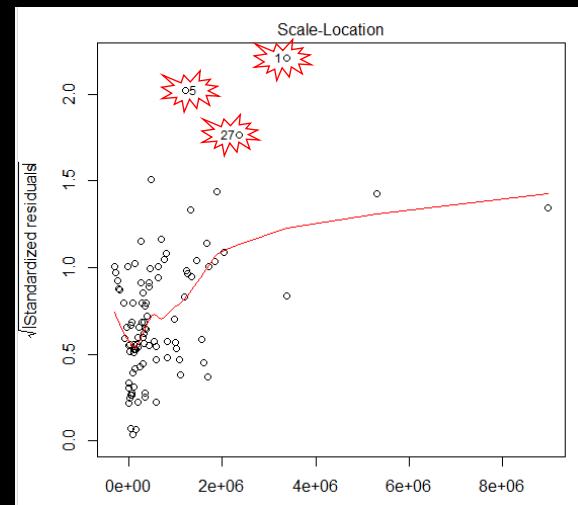
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -6.575e+05  1.572e+05 -4.183 6.47e-05 ***
개봉.이틀째.관객수 3.397e+00  2.289e-01 14.843 < 2e-16 ***
X2일차.스크린수  3.536e+02  7.409e+01   4.773 6.65e-06 ***
긍정비율        6.405e+03  1.888e+03   3.392 0.00102 **  
SF              -2.671e+05  1.237e+05  -2.160 0.03334 *   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 309000 on 94 degrees of freedom
Multiple R-squared:  0.9398    Adjusted R-squared:  0.9372 
F-statistic: 366.9 on 4 and 94 DF,  p-value: < 2.2e-16
```

## 4 | 데이터 분석 – 다중회귀



## 4 | 데이터 분석 – 다중회귀

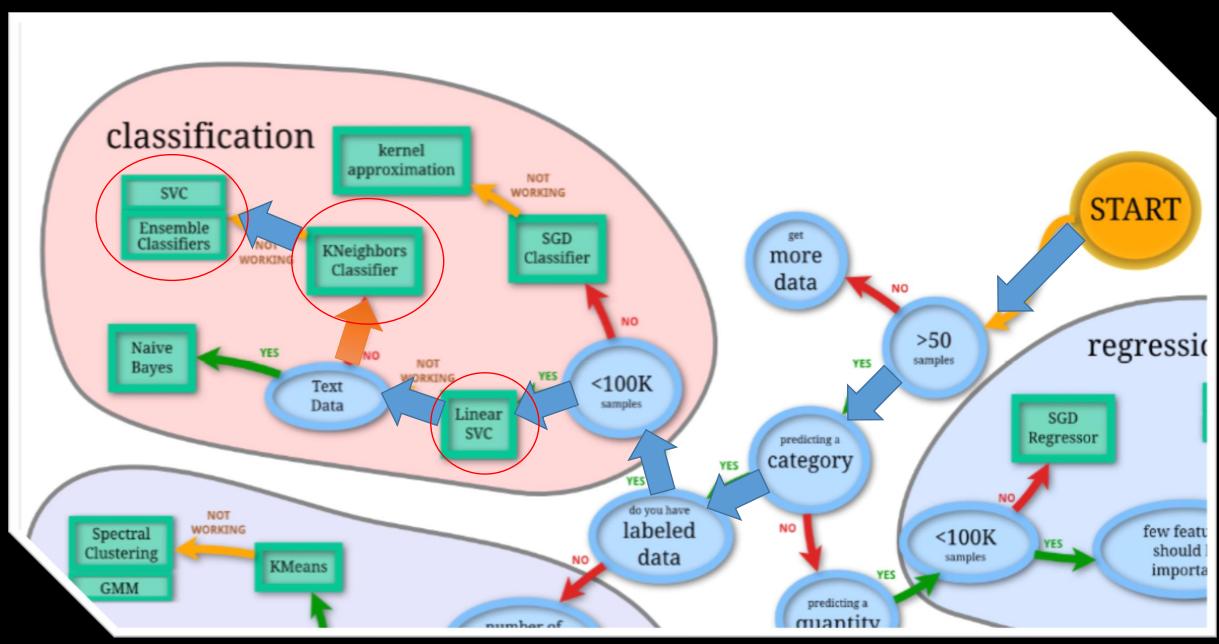


## 4 | 데이터 분석 – 다중회귀

### 실제 다른 영화 데이터 관객수 예측

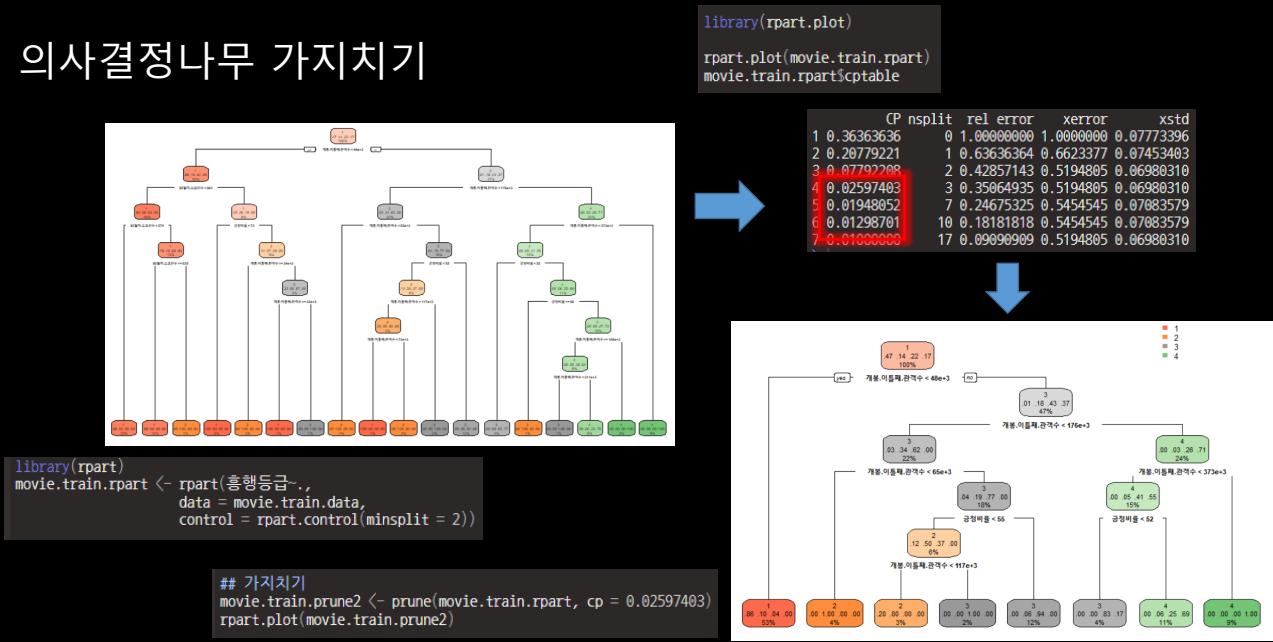
영화제목	예측 관객수	실제 관객수
레드슈즈	558,969	437,618
나랏말싸미	1,329,074	909,823
국가부도의날	2,339,033	2,093,097
베놈	3,409,138	2,755,750

## 4 | 데이터 분석 – 분류분석



## 4 | 데이터 분석 - 분류분석 - DT

### 의사결정나무 가지치기



## 4 | 데이터 분석 - 분류분석 - DT

## 예측하기  
predict(movie.train.prune2, movie.test.data, type='class')|

Confusion Matrix and Statistics				
Reference		Prediction		
Prediction	1	2	3	4
1	16	2	1	0
2	0	1	2	0
3	0	1	1	0
4	0	0	4	6

Overall Statistics				
Accuracy : 0.7059				95% CI : (0.5527, 0.849)
No Information Rate : 0.4706				P-Value [Acc > NIR] : 0.004738
Kappa : 0.555				
McNemar's Test P-Value : NA				
Statistics by Class:				
Class: 1 Class: 2 Class: 3 Class: 4				
Sensitivity	1.0000	0.25000	0.12500	1.0000
Specificity	0.8333	0.93333	0.96154	0.8571
Pos Pred Value	0.8421	0.33333	0.50000	0.6000
Neg Pred Value	1.0000	0.98233	0.78125	1.0000
Precision	0.8421	0.33333	0.50000	0.6000
Recall	1.0000	0.25000	0.12500	1.0000
F1	0.9143	0.28571	0.20000	0.7500
Prevalence	0.4706	0.11765	0.23529	0.1765
Detection Rate	0.4706	0.02941	0.02941	0.1765
Detection Prevalence	0.5588	0.08824	0.08824	0.2941
Balanced Accuracy	0.9167	0.59167	0.54327	0.9286

$$CP = 0.01948052$$

$$\text{Accuracy} = 0.7059$$

Confusion Matrix and Statistics				
Reference		Prediction		
Prediction	1	2	3	4
1	16	2	1	0
2	0	1	2	0
3	0	1	2	0
4	0	0	3	6

Overall Statistics				
Accuracy : 0.7353				95% CI : (0.5554, 0.8712)
No Information Rate : 0.4706				P-Value [Acc > NIR] : 0.001582
Kappa : 0.5984				
McNemar's Test P-Value : NA				
Statistics by Class:				
Class: 1 Class: 2 Class: 3 Class: 4				
Sensitivity	1.0000	0.25000	0.25000	1.0000
Specificity	0.8333	0.93333	0.96154	0.8571
Pos Pred Value	0.8421	0.33333	0.66667	0.66667
Neg Pred Value	1.0000	0.98233	0.88645	1.0000
Precision	0.8421	0.33333	0.66667	0.66667
Recall	1.0000	0.25000	0.25000	1.0000
F1	0.9143	0.28571	0.36364	0.8000
Prevalence	0.4706	0.11765	0.23529	0.1765
Detection Rate	0.4706	0.02941	0.02941	0.1765
Detection Prevalence	0.5588	0.08824	0.08824	0.2647
Balanced Accuracy	0.9167	0.59167	0.60577	0.9464

$$CP = 0.01298701$$

$$\text{Accuracy} = 0.7353$$

Confusion Matrix and Statistics				
Reference		Prediction		
Prediction	1	2	3	4
1	16	2	1	0
2	0	1	2	0
3	0	1	3	0
4	0	0	4	6

Overall Statistics				
Accuracy : 0.7647				95% CI : (0.5883, 0.8925)
No Information Rate : 0.4706				P-Value [Acc > NIR] : 0.0004606
Kappa : 0.6402				
McNemar's Test P-Value : NA				
Statistics by Class:				
Class: 1 Class: 2 Class: 3 Class: 4				
Sensitivity	1.0000	0.25000	0.25000	1.0000
Specificity	0.8333	0.93333	0.96154	0.8571
Pos Pred Value	0.8421	0.33333	0.66667	0.75000
Neg Pred Value	1.0000	0.98233	0.88645	1.0000
Precision	0.8421	0.33333	0.66667	0.66667
Recall	1.0000	0.25000	0.25000	1.0000
F1	0.9143	0.28571	0.36364	0.8000
Prevalence	0.4706	0.11765	0.23529	0.1765
Detection Rate	0.4706	0.02941	0.02941	0.1765
Detection Prevalence	0.5588	0.08824	0.08824	0.2647
Balanced Accuracy	0.9167	0.62500	0.66827	0.9286

$$CP = 0.02597403$$

$$\text{Accuracy} = 0.7647$$

## 4 | 데이터 분석 – 분류분석 - 랜덤포레스트

```
## 미니프로젝트 랜덤포레스트
library(randomForest)
## 랜덤포레스트 모델 생성
model <- randomForest(흥행등급 ~.,
                      data = movie.train.data,
                      ntree = 500,          ## 나무의 수
                      mtry = 4,             ## 기본 변수의 수
                      importance = TRUE)
model
```

```
Call:
randomForest(formula = 흥행등급 ~ ., data = movie.train.data,      ntree = 500, mtry = 4, importance = TRUE)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 4

OOB estimate of error rate: 26.39%
Confusion matrix:
  1 2 3 4 class.error
1 61 5 1 0 0.08955224
2 8 9 2 1 0.55000000
3 1 3 18 10 0.43750000
4 1 0 0 7 18 0.28000000
```

```
predTrain <- predict(model, movie.train.data, type = "class")
table(predTrain, movie.train.data$흥행등급)
```

```
predTrain 1 2 3 4
1 67 0 0 0
2 0 20 0 0
3 0 0 32 0
4 0 0 0 25
```

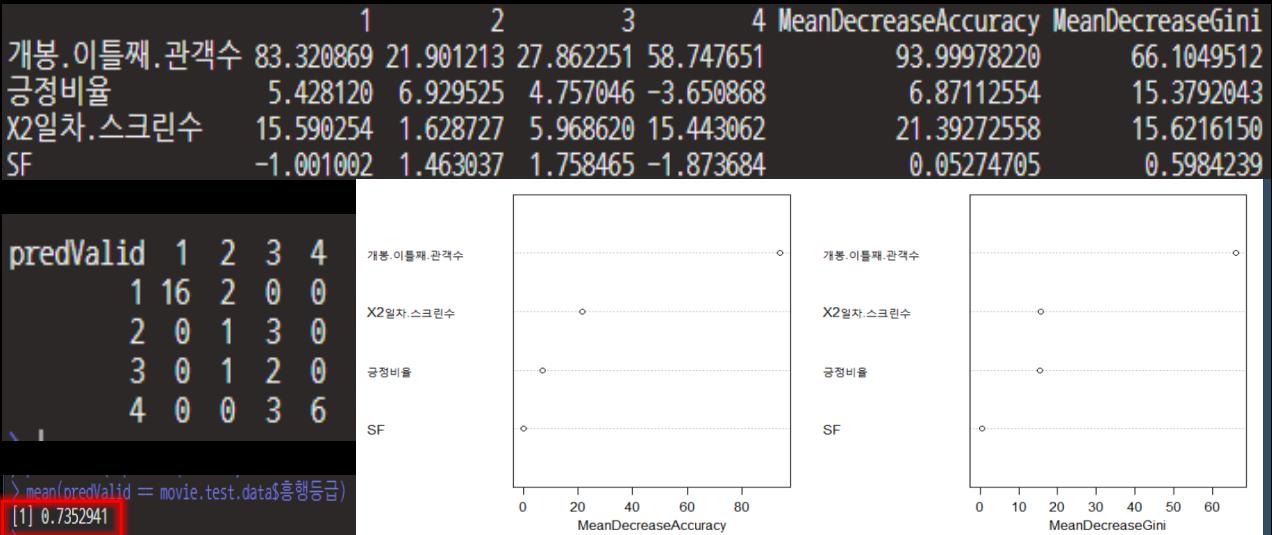
```
predValid <- predict(model, movie.test.data, type = "class")
table(predValid, movie.test.data$흥행등급)
```

```
predValid 1 2 3 4
1 14 0 0 0
2 2 2 2 0
3 0 2 5 0
4 0 0 1 6
```

```
> mean(predValid == movie.test.data$흥행등급)
[1] 0.7941176
```

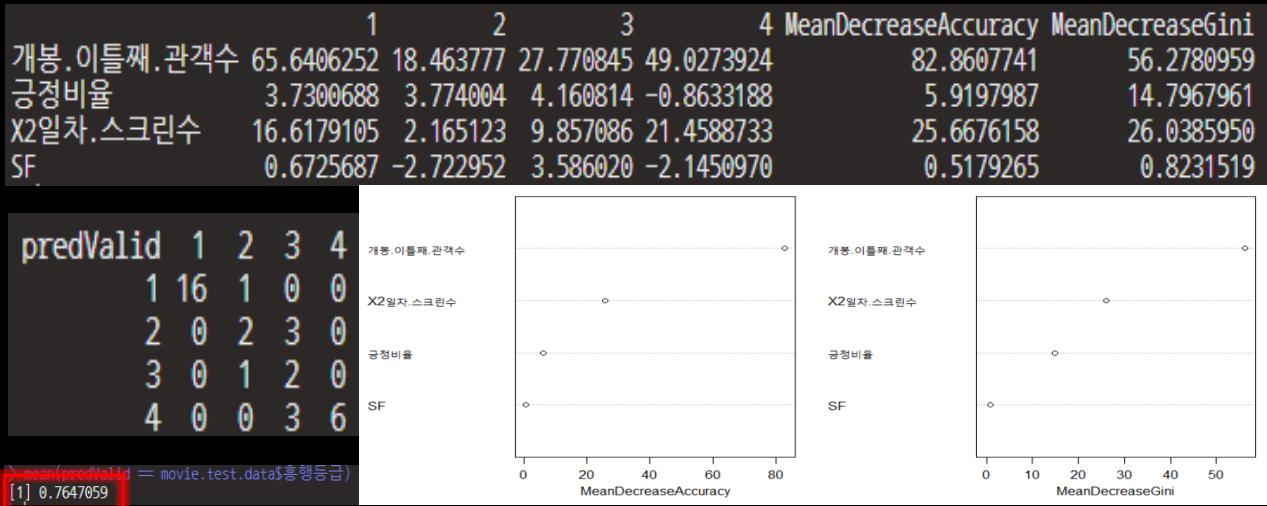
## 4 | 데이터 분석 – 분류분석 - 랜덤포레스트

[ mtry = 4 ]



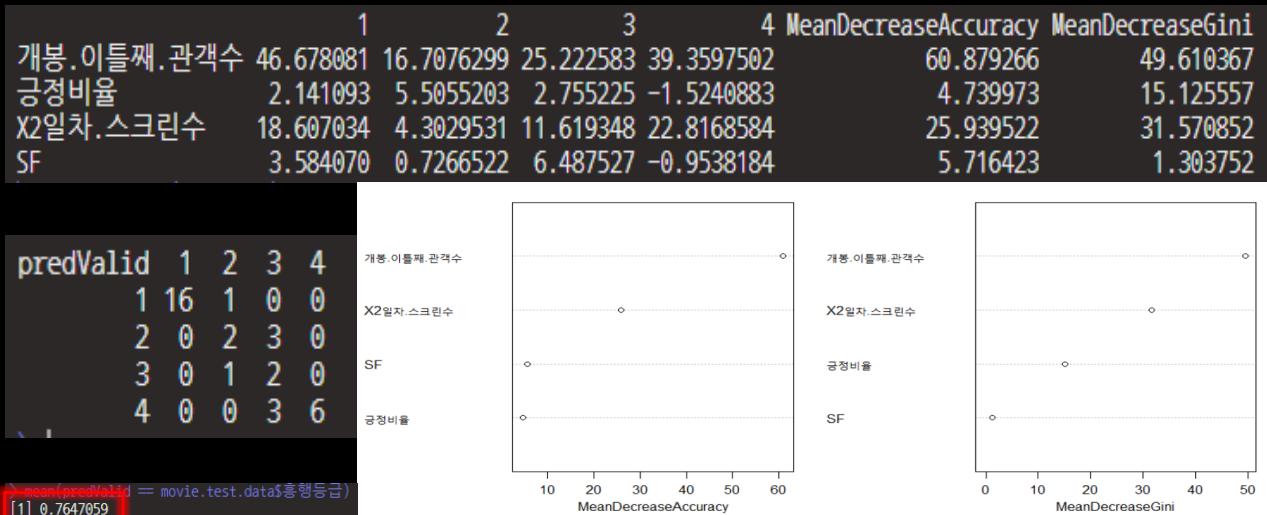
## 4 | 데이터 분석 – 분류분석 – 랜덤포레스트

[ mtry = 3 ]



## 4 | 데이터 분석 – 분류분석 – 랜덤포레스트

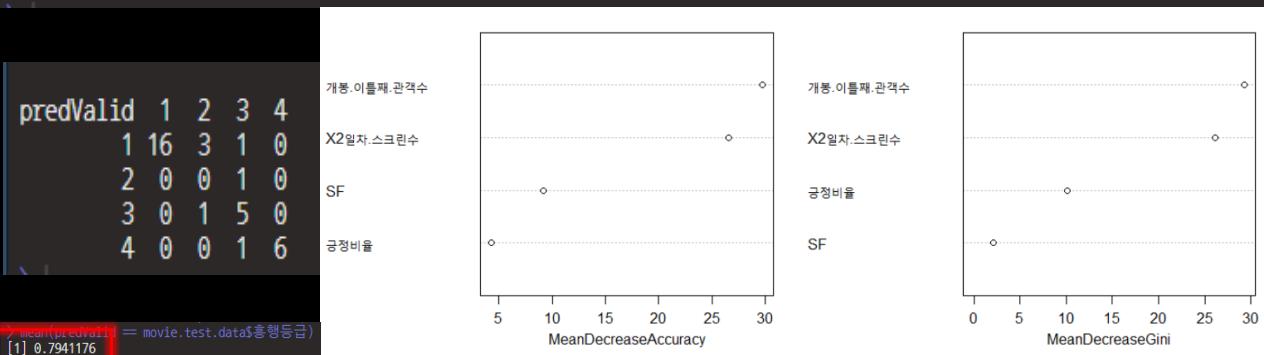
[ mtry = 2 ]



## 4 | 데이터 분석 – 분류분석 – 랜덤포레스트

```
[ mtry = 1 ]
```

	1	2	3	4	MeanDecreaseAccuracy	MeanDecreaseGini
개봉.이틀째.관객수	26.581927	7.6023273	16.596321	22.8620750	29.714321	29.330553
긍정비율	1.030003	4.9491530	2.481997	0.6358304	4.321289	10.075298
X2일자.스크린수	22.094229	1.6110384	15.301059	21.5904900	26.568740	26.163899
SF	9.637068	0.8549915	3.378190	3.3314283	9.161163	2.155953



## 4 | 데이터 분석 – 분류분석 – 분류 모델을 위한 전처리

```
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.ensemble import GradientBoostingClassifier

movie_df = pd.read_excel("종합.xlsx")
y = movie_df["총행등급"].values
X = movie_df[["긍정비율", "개봉.이틀째.관객수", "2일자.스크린수", "SF"]]

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

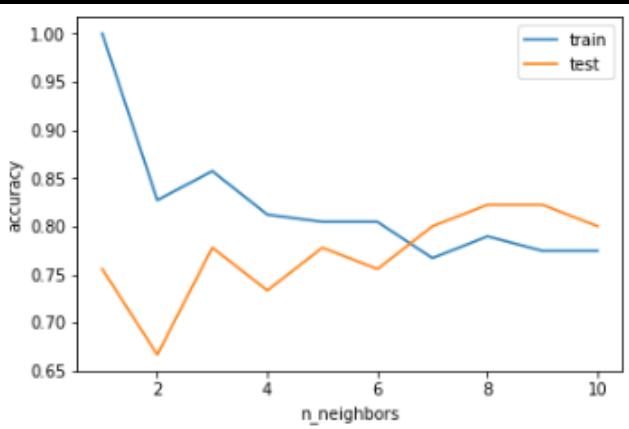
178개의 총 데이터를 133개의 Traindata와 45개의 Testdata로 나눔  
(75%: 25%)

## 4 | 데이터 분석 – 분류분석 – KNN

```
neighbors_settings = range(1, 11)

training_accuracy = []
test_accuracy = []
for n_neighbors in neighbors_settings:
    # 모델 생성
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)
    clf.fit(X_train, y_train)
    # 훈련 세트 정확도 저장
    training_accuracy.append(clf.score(X_train, y_train))
    # 테스트 세트 정확도 저장
    test_accuracy.append(clf.score(X_test, y_test))

plt.plot(neighbors_settings, training_accuracy, label="훈련 정확도")
plt.plot(neighbors_settings, test_accuracy, label="테스트 정확도")
plt.ylabel("정확도")
plt.xlabel("n_neighbors")
plt.legend()
plt.show()
```



## 4 | 데이터 분석 – 분류분석 – KNN

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_neighbors': [3,5,7,9,11],
    'weights': ['uniform','distance'],
    'metric': ['euclidean','manhattan']
}

clf = KNeighborsClassifier()
clf_grid = GridSearchCV(estimator=clf,param_grid = param_grid, cv=10)
clf_grid.fit(X_train, y_train)
print(clf_grid.best_params_)

{'metric': 'euclidean', 'n_neighbors': 11, 'weights': 'uniform'}
```

```
clf = KNeighborsClassifier(metric='euclidean', n_neighbors=11, weights='uniform')
clf.fit(X_train, y_train)
print("훈련 세트 정확도: {:.2f}".format(clf.score(X_train, y_train)))
print("테스트 세트 정확도: {:.2f}".format(clf.score(X_test, y_test)))
```

훈련 세트 정확도: 0.78  
테스트 세트 정확도: 0.82

## 4 | 데이터 분석 – 분류분석 – LinearSVC

변수선택법을 통해 뽑은 4가지 변수에 대한 LinearSVC 분류 모델

```
svm = LinearSVC()
svm.fit(X_train,y_train)
print("훈련 세트 정확도: {:.2f}".format(svm.score(X_train, y_train)))
print("테스트 세트 정확도: {:.2f}".format(svm.score(X_test, y_test)))

훈련 세트 정확도: 0.65
테스트 세트 정확도: 0.58
```

주요 2가지 변수에 대한 LinearSVC 분류 모델

```
svm = LinearSVC()
svm.fit(X2_train,y_train)
print("훈련 세트 정확도: {:.2f}".format(svm.score(X_train, y_train)))
print("테스트 세트 정확도: {:.2f}".format(svm.score(X_test, y_test)))

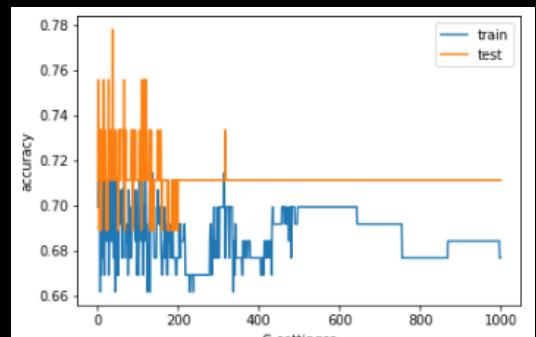
훈련 세트 정확도: 0.69
테스트 세트 정확도: 0.71
```

## 4 | 데이터 분석 – 분류분석 – Multi Logistic Regression

```
C_settings = range(1,1000)

training_accuracy = []
test_accuracy = []
for C_setting in C_settings:
    # 모델 생성
    softmax_reg = LogisticRegression(multi_class='multinomial',solver='lbfgs',C=C_setting,random_state=42)
    softmax_reg.fit(X_train, y_train)
    # 훈련 세트 정확도 저장
    training_accuracy.append(softmax_reg.score(X_train, y_train))
    # 일반화 정확도 저장
    test_accuracy.append(softmax_reg.score(X_test, y_test))

plt.plot(C_settings, training_accuracy, label="훈련 정확도")
plt.plot(C_settings, test_accuracy, label="테스트 정확도")
plt.ylabel("정확도")
plt.xlabel("C_settingss")
plt.legend()
plt.show()
```



## 4 | 데이터 분석 – 분류분석 – Multi Logistic Regression

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'solver': ['newton-cg', 'lbfgs', 'sag', 'saga'],
    'C' : [i for i in range(1,200)]
}

MLR = LogisticRegression(multi_class='multinomial')
MLR_grid = GridSearchCV(estimator=MLR,param_grid = param_grid, cv=10)
MLR_grid.fit(X_train, y_train)
print(MLR_grid.best_params_)

{'C': 40, 'solver': 'newton-cg'}

MLR = LogisticRegression(multi_class='multinomial',solver='newton-cg',C=40)
MLR.fit(X_train, y_train)
print("훈련 세트 정확도: {:.2f}".format(MLR.score(X_train, y_train)))
print("테스트 세트 정확도: {:.2f}".format(MLR.score(X_test, y_test)))

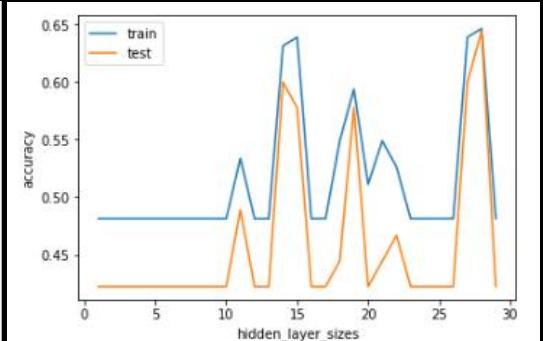
훈련 세트 정확도: 0.88
테스트 세트 정확도: 0.73
```

## 4 | 데이터 분석 – 분류분석 – MLP(Multi Layer Perceptron)

```
hidden_layer_sizes = range(1,30)

training_accuracy = []
test_accuracy = []
for hidden_layer_size in hidden_layer_sizes:
    # 모델 생성
    MLP = MLPClassifier(solver='lbfgs', activation='tanh',hidden_layer_sizes=[hidden_layer_size],hidden_layer_size)
    MLP.fit(X_train, y_train)
    # 훈련 세트 정확도 저장
    training_accuracy.append(MLP.score(X_train, y_train))
    # 일반화 정확도 저장
    test_accuracy.append(MLP.score(X_test, y_test))

plt.plot(hidden_layer_sizes, training_accuracy, label="train")
plt.plot(hidden_layer_sizes, test_accuracy, label="test")
plt.ylabel("accuracy")
plt.xlabel("hidden_layer_sizes")
plt.legend()
plt.show()
```



## 4 | 데이터 분석 – 분류분석 – MLP(Multi Layer Perceptron)

```
from sklearn.model_selection import GridSearchCV
layer_sizes = []
for i in range(1,31):
    layer_sizes.append([i,i])

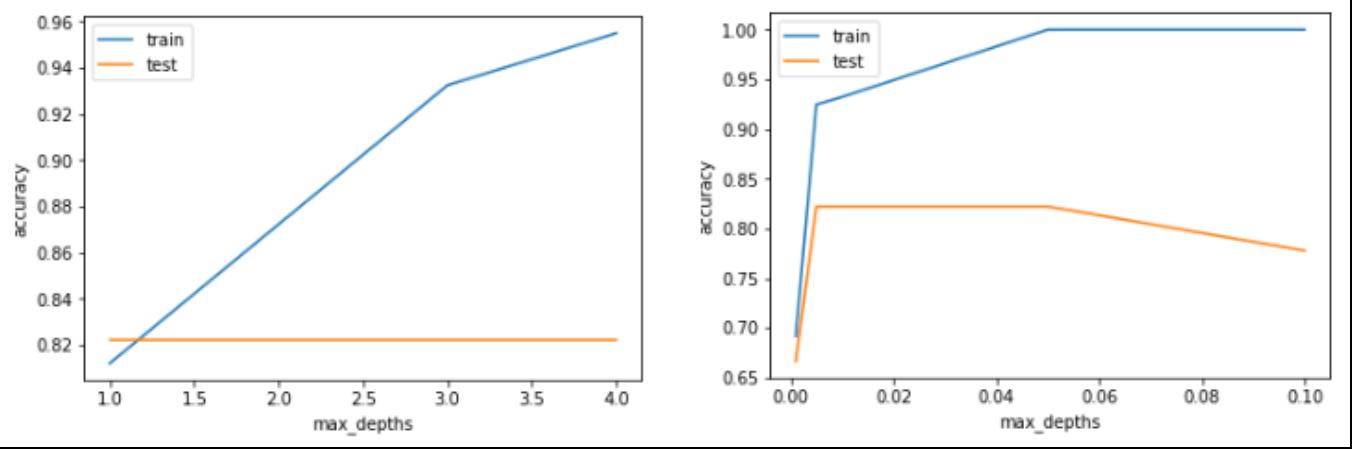
param_grid = {
    'hidden_layer_sizes' : layer_sizes,
    'max_iter' : [i*100 for i in range(10,21)],
    'solver' : ['lbfgs', 'adam', 'sgd'],
    'activation' : ['identity', 'logistic', 'tanh', 'relu']
}

MLP = MLPClassifier(alpha=1,random_state=0)
MLP_grid = GridSearchCV(estimator=MLP, param_grid = param_grid)
MLP_grid.fit(X_train, y_train)
print(MLP_grid.best_params_)

{'activation': 'relu', 'hidden_layer_sizes': [3, 3], 'max_iter': 1000, 'solver': 'lbfgs'}
MLP = MLPClassifier(alpha=1,solver='lbfgs',max_iter=1000,random_state=0,hidden_layer_sizes=[3,3],activation='relu')
MLP.fit(X_train, y_train)

print("훈련 세트 정확도: {:.3f}".format(MLP.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(MLP.score(X_test, y_test)))
훈련 세트 정확도: 0.820
테스트 세트 정확도: 0.889
```

## 4 | 데이터 분석 – 분류분석 – GradientBoosting



## 4 | 데이터 분석 – 분류분석 – GradientBoosting

```
from sklearn.model_selection import GridSearchCV

layer_sizes = []
for i in range(1,31):
    layer_sizes.append([i,i])

param_grid = {
    'hidden_layer_sizes' : layer_sizes,
    'max_iter' : [i*100 for i in range(10,21)],
    'solver' : ['lbfgs', 'adam', 'sgd'],
    'activation' : ['identity', 'logistic', 'tanh', 'relu']
}

MLP = MLPClassifier(alpha=1, random_state=0)
MLP_grid = GridSearchCV(estimator=MLP, param_grid = param_grid)
MLP_grid.fit(X_train, y_train)
print(MLP_grid.best_params_)

{'activation': 'tanh', 'hidden_layer_sizes': [9, 9], 'max_iter': 1200, 'solver': 'lbfgs'}
MLP = MLPClassifier(alpha=1,solver='lbfgs',max_iter=1200,random_state=0,hidden_layer_sizes=[9,9],activation='tanh')
MLP.fit(X_train, y_train)

print("훈련 세트 정확도: {:.3f}".format(MLP.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(MLP.score(X_test, y_test)))

훈련 세트 정확도: 0.744
테스트 세트 정확도: 0.667
```

## 4 | 데이터 분석 – 분류분석 – 모델 선택

Classifier	Accuracy(Test)
Decision Tree	0.76
Random Forest	0.794
KNN Classifier	0.82
LinearSVC	0.71
Multi Logistic Regression	0.73
MLP Classifier	0.667
Gradient Boosting	0.822

#### 4 | 데이터 분석 – 분류분석 – 실제 예측

Movie Name	Predict	Real
변신	3	3
광대들: 풍문조작단	3	3
커런트 워	1	2
마이펫의 이중생활 2	3	3
레드슈즈	3	3
분노의 질주: 흡스&쇼	4	4
우리집	1	1



개봉 후 2일간의 평점과 댓글이  
흥행에 관련이 있을까?



- > PCA와 변수 선택법을 통해 긍정비율  
이 흥행과 관련이 있다는 것을 알 수 있다.

평점과 댓글로 흥행 예측이  
가능하지 않을까?



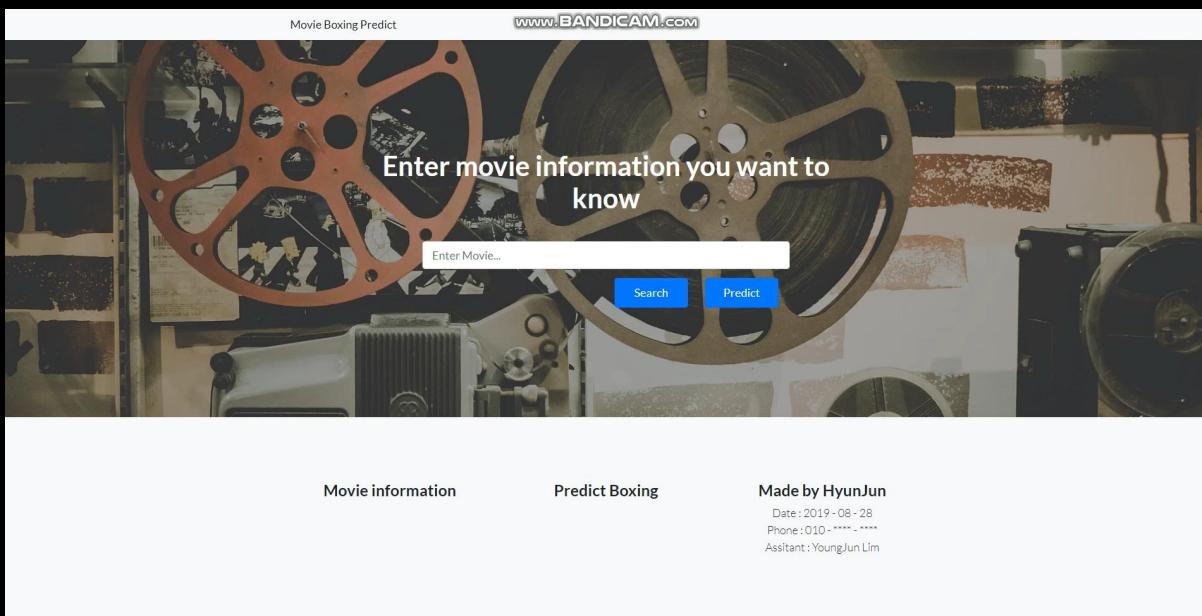
-> 댓글과 평점을 통한 긍정비율과 초기의 관객  
수 등으로 흥행 예측이 어느정도 가능하다.

## 한계점

- 리뷰, 별점 변수 외 기타 요인(배급사, 제작사 등)을 배제한 결과, 예측의 신뢰성이 다소 떨어진다는 점.
- 데이터 수집 및 정제 시간이 오래 걸려 많은 양의 데이터 수집을 하기에 어려움이 있었고 인스타그램, 트위터 등 SNS의 데이터를 활용하지 못한 점.
- 영화별 손익분기점, 제작비 등이 다른데 UBD라는 기준치 하나로 흥행 여부를 판단하여 실제 흥행의 여부가 불분명하다는 점.



## 6 웹 서비스 소개



## 7 참고문헌 / 분석도구

### 참고문헌

#입소문 #20대 #팬덤...CGV 2018 영화계 총결산,  
<노컷뉴스>, 2018-12-06, <https://www.nocutnews.co.kr/news/5072020>

CGV 리서치센터 "영화선택영향도 조사" 결과

영화 흥행 평가 새로운 척도, 'UBD'와 데이터 단위 변화의 이유,<한겨례>,  
2019-05-06, <http://www.hani.co.kr/arti/science/future/892738.html>

<https://github.com/e9t/nsmc/>

### 분석도구





Thank You