

2021-2학기 빅데이터 프로그래밍 TERM PROJECT

# 프로젝트 최종 보고서

ICT 3학년 60171672 황준하

# 목차

1. 프로젝트 주제
2. 실행 계획
3. 데이터 분석
4. 결론

## 1. 프로젝트 주제

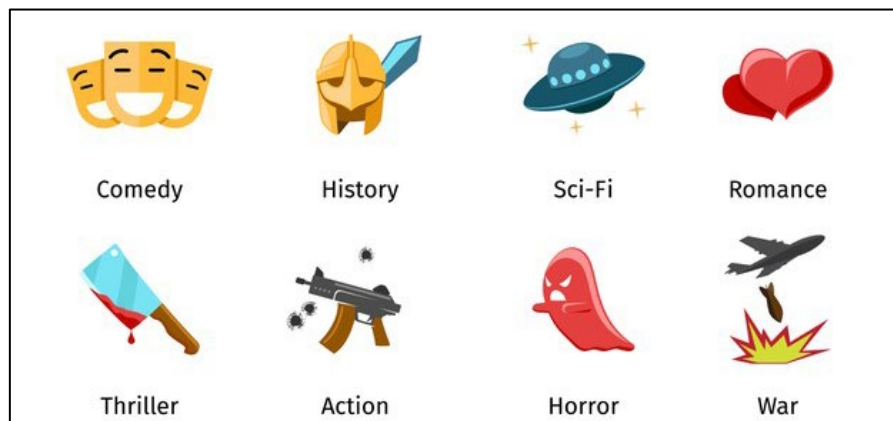
모바일 게임 중 <Game Dev Story> 라는 작품이 있다. 가상으로 게임을 개발하는 게임으로, 제작하는 과정에서 장르와 헤드 프로듀서, 음악 감독 등을 선정할 수 있다. 이들 사이에는 시너지라는 것이 존재하며, 이 시너지를 통해 예술성과 대중성을 높임으로써 높은 수익을 창출해낼 수 있다.



<Game Dev Story (출처: 구글 검색)>

영화를 좋아하는 사람으로써, <Game Dev Story> 내의 시너지처럼 관객이 영화를 관람하고 평가를 할 때 영향을 주는 요소를 수치화 할 수 있는지가 궁금하였다. 만약 **수치화가 되고, 시너지와 같은 특정한 패턴을 보인다면 높은 확률로 성공할 영화들을 제작할 수 있을 것이라** 생각한다.

이에 다라 프로젝트 주제를 **“영화의 평점에 영향을 미치는 요소”**에 대해 알아보고 싶었다. 영화에는 감독부터 배우, 시나리오, 장르 등 여러가지 카테고리가 존재한다. 그 카테고리 내에 어떤 항목이 가장 인기가 많고 어떠한 조합으로 영화를 만들면 성공의 확률이 높을지 알아보고자 한다.



<Movie Genre (출처: 구글 검색)>

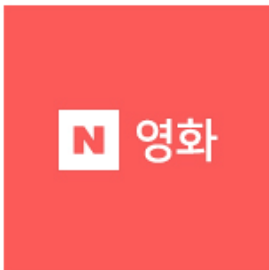
## 2. 실행 계획

영화에 대한 정보가 필요하기에 영화 데이터를 다운 받을 곳을 물색하였다. “Rotten Tomatoes”, “네이버 영화”, “IMDB” 로 총 세 곳으로 좁혀졌다. 영화 평론 관련 대표적인 사이트들이다.



### # Rotten Tomatoes

: Rotten Tomatoes는 영화가 얼마나 신선한지, 신선하지 못했는지에 중점을 두고 있는 사이트이기에 프로젝트와 맞지 않는 데이터를 갖고 있다고 판단하였다.



### # 네이버 영화

: 네이버 영화는 우리나라 사람들이 많이 사용하는 사이트이다. 이에 따라 한국 영화들에 편중될 수 밖에 없으며, 다양성이 떨어진다고 판단하였다.



### # IMDB

: IMDB는 영화 평론에서 신뢰성이 가장 높은 곳으로 객관적인 데이터 및 사람들의 리뷰와 평가들까지 존재하기 때문에 적합하다고 판단하였다.

이에 따라 IMDB 내의 영화 데이터들을 수집하고자 하였다. 필요한 데이터들은

- 영화에 대한 정보 / 장르, 평점, 시나리오 등
- 영화 제작자에 대한 정보 / 배우, 감독 등

으로 해당 정보들을 조합하여 식을 세워 분석하고자 한다.

## Feature Film, User Rating between 1 and 10, Rating Count at least 10,000 (Sorted by Popularity Ascending)

1-50 of 9,353 titles. | [Next »](#)

View Mode: [Compact](#) | [Detailed](#)

Sort by: [Popularity▲](#) | [A-Z](#) | [User Rating](#) | [Number of Votes](#) | [US Box Office](#) | [Runtime](#) | [Year](#) | [Release Date](#) | [Date of Your Rating](#) | [Your Rating](#)



### 1. **Eternals** (2021)

PG-13 | 156 min | Action, Adventure, Fantasy



6.9



[Rate this](#)

52

Metascore

The saga of the Eternals, a race of immortal beings who lived on Earth and shaped its history and civilizations.

Director: [Chloé Zhao](#) | Stars: [Gemma Chan](#), [Richard Madden](#), [Angelina Jolie](#), [Salma Hayek](#)

Votes: 81,362



### 2. **Dune** (2021)

PG-13 | 155 min | Action, Adventure, Drama



8.2



[Rate this](#)

74

Metascore

Feature adaptation of Frank Herbert's science fiction novel, about the son of a noble family entrusted with the protection of the most valuable asset and most vital element in the galaxy.

Director: [Denis Villeneuve](#) | Stars: [Timothée Chalamet](#), [Rebecca Ferguson](#), [Zendaya](#), [Oscar Isaac](#)

Votes: 313,011



<TOP 1000 Sorted by Number of Votes Ascending /

User Rating between 1 and 7, Rating Count at least 20,000

(Sorted by Popularity Ascending (출처: IMDB)>

IMDB 내에 해당 데이터를 다운 받고자 하였다. IMDB 검색기를 통해 해당 조건들을 설정하였다.

#### 1. Feature Film

: 프로젝트 주제가 “영화”이기에 영화로 설정하였다.

#### 2. User Rating between 1 and 10

: 평점이 한쪽으로 편중되지 않게 1과 10 사이로 설정하였다.

#### 3. Rating Count at least 10,000

: 신뢰도를 위하여 Rating Count 가 최소 만개는 있는 어느 정도 인지도가 있는 영화로 설정하였다.

데이터 다운로드 같은 경우 **Python Selenium**을 사용하였다.

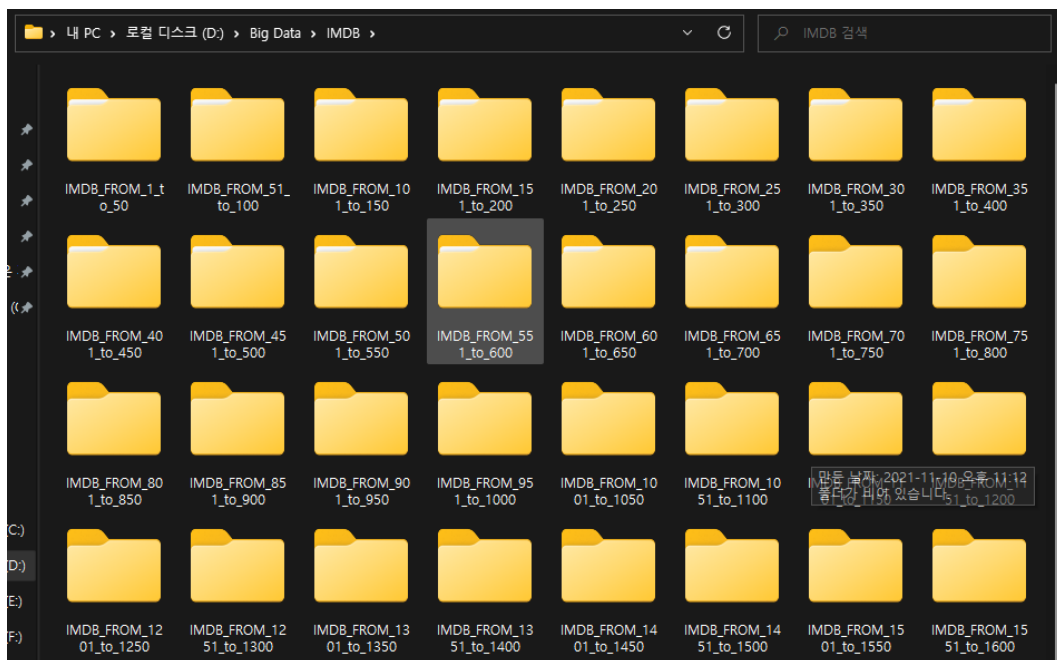


<Selenium (출처: 구글 검색)>

Selenium을 이용해 해당 사이트를 접속하고 영화에 대한 데이터들을 다운 받았다. 다운 받은 데이터들을 아래와 같다.

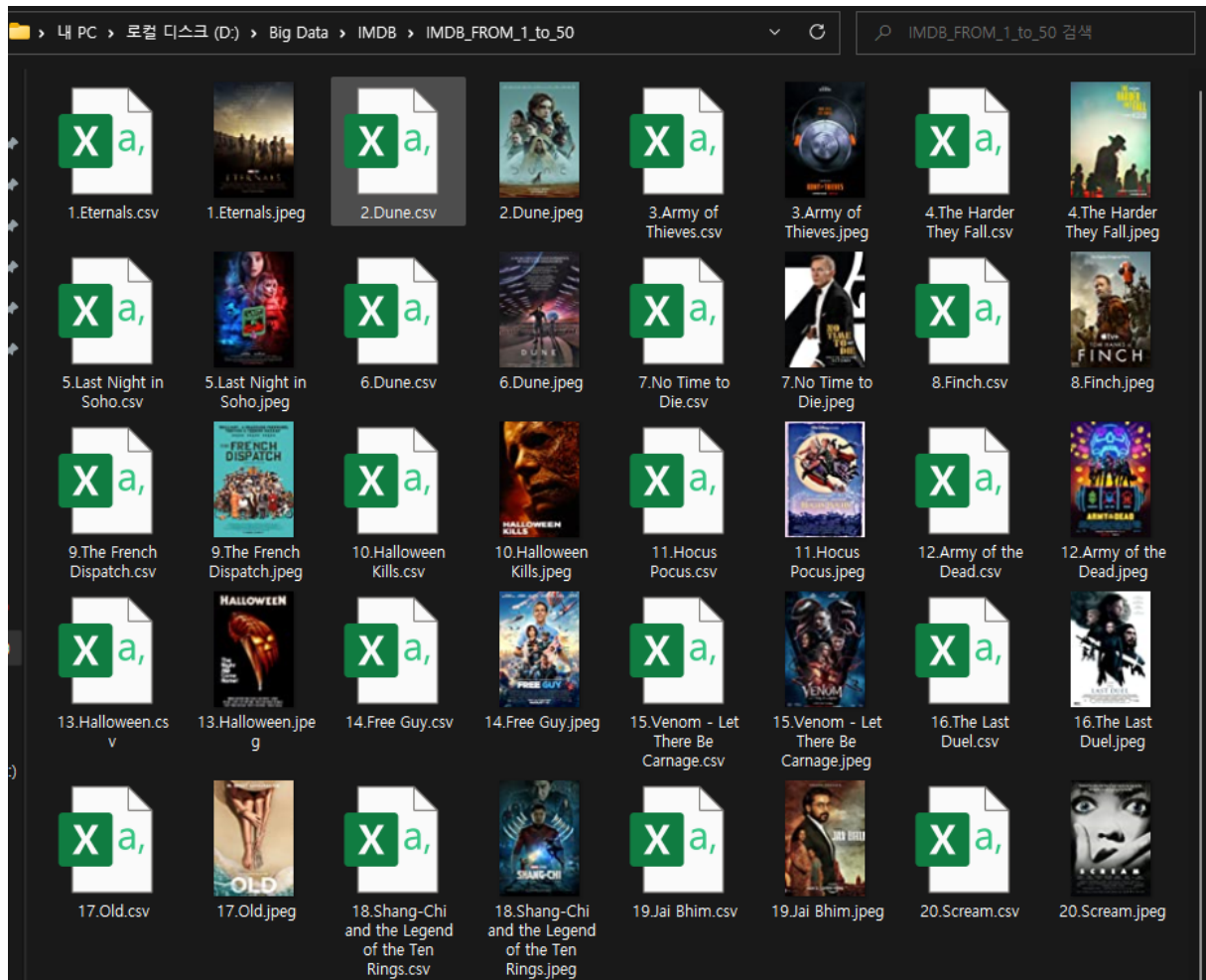
#### 1. 50개 단위로 영화들을 분할 저장

: 영화 한편의 데이터를 받는데 너무 오랜 시간이 걸리기에 작업이 중단되더라도 최소한의 데이터를 살리기 위하여 50개의 단위로 잘랐다.



## 2. 영화 포스터 및 영화 리뷰

: 영화 포스터 및 관람객들이 남긴 영화 리뷰를 다운로드 하였다. 파일 형식은 jpeg와 csv이다.



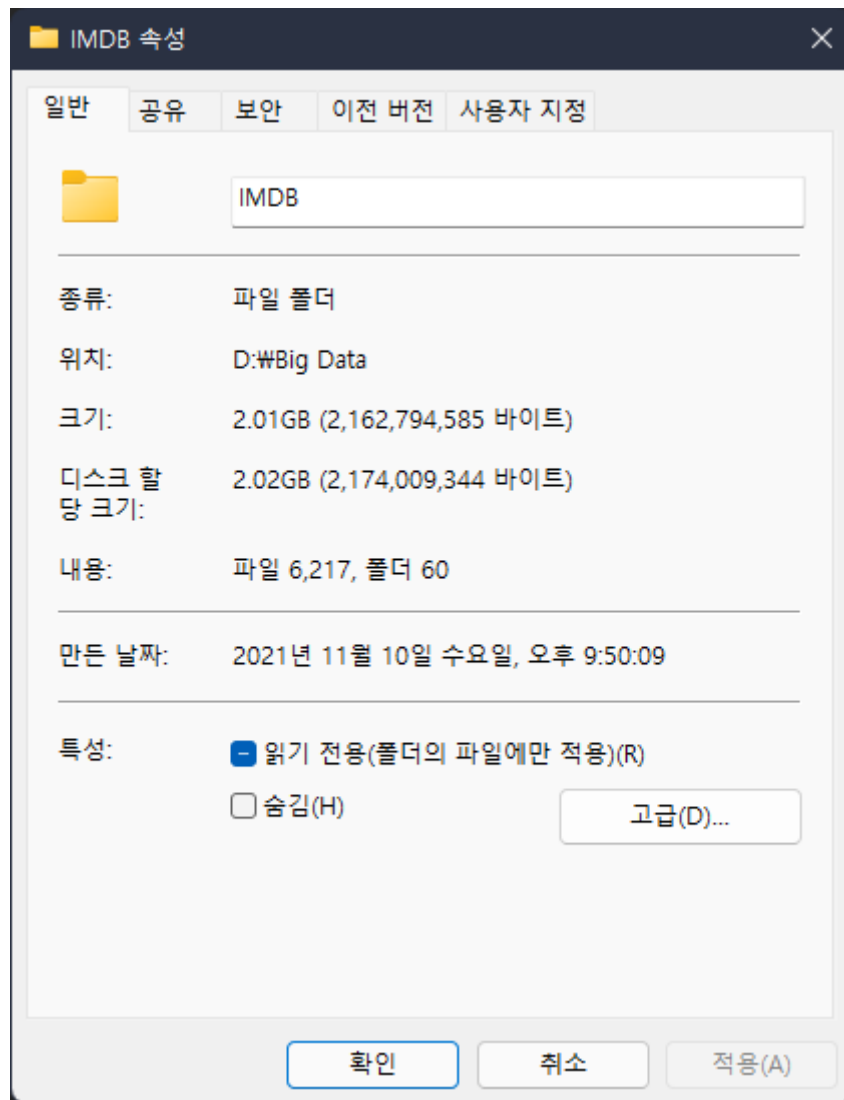
## 3. 50편에 대한 영화 정보

: 50편에 대한 영화 정보들이 담긴 파일이다. 제목부터 개봉연도, 장르, 평점, 투표 수, 감독, 배우, 시나리오, 매출, 관람연령 등이 존재한다.

A	B	C	D	E	F	G	H	I	J	K	L
	NO	Movie_title	Year	Genre	Rating	Votes	Directors	Stars	Scenario	Gross	Certificate
0	1	Eternals	-2021	[Action, A	6.9	66425	[Chloé Zh	[Gemma	[The saga	NA	PG-13
1	2	Dune	-2021	[Action, A	8.2	299702	[Denis Vil	[Timothée	[Feature	NA	PG-13
2	3	Army of T	-2021	[Action, C	6.4	32206	[Matthias	[Matthias	[A preque	NA	TV-MA
3	4	The Harde	-2021	[Action, D	6.6	21795	[Jeymes S	[Jonathan	[When an	NA	R
4	5	Last Night	-2021	[Drama, F	7.5	18035	[Edgar Wi	[Thomasir	[An aspiri	NA	R
5	6	Dune	-1984	[Action, A	6.4	150182	[David Ly	[Kyle Mac	[A Duke's	\$30.93M	PG-13
6	7	No Time t	-2021	[Action, A	7.5	155307	[Cary Joji	[Daniel C	[James Bc	NA	PG-13
7	8	Finch	-2021	[Adventur	7	21230	[Miguel S	[Tom Har	[On a po	NA	PG-13
8	9	The Frencl	-2021	[Comedy,	7.5	18215	[Wes And	[Benicio	[A love le	NA	R
9	10	Halloween	-2021	[Horror, T	5.7	54018	[David Gc	[Jamie Le	[The saga	NA	R

<movie\_info.csv>

다운받은 데이터들은 총 2기가로 3,000편의 영화에 대한 정보를 담고 있으며, 약 3백만의 리뷰 데이터를 다운받았다. 데이터 분석을 하기에 충분한 용량이라 판단하였다.



<데이터를 다운로드한 폴더>



## # FULL CODE

```
Projects > 빅데이터프로그래밍 > 📁 IMDB.py > 📄 get_review
1  -*- coding: utf-8 -*-
2  import pandas as pd
3  import selenium
4  from selenium import webdriver
5  from webdriver_manager.chrome import ChromeDriverManager
6  import requests
7  from bs4 import BeautifulSoup
8  from selenium.webdriver.chrome.options import Options
9  import time
10 import urllib.request
11 from selenium.webdriver.common.keys import Keys
12 import warnings
13 import datetime
14
15 # PATH 설정
16 PATH = r"D:\Big Data\chromedriver.exe"
17
18 warnings.filterwarnings(action='ignore')
19
20 def get_review(url, folder_name, no, no_):
21
22     # 경과 시간 확인을 위한 설정
23     start = time.time()
24
25     # 시간 확인용 현재 시간 설정
26     now = datetime.datetime.now()
27     nowDatetime = now.strftime('%Y-%m-%d %H:%M:%S')
28
29     # SELENIUM
30     options = webdriver.ChromeOptions()
31     options.add_argument("headless")
32     driver = webdriver.Chrome(PATH, options=options)
33     driver.get(url)
34     driver.implicitly_wait(1)
35
36
37     # LOGIN / 한국어로 나오는 제목 존재
38     driver.find_element_by_css_selector("div.ipc-page-content-container.ipc-page-content")
39     driver.find_element_by_class_name("auth-provider-text").click()
40     driver.find_element_by_id('ap_email').send_keys('hwangjoon0@naver.com')
41     driver.find_element_by_id('ap_password').send_keys('ghkdwns0')
42     driver.find_element_by_css_selector("div:nth-child(2) > div > div > form > div > div")
43     driver.find_element_by_id('signInSubmit').click()
44     driver.execute_script('window.open("about:blank", "_blank");')
45     driver.get(url)
46     driver.switch_to.window((driver.window_handles[0]))
47
48     # 이미지 로딩을 위한 위아래 스크롤
49     for i in range(0,30):
50         try:
51             driver.find_element_by_tag_name('body').send_keys(Keys.PAGE_DOWN)
52         except:
53             continue
54
55     for i in range(0,30):
56         try:
57             driver.find_element_by_tag_name('body').send_keys(Keys.PAGE_UP)
58         except:
59             continue
```

```

62     # 정보 저장을 위한 list
63     NO = []
64     title = []
65     titles = []
66     link = []
67     year = []
68     genre = []
69     rate = []
70     votes = []
71     directors = []
72     stars = []
73     story = []
74     gross = []
75     url = []
76     certificate = []
77
78
79     # 영화 한편을 단위로 설정
80     block = driver.find_elements_by_class_name('list-item')
81
82     for i in range(0,50):
83
84         ftitle = block[i].find_element_by_class_name('list-item-header').text
85
86         # 순서
87         try:
88             forder = block[i].find_element_by_class_name('list-item-index').text
89         except:
90             forder = "NA"
91
92         # 연도
93         try:
94             fyear = ftitle[-6:]
95             #Drop the order, year and only keep the movie's name
96         except:
97             fyear = "NA"
98
99         # 제목
100        try:
101            ftitles = block[i].find_element_by_css_selector('h3 > a').text
102            try:
103                ftitles = ftitles.replace(":", " -")
104                ftitles = ftitles.replace("?", "")
105                ftitles = ftitles.replace("/", " -")
106            except:
107                continue
108        except:
109            ftitles = "NA"
110
111        # 리뷰 링크
112        try:
113            #Then extract the link with cleaned title
114            flink = block[i].find_element_by_css_selector('h3 > a').get_attribute('href')
115        except:
116            flink = "NA"

```

```

118     # 장르
119     try:
120         fgenre = block[i].find_element_by_class_name('genre').text
121     except:
122         fgenre = "NA"
123
124     # 평점
125     try:
126         frates = block[i].find_element_by_css_selector('.ratings-imdb-rating').find_e
127     except:
128         frates = "NA"
129
130     # 투표 수
131     try:
132         fvotes = block[i].find_elements_by_name('nv')[0].get_attribute('data-value')
133     except:
134         fvotes = "NA"
135
136     # 감독 및 배우
137     try:
138         f_info = block[i].find_element_by_css_selector('p:nth-child(5)').text
139         f_info_1 = f_info.split("|")
140         fdirectors = f_info_1[0].replace("Director: ", "")
141         fstars = f_info_1[1].replace(" Stars: ", "")
142     except:
143         fdirectors = "NA"
144         fstars = "NA"
145
146     # 줄거리
147     try:
148         fstory = block[i].find_element_by_css_selector('p:nth-child(4)').text
149     except:
150         fstory = "NA"
151
152     # 연령 제한
153     try:
154         fcertificate = block[i].find_element_by_class_name("certificate").text
155     except:
156         fcertificate = "NA"
157
158     # 매출
159     try:
160         fgross = block[i].find_element_by_css_selector('p.sort-num_votes-visible > sp
161     except:
162         fgross = "NA"
163
164     # 포스터/이미지
165     try:
166         fimage = block[i].find_element_by_class_name("loadlate")
167         url.append(fimage.get_attribute("src"))
168     except:
169         continue

```

```

171         NO.append(no)
172         no +=1
173         title.append(ftitles)
174         titles.append(str(ftitles))
175         year.append(fyear)
176         link.append(flink)
177
178         genre.append([fgenre])
179         rate.append(frates)
180         votes.append(fvotes)
181         directors.append([fdirectors])
182         stars.append([fstars])
183         story.append([fstory])
184         gross.append(fgross)
185         certificate.append(fcertificate)
186
187     print("\n",len(NO)," Movies LOADED")
188     print(nowDatetime)
189
190     # 포스터/이미지 저장
191     for i in range(len(url)):
192         urllib.request.urlretrieve(url[i], f'D:/Big Data/IMDB/{folder_name}/{str(NO[i])}')
193
194     # 경과 시간 알리기 용
195     print("movie_info COMPLETE : ",round((time.time() - start)/60,3),"MIN\n")
196
197     # 리뷰 링크
198     user_review_links = []
199     for url in link:
200         review_link = url.replace("?ref=adv_li_tt","reviews?ref=tt_urv")
201         user_review_links.append(review_link)
202
203     # DataFrame 생성
204     top_data = {'Movie_name': title,
205                'Year': year,
206                'link': link,
207                'user_review' : user_review_links,
208                }
209     top = pd.DataFrame(data = top_data) #create dataframe
210     driver.quit() #tell Selenium to close the webpage
211
212     # DataFrame 생성
213     movie_inf = {'NO' : NO,
214                 'Movie_title': titles,
215                 'Year': year,
216                 'Genre': genre,
217                 'Rating' : rate,
218                 'Votes' : votes,
219                 'Directors' : directors,
220                 'Stars' : stars,
221                 'Scenario' : story,
222                 'Gross' : gross,
223                 'Certificate' : certificate
224                 }
225

```

```

226 movie_info = pd.DataFrame(data = movie_inf)
227 movie_info.to_csv(f'D:/Big Data/IMDB/{folder_name}/movie_info.csv',encoding='utf-8-si
228
229
230 # 유저 리뷰 가져오기
231 for i in range(len(top['user_review'])):
232     options = webdriver.ChromeOptions()
233     options.add_argument("headless")
234     options.add_argument('window-size=1920x1080')
235     options.add_argument("disable-gpu")
236     driver = webdriver.Chrome(PATH, options=options)
237     driver.get(top['user_review'][i])
238     driver.implicitly_wait(1)
239
240
241 # LOAD MORE 를 위한 설정
242 page = 1
243 while page < 10000:
244     try:
245         load_more = driver.find_element_by_id('load-more-trigger')
246         load_more.click()
247         page+=1
248         driver.implicitly_wait(3)
249     except:
250         print("\n",page," pages LOAD COMPLETE : ",round((time.time() - start)/60,
251         break
252
253 review = driver.find_elements_by_class_name('review-container')
254 title = []
255 content = []
256 rating = []
257 date = []
258 user_name = []
259 for n in range(0,100000):
260     try:
261         ftitle = review[n].find_element_by_class_name('title').text
262         try:
263             fcontent = review[n].find_element_by_class_name('text.show-more__cont
264         except:
265             fcontent = ftitle
266         frating = review[n].find_element_by_css_selector('div.lister-item-content
267         fdate = review[n].find_element_by_class_name('review-date').text
268         fname = review[n].find_element_by_class_name('display-name-link').text
269
270         title.append(ftitle)
271         content.append(fcontent)
272         rating.append(frating)
273         date.append(fdate)
274         user_name.append(fname)
275         print(no_,":",top['Movie_name'][i],"s ",n+1," REVIEWS LOADED ",round((ti
276     except:
277         continue
278
279 data = {'User_name': user_name,
280        'Review title': title,
281        'Review Rating': rating,
282        'Review date' : date,
283        'Review_body' : content
284        }
285

```

```

286     print("\n",no_,":",top['Movie_name'][i], "review LOAD COMPLETED",n+1,round((time.
287     print(nowDatetime)
288
289     review = pd.DataFrame(data = data)
290     movie = top['Movie_name'][i]
291     review['Movie_name'] = movie
292     review.to_csv(f'D:/Big Data/IMDB/{folder_name}/{no_}.{movie}.csv',encoding='utf-8
293     no_+=1
294     driver.quit()
295
296     # 실행
297     for i in range(1,2951,50):
298         IMDB = 'https://www.imdb.com/search/title/?title_type=feature&user_rating=1.0,10.0&nu
299         get_review(IMDB, ('IMDB_FROM_'+str(i)+'_to_'+str(i+49)),int(i),int(i))

```

그 이후 50편의 영화 정보들이 담긴 각 폴더 내 movie\_info.csv 파일들을 하나의 파일로 concat 및 parsing을 진행하여 "IMDB\_movie\_info.csv"로 만들었다.

```

Projects > 빅데이터프로그래밍 > put_together.py
1  import pandas as pd
2  import glob
3  import os
4  import re
5
6  allData = []
7  output_file = r'D:/Big Data/IMDB/IMDB_movie_info.csv'
8
9  print("작업 시작")
10
11 for i in range(1,3001,50):
12     #input_file = r'D:/Big Data/IMDB/IMDB_FROM_%d_to_%d/test'
13     j = i+49
14     print("IMDB FROM {0} TO {1} STARTED".format(i,j))
15     input_file = "D:/Big Data/IMDB/IMDB_FROM_{0}_to_{1}/movie_info.csv".format(i,j)
16     df = pd.read_csv(input_file)
17     df = df.drop([df.columns[0]],axis=1)
18     df["Directors"] = df["Directors"].str.replace("'", "")
19     df["Directors"] = df["Directors"].str.replace('[', "")
20     df["Directors"] = df["Directors"].str.replace(']', "")
21     df["Directors"] = df["Directors"].str.replace('Directors: ', "")
22     df["Year"] = df["Year"].str.replace('(', "")
23     df["Year"] = df["Year"].str.replace(')', "")
24     df["Year"] = pd.to_numeric(df["Year"])
25     df["Genre"] = df["Genre"].str.replace("'", "")
26     df["Genre"] = df["Genre"].str.replace('[', "")
27     df["Genre"] = df["Genre"].str.replace(']', "")
28     df["Stars"] = df["Stars"].str.replace("'", "")
29     df["Stars"] = df["Stars"].str.replace('[', "")
30     df["Stars"] = df["Stars"].str.replace(']', "")
31     df["Scenario"] = df["Scenario"].str.replace("'", "")
32     df["Scenario"] = df["Scenario"].str.replace('[', "")
33     df["Scenario"] = df["Scenario"].str.replace(']', "")
34     df["Scenario"] = df["Scenario"].str.replace(' ', "")
35     allData.append(df)
36     print("Work Finished")
37
38
39
40 dataCombine = pd.concat(allData, ignore_index=True)
41 dataCombine.to_csv(output_file)

```

NO	Movie_title	Year	Genre	Rating	Votes	Directors	Stars	Scenario	Gross	Certificat
1	Eternals	2021	Action, Ad	6.9	66425	Chlo채 Zh	Gemma C	The saga of the Etern		PG-13
2	Dune	2021	Action, Ad	8.2	299702	Denis Ville	Timoth채	Feature adaptation c		PG-13
3	Army of Tl	2021	Action, Co	6.4	32206	Matthias S	Matthias S	A prequel, set before		TV-MA
4	The Harde	2021	Action, Dr	6.6	21795	Jeymes Sa	Jonathan	When an outlaw disc		R
5	Last Night	2021	Drama, Ho	7.5	18035	Edgar Wri	Thomasin	An aspiring fashion c		R
6	Dune	1984	Action, Ad	6.4	150182	David Lyn	Kyle MacL	A Dukes s	\$30.93M	PG-13

<IMDB\_movie\_info>

### 3. 데이터 분석

데이터 분석 방법으로는 학술지 평가를 위한 인용분석 자료로 사용되는 영향력 지수(Impact Factor : IF)를 참고하였다.

## ≡ InCites Journal Citation Reports: Impact Factor 계산방식

### InCites Journal Citation Reports: Impact Factor 계산방식

🕒 6월 27, 2018 · Knowledge

#### Article

학술지 평가를 위한 인용분석 자료로 영향력 지수(Impact Factor : IF)를 주로 이용하는데, 특정 기간 동안 한 학술지에 수록된 하나의 논문이 다른 논문에 인용된 평균 횟수로서, 산출을 위한 공식은 다음과 같다.

영향력 계수(IF) = 학술지의 논문이 인용된 총 횟수 / 학술지에 수록된 논문의 수

예를 들어 InCites Journal Citation Reports에서 영향력 지수는 다음과 같이 계산된다.

Journal : A, Impact Factor : 1.489

2010 년 출판된 논문들에서 2008, 2009년에 출판된 "A"의 논문들이 인용된 횟수 :

2008년 = 214회  
+ 2009년 = 112회  
= 326회 <--- 피인용횟수

2008, 2009 년에 출판된 "A"의 전체 논문수 :

2008년 = 107건  
+ 2009년 = 112건  
= 219건 <--- 총수록논문수

IF계산  $\Rightarrow$  피인용횟수/총수록논문수 =  $326/219 = 1.489$

즉 학술지 A의 영향력 지수는 1.489이다.

<Impact Factor 계산방식>

<https://support.clarivate.com/ScientificandAcademicResearch/s/article/InCites-Journal-Citation-Reports-Impact-Factor-%EA%B3%84%EC%82%B0%EB%B0%A9%EC%8B%9D?language=ko>

#### ● 영화 영향력 지수

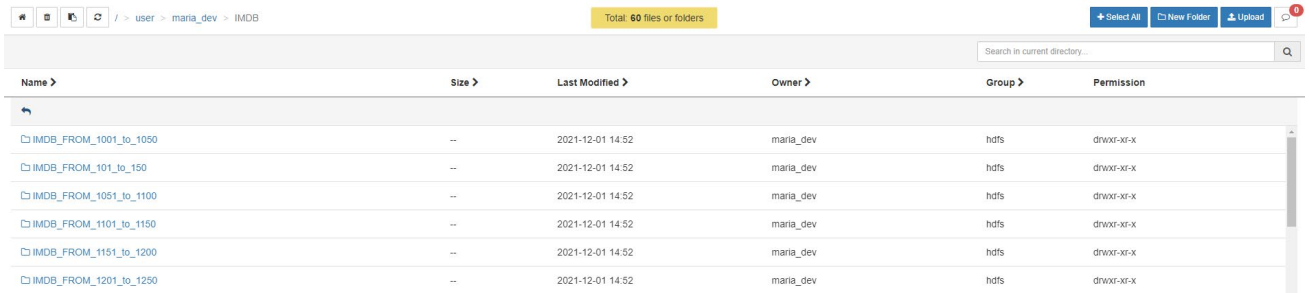
- VOTES : 영화 "A"가 받은 투표 수 / 같은 연도에 개봉한 영화들이 받은 영화들의 총 투표 수

- RATING : 영화 "A"가 받은 평점 / 같은 연도에 개봉한 영화들이 받은 평점들의 총합

=> 영화 "A"의 영향력 지수 = VOTES + RATING



데이터 분석을 위하여 해당 데이터들을 Ambari 내에 maria\_dev에 업로드하였다.



The screenshot shows the Ambari file browser interface. At the top, the breadcrumb path is "/ > user > maria\_dev > IMDB". A yellow status bar indicates "Total: 60 files or folders". On the right, there are buttons for "Select All", "New Folder", "Upload", and a notification icon. Below this is a search bar with the placeholder text "Search in current directory...". The main area is a table with columns: Name, Size, Last Modified, Owner, Group, and Permission. The table lists six files, all named "IMDB\_FROM\_XXXX\_to\_YYYY", with sizes of "--", last modified dates of "2021-12-01 14:52", owners of "maria\_dev", groups of "hdfs", and permissions of "drwxr-xr-x".

Name >	Size >	Last Modified >	Owner >	Group >	Permission
IMDB_FROM_1001_to_1050	--	2021-12-01 14:52	maria_dev	hdfs	drwxr-xr-x
IMDB_FROM_101_to_150	--	2021-12-01 14:52	maria_dev	hdfs	drwxr-xr-x
IMDB_FROM_1051_to_1100	--	2021-12-01 14:52	maria_dev	hdfs	drwxr-xr-x
IMDB_FROM_1101_to_1150	--	2021-12-01 14:52	maria_dev	hdfs	drwxr-xr-x
IMDB_FROM_1151_to_1200	--	2021-12-01 14:52	maria_dev	hdfs	drwxr-xr-x
IMDB_FROM_1201_to_1250	--	2021-12-01 14:52	maria_dev	hdfs	drwxr-xr-x

그 후 HIVE를 통하여 분석을 진행하였다.

```
1 DROP TABLE IMDB_YEAR;
2 CREATE TABLE IMDB_YEAR(
3   YEAR INT,
4   RATING FLOAT,
5   VOTES INT
6 );
7
8 INSERT INTO IMDB_YEAR
9 SELECT I.YEAR, SUM(I.RATING), SUM(I.VOTES) FROM imdb_movie_info AS I GROUP BY I.YEAR HAVING COUNT(I.YEAR) > 100;
10
11 DROP TABLE INFLUENCE;
12 CREATE TABLE IF NOT EXISTS INFLUENCE(
13   NO INT,
14   TITLE STRING,
15   YEAR INT,
16   INFLUENCE FLOAT,
17   GENRE STRING,
18   DIRECTORS STRING,
19   STARS STRING,
20   SCENARIO STRING,
21   GROSS STRING,
22   CERTIFICATE STRING
23 );
24
25 INSERT INTO INFLUENCE
26 SELECT INFO.NO, INFO.TITLE, INFO.YEAR, (INFO.RATING/Y.RATING + INFO.VOTES/Y.VOTES), INFO.GENRE, INFO.DIRECTORS, INFO.STARS
27 FROM imdb_movie_info AS INFO JOIN IMDB_YEAR Y ON INFO.year = Y.YEAR;
28
29 CREATE TABLE GENRE(
30   GENRE STRING,
31   INFLUENCE FLOAT
32 );
33
34 INSERT INTO GENRE
35 SELECT ITEM, INFLUENCE FROM INFLUENCE LATERAL VIEW EXPLODE(SPLIT(GENRE, ',')) T AS ITEM;
36
37 CREATE TABLE DIRECTORS(
38   DIRECTOR STRING,
39   INFLUENCE FLOAT
40 );
41
42 INSERT INTO DIRECTORS
43 SELECT ITEM, INFLUENCE FROM INFLUENCE LATERAL VIEW EXPLODE(SPLIT(DIRECTORS, ',')) T AS ITEM;
44
45 CREATE TABLE STARS(
46   STAR STRING,
47   INFLUENCE FLOAT
48 );
49
50 INSERT INTO STARS
51 SELECT ITEM, INFLUENCE FROM INFLUENCE LATERAL VIEW EXPLODE(SPLIT(STARS, ',')) T AS ITEM;
52
53 CREATE TABLE SCENARIO(
54   SCENARIO STRING,
```

```

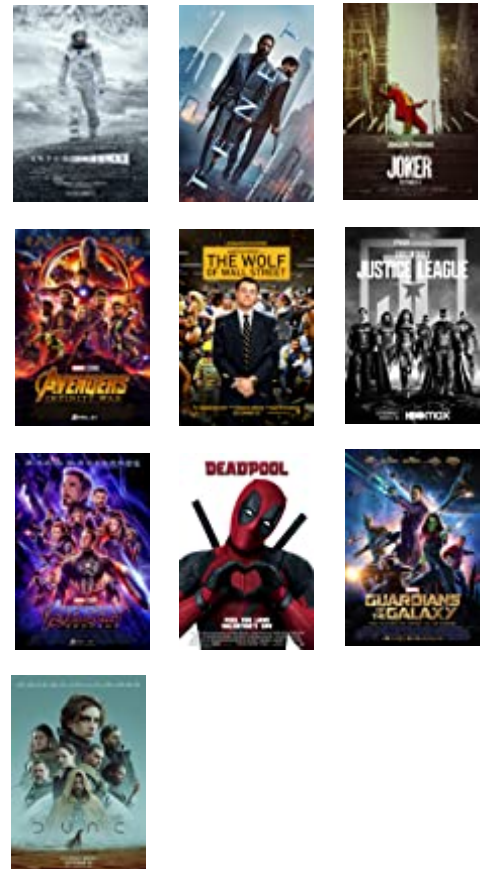
54 SCENARIO STRING,
55 INFLUENCE FLOAT
56 );
57
58 INSERT INTO SCENARIO
59 SELECT ITEM, INFLUENCE FROM INFLUENCE LATERAL VIEW EXPLODE(SPLIT(SCENARIO, ' ')) T AS ITEM;
60
61 CREATE TABLE CERTIFICATE(
62 CERTIFICATE STRING,
63 INFLUENCE FLOAT
64 );
65
66 INSERT INTO CERTIFICATE
67 SELECT ITEM, INFLUENCE FROM INFLUENCE LATERAL VIEW EXPLODE(SPLIT(CERTIFICATE, ' ')) T AS ITEM;
68
69 DROP TABLE IMDB_DIRECTOR;
70 CREATE TABLE IMDB_DIRECTOR(
71 DIRECTOR STRING,
72 INFLUENCE FLOAT
73 );
74 INSERT INTO IMDB_DIRECTOR
75 SELECT DIRECTOR, SUM(INFLUENCE) FROM DIRECTORS GROUP BY DIRECTOR;
76
77 DROP TABLE IMDB_GENRE;
78 CREATE TABLE IMDB_GENRE(
79 GENRE STRING,
80 INFLUENCE FLOAT
81 );
82 INSERT INTO IMDB_GENRE
83 SELECT GENRE, SUM(INFLUENCE) FROM GENRE GROUP BY GENRE;
84
85 DROP TABLE IMDB_STAR;
86 CREATE TABLE IMDB_STAR(
87 STAR STRING,
88 INFLUENCE FLOAT
89 );
90 INSERT INTO IMDB_STAR
91 SELECT STAR, SUM(INFLUENCE) FROM STARS GROUP BY STAR;
92
93 DROP TABLE IMDB_SCENARIO;
94 CREATE TABLE IMDB_SCENARIO(
95 SCENARIO STRING,
96 INFLUENCE FLOAT
97 );
98 INSERT INTO IMDB_SCENARIO
99 SELECT SCENARIO, SUM(INFLUENCE) FROM SCENARIO GROUP BY SCENARIO;
100
101 DROP TABLE IMDB_CERTIFICATE;
102 CREATE TABLE IMDB_CERTIFICATE(
103 CERTIFICATE STRING,
104 INFLUENCE FLOAT
105 );
106 INSERT INTO IMDB_CERTIFICATE
107 SELECT CERTIFICATE, SUM(INFLUENCE) FROM CERTIFICATE GROUP BY CERTIFICATE;

```

#### 4. 결론

##### # 영향력 높은 영화 TOP 10

influence.no	influence.title	influence.year	influence.influence
54	Interstellar	2014	0.0731208
80	Tenet	2020	0.067097194
66	Joker	2019	0.06385778
232	Avengers - Infinity War	2018	0.062093824
95	The Wolf of Wall Street	2013	0.058487933
107	Zack Snyder's Justice League	2021	0.057462297
84	Avengers - Endgame	2019	0.05697855
347	Deadpool	2016	0.055124767
198	Guardians of the Galaxy	2014	0.05248387
2	Dune	2021	0.051280785



: 10위 이내에 마블 영화 4개, DC 영화 2개가 있다는 것이 인상 깊었다. 그만큼 히어로 영화가 영향력이 큼을 알 수가 있다. 그 외에는 크리스토퍼 놀란 감독의 <인터스텔라>, <테넷>가 1,2위를 차지하였고 마틴 스코세이지 감독의 <더 울프 오브 월스트리트>가 5위를 차지하였다. 10위로는 드니 빌뇌브 감독의 <둔>이 차지하였다.

# 영향력 높은 장르 TOP 10

imdb_genre.genre	imdb_genre.influence
Action	6.3570766
Drama	5.0137315
Adventure	4.234676
Thriller	3.1131642
Drama	2.4914634
Comedy	2.3268783
Sci-Fi	2.2487767
Comedy	2.055341
Mystery	2.0354764
Crime	1.8958857

: 1위로 액션이 차지하였다. 그 뒤로는 드라마, 어드벤처, 스릴러, 드라마, 코미디가 차지하였다. TOP 10 영화들을 보면 액션이 1위를 차지하는 것이 쉽게 예상 갔으나 드라마는 예상 외였다.

# 영향력 높은 감독 TOP 10

imdb_director.director	imdb_director.influence
Joe Russo	0.21405827
Anthony Russo	0.21405827
Zack Snyder	0.18835652
Christopher Nolan	0.177849
Denis Villeneuve	0.17601946
James Gunn	0.13275772
James Wan	0.1104889
Antoine Fuqua	0.105452575
David Ayer	0.10527039
Martin Scorsese	0.09767947

: 많은 마블 영화들의 메가폰을 잡은 루소 형제(조 루소, 앤서니 루소)가 1,2위를 차지하였다. 그 뒤에는 잭 스나이더 감독(<300>, <잭 스나이더의 저스티스 리그> 등), 크리스토퍼 놀란 감독(<테넷>, <인터스텔라> 등)이 2,3위를 차지하였다. 유명한 감독들이 대부분 차지하였으나 데이비드 에이어 감독(David Ayer)은 의외였다. <퓨리> 외에는 좋은 작품이었다고 생각한 영화가 없었기 때문이다.

# 영향력 높은 배우 TOP 10

imdb_star.star	imdb_star.influence
Samuel L. Jackson	0.29825598
Mark Ruffalo	0.24388464
Woody Harrelson	0.22901167
Scarlett Johansson	0.21352038
Ryan Reynolds	0.20044094
Amy Adams	0.19264404
Dwayne Johnson	0.1797283
Robert Downey Jr.	0.17593813
Matthew McConaughey	0.17121471
Chris Pratt	0.1690331

: 히어로 영화의 힘이 대단하다는 걸 느꼈다. 사무엘 잭슨 같은 경우 <펄프픽션>부터 정말 다양한 영화들에 출연을 하였기에 수궁이 갔지만 2위로 마크 러팔로가 나오는게 진짜 의외였다. 또한 에이미 애담스 역시 슈퍼맨의 여자친구역을 제외하고는 자주 보지 못했는데도 6위를 차지하였다.

# 영향력 높은 시나리오 TOP 20

imdb_scenario.scenario	imdb_scenario.influence
the	21.044092
a	20.363142
to	14.4237585
of	12.966474
and	11.08172
in	7.4289627
his	6.519322
A	5.0675254
with	4.409254
is	4.1851025
an	4.141831
on	3.6695378
her	3.6163263
their	3.0731978
for	3.0084589
that	2.6495564
by	2.5200932
from	2.4296007
who	2.3162136
as	2.1255288
he	1.983676

: “A”나 “The” 같은 단어들이 문장에 많이 나오기에 어느정도 예상했지만 분석을 진행할 수 없을 정도로 전치사 등이 많이 나온 것을 알 수 있다.

# 영향력 높은 관람 연령 TOP 10

imdb_certificate.certificate	imdb_certificate.influence
R	7.121505
PG-13	5.832142
PG	1.3648998
TV-MA	0.75203794
Not	0.42738017
Rated	0.42738017
TV-14	0.15795733
'''	0.15066169
G	0.05959415
Unrated	0.049981773

: R 등급(청소년 관람 불가)가 1 위를 차지하였다. 그 다음으론 PG-13, PG 가 차지하였으나 1 위의 영향력 점수가 압도적이다. R 등급이기에 많은 시청자들을 확보하지 못하지만 그만큼 표현의 자유가 그만큼 영향력이 높았던 것 같다.

이를 통해 루소 형제가 제작한 R 등급의 액션 영화에 사무엘 잭슨이 출연하면 엄청난 파장을 일으킬 영화가 나올 수 있다는 결론을 얻었다. PG-13 등급의 영화는 많지만(마블 영화, 사무엘 잭슨 출연) 아직 R 등급의 영화는 만든 적 없는 것으로 알고 있는데 이 조합이 신선하진 않지만 R 등급이라면 재밌게 볼 것 같다.