# 영화추천
# 〈선호의 발견〉
# 프로젝트

파이썬, R을 활용한 빅데이터 시각화 과정 윤지환

# Gantt Chart를 이용한 일정관리

| 프로젝트 시작일 | 2021-08-20 |
|---|---|

| | 시작일 | 종료일 | Fri 20 | Sat 21 | Sun 22 | Mon 23 | Tue 24 | Wed 25 | Thu 26 | Fri 27 | Sat 28 | Sun 29 | Mon 30 | Tue 31 | Wed 01 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **준비 및 기획** | | | | | | | | | | | | | | | |
| 주제탐색, 분석 데이터확보 | Aug-20 | Aug-20 | | | | | | | | | | | | | |
| 문제정의, 분석계획수립, 관련자료수집 | Aug-20 | Aug-20 | | | | | | | | | | | | | |
| **서론** | | | | | | | | | | | | | | | |
| 도구점검, 필요역량확인 | Aug-23 | Aug-23 | | | | | | | | | | | | | |
| 개발목표 서비스 리스트 작성 | Aug-23 | Aug-24 | | | | | | | | | | | | | |
| 테마 및 서비스 항목 구성 | Aug-24 | Aug-24 | | | | | | | | | | | | | |
| **본론** | | | | | | | | | | | | | | | |
| 데이터 정제, 가공, 분류, 정렬 | Aug-24 | Aug-26 | | | | | | | | | | | | | |
| 단순시각화 | Aug-26 | Aug-26 | | | | | | | | | | | | | |
| 웹크롤링 & 고객선호분석 | Aug-27 | Aug-31 | | | | | | | | | | | | | |
| **검토** | | | | | | | | | | | | | | | |
| 서비스 추가구성, 데이터재구성, 추가 시각화 작업 | Sep-01 | Sep-01 | | | | | | | | | | | | | |
| 차후 개선과제 및 보완점 확인 | Sep-01 | Sep-01 | | | | | | | | | | | | | |
| | | | 4h | | | 4h | 4h | 4h | 4h | 4h | | | 4h | 4h | 4h |

total : 36h

# 개발환경 (resource)

| | |
|---|---|
| OS | Windows 10 Pro |
| Language | Python 3.8.5 |
| IDE | Anacomda jyputer notebook |
| Open Source | Pandas 1.2.4, seaborn 0.11.1, bs4 4.6.0, selenium, kkma, wordcloud |

# 순서

1. 도입
   - 프로젝트 목표설정, 자료구성 확인

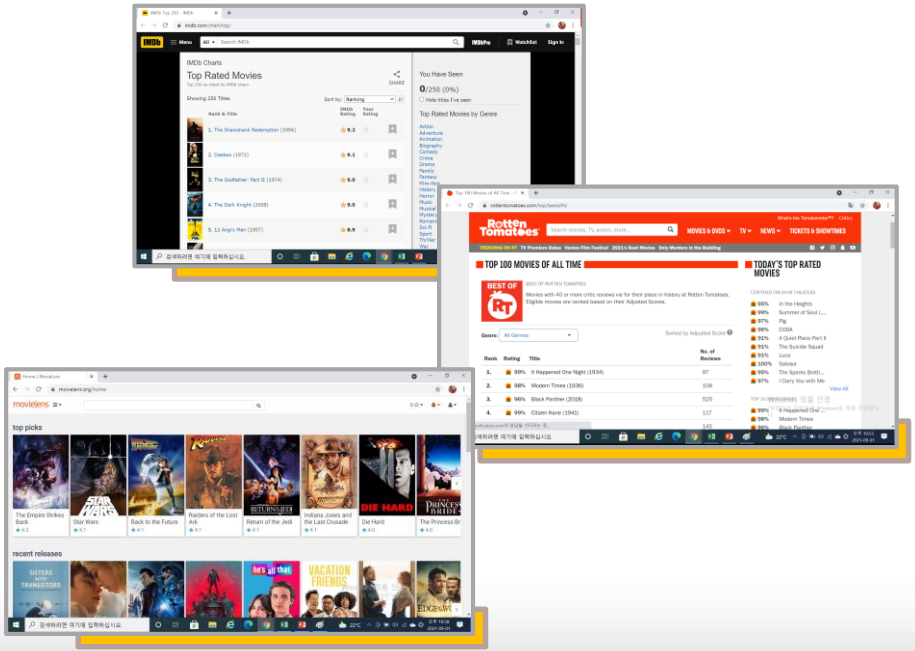
2. 전개
   - 어떤 서비스가 필요한가? ➡ 5가지로 구성

3. 분석 & 구현
   - 5가지 기능의 분석 과정과 구현 방법 및 결과 소개

4. 평가
   - 각 기능의 한계 확인 및 보완방안
   - 제공되는 서비스 외의 필요한 추가 기능 점검

1. 준비 및 기획
   - 주제탐색, 분석데이터확보
   - 문제정의, 계획수립, 관련자료수집
2. 서론
   - 분석도구점검, 필요역량확인
   - 타겟지표 리스트작성
   - 테마정의 및 서비스항목 구성
3. 본론
   - 데이터 정제·가공·분류·정렬
   - 단순시각화
   - 웹크롤링 & 고객선호분석
4. 검토
   - 타겟지표 추가구성, 데이터구성, 추가시각화작업
   - 한계 및 보완점 확인

# 1. 도입 – 프로젝트 목표

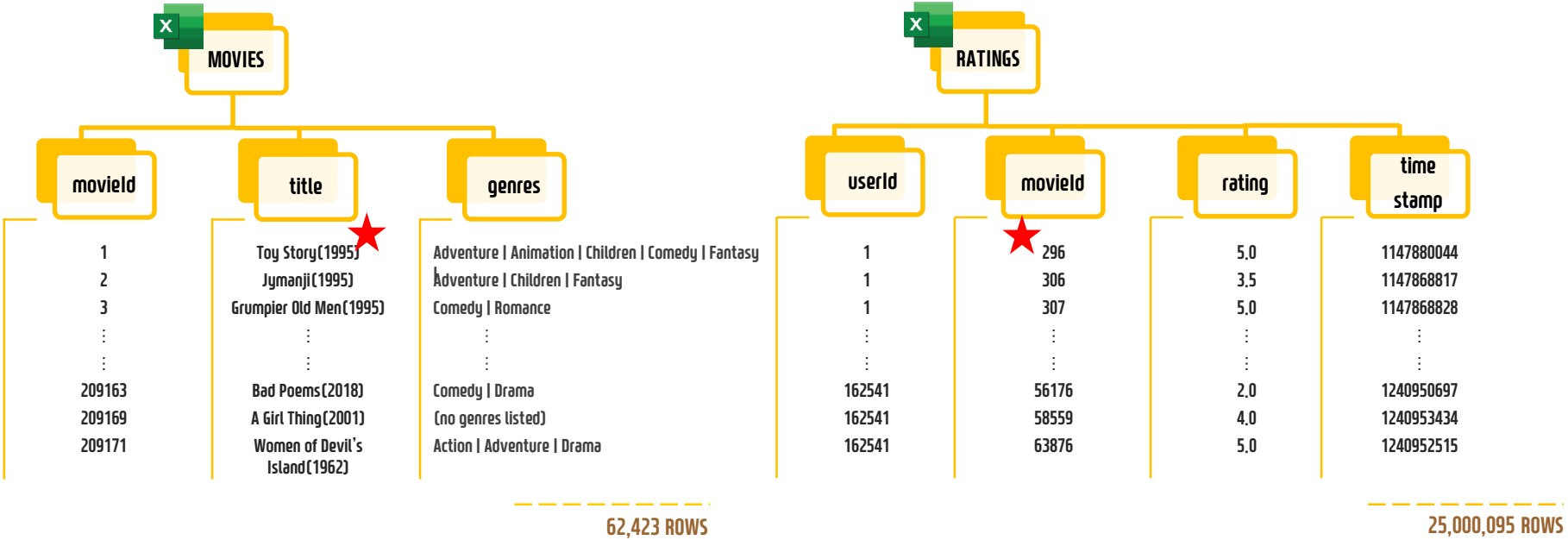



- 수십년간 축적된 데이터를 활용
- 시대별, 장르별, 관객평가 등 다양한 테마로 영화를 분류
- 영화를 분류를 통해 이용자가 자신이 원하는 영화목록을 바로 확인할 수 있도록 구성
- 프로젝트를 통해 구현된 서비스로 이용자가 (새로운 영화를 소개 / 영화 관련 정보를 제공) 받는 경험을 획득
- 원하는 영화에 대한 정보를 입력하면 연관된 영화가 조회 가능한 데이터를 준비할 수 있도록 개발

https://movielens.org/home 에서 수집한 데이터를 이용

Source : https://movielens.org/home

# 1. 도입 – 자료구성




TAGS

GENOME –TAGS

LINKS

GENOME –SCORES

6개의 csv파일로 구성

Movies와 ratings 파일을 주로 활용

# 2. 전개 – 어떤 서비스가 필요한가?

## 1. 자체 알고리즘에 의한 영화추천

- 평점 + 피평가횟수(=인기도) + 조정값 을 더한 순서
  - 평점/피평가횟수의 스케일 조정

## 2. 카테고리별 영화조회 서비스

- 연도별 검색
  - 희망 연도를 입력 -> 해당되는 데이터 반환
  - 최소 피평가횟수(25회)가 적용된 목록이 제공

## 3. 인기순 영화목록제공 서비스

- 관객 평가횟수(=인기도) 순서
  - 조정치를 구해봐야할듯

## 4. 평점순 영화목록 제공서비스

- 평점(rating) 순서
  - 최소 피평가횟수(25회) 적용

## 5. 실시간 트렌드 시각화(관객평)

- 네이버 평점·리뷰 페이지 웹크롤링을 통한 관객평 수집
- 웹크롤링 자료에 대해 자연어처리를 실시하고 일반명사(NNG)를 추출하여 상영 중인 영화들에 대한 관객들의 반응을 확인
- 수집·가공된 텍스트를 워드클라우드로 시각화

# 3. 분석& 구현 - (0)

## 공통


1. 데이터 읽기오기

(변수) = pd.read_csv('경로')

2. 결측치 확인 및 제거

df.dropna(axis=0, how='any', inplace = True)

3. 데이터프레임 그룹별 집계

-> 각 영화별 rating :

```
ratings_m= ratings.groupby('movieId')['rating'].mean()
movies_1 = pd.merge(movies, ratings_m, on=['movieId'])
```

4. 개봉년도

```
movies_1['released'] = movies_1['title'].str.extract(r'(₩d{4})')
                                    # 정규식을 활용한 개봉연도 추출
```

# 3. 분석& 구현 - (1)

Ratings를 movieId를 기준으로 그룹별 count를

## 기능 3. 인기순 영화목록제공 서비스

(기능 1~4에 대한 공통 데이터 정제과정이 이루어진 상태에서)

• **counts 필드의 값**(=이용자평가 횟수)**을 기준으로 정렬**

```
movies_2 = movies_1.sort_values('counts', ascending=False,
                                ignore_index=True)
Movies_2
```

MOVIES RATINGS

| | movieId | title | genres | rating | released | counts |
|---|---|---|---|---|---|---|
| 0 | 356 | Forrest Gump (1994) | Comedy\|Drama\|Romance\|War | 4.048011 | 1994 | 81491 |
| 1 | 318 | Shawshank Redemption, The (1994) | Crime\|Drama | 4.413576 | 1994 | 81482 |
| 2 | 296 | Pulp Fiction (1994) | Comedy\|Crime\|Drama\|Thriller | 4.188912 | 1994 | 79672 |
| 3 | 593 | Silence of the Lambs, The (1991) | Crime\|Horror\|Thriller | 4.151342 | 1991 | 74127 |
| 4 | 2571 | Matrix, The (1999) | Action\|Sci-Fi\|Thriller | 4.154099 | 1999 | 72674 |
| 5 | 260 | Star Wars: Episode IV - A New Hope (1977) | Action\|Adventure\|Sci-Fi | 4.120189 | 1977 | 68717 |
| 6 | 480 | Jurassic Park (1993) | Action\|Adventure\|Sci-Fi\|Thriller | 3.679175 | 1993 | 64144 |
| 7 | 527 | Schindler's List (1993) | Drama\|War | 4.247579 | 1993 | 60411 |
| 8 | 110 | Braveheart (1995) | Action\|Drama\|War | 4.002273 | 1995 | 59184 |
| 9 | 2959 | Fight Club (1999) | Action\|Crime\|Drama\|Thriller | 4.228311 | 1999 | 58773 |

1.추천영화 | 2. 연도별 추천영화 | 3. 인기영화 | 4. 고평점 영화 | 5. 실시간 트렌드(관객평) | 9. 종료

메뉴선택

# 3. 분석& 구현 - (2)

## 기능 4. 평점순 영화목록제공 서비스

MOVIES RATINGS

(기능 1~4에 대한 공통 데이터 정제과정이 이루어진 상태에서)

1. 일정 횟수 이상 평가가 이루어진 영화들만을 대상으로 함

(eg : 1회 별5개 평가시 1위에 랭크되는 것을 방지)

2. Rating 값을 기준으로 정렬

```
movies_3 = movies_1[movies_1['counts']>25]
movies_3 = movies_3.sort_values('rating', ascending=False,
                                ignore_index = True)
movies_3
```

| | movieId | title | genres | rating | released | counts |
|---|---|---|---|---|---|---|
| 0 | 171011 | Planet Earth II (2016) | Documentary | 4.483096 | 2016 | 1124 |
| 1 | 159817 | Planet Earth (2006) | Documentary | 4.464797 | 2006 | 1747 |
| 2 | 318 | Shawshank Redemption, The (1994) | Crime\|Drama | 4.413576 | 1994 | 81482 |
| 3 | 170705 | Band of Brothers (2001) | Action\|Drama\|War | 4.398599 | 2001 | 1356 |
| 4 | 171495 | Cosmos | (no genres listed) | 4.326715 | NaN | 277 |
| 5 | 858 | Godfather, The (1972) | Crime\|Drama | 4.324336 | 1972 | 52498 |
| 6 | 179135 | Blue Planet II (2017) | Documentary | 4.289833 | 2017 | 659 |
| 7 | 50 | Usual Suspects, The (1995) | rime\|Mystery\|Thriller | 4.284353 | 1995 | 55366 |
| 8 | 174551 | Obsession (1965) | Comedy | 4.277778 | 1965 | 36 |
| 9 | 198185 | Twin Peaks (1989) | Drama\|Mystery | 4.267361 | 1989 | 288 |

1.추천영화 | 2. 연도별 추천영화 | 3. 인기영화 | 4. 고평점 영화 | 5. 실시간 트렌드(관객평) | 9. 종료

메뉴선택

# 3. 분석& 구현 - (3)

## 기능 1. 자체 알고리즘에 의한 영화추천

MOVIES RATINGS

### 1. NaN값의 제거

released(개봉년도)가 결측치인 데이터가 존재

```
# title로부터 released값을 뽑아낼 때 정규표현식에 해당하는 값이 없던 행들의 경우 NaN값을 갖게 됨
movies_1[movies_1['released'].isnull()]
```

executed in 37ms, finished 17:00:28 2021-09-01

| | movieId | title | genres | rating | released |
|---|---|---|---|---|---|
| **15023** | 79607 | Millions Game, The (Das Millionenspiel) | Action\|Drama\|Sci-Fi\|Thriller | 3.400000 | NaN |
| **24817** | 123619 | Terrible Joe Moran | (no genres listed) | 2.500000 | NaN |
| **25555** | 125571 | The Court-Martial of Jackie Robinson | (no genres listed) | 3.500000 | NaN |
| **25580** | 125632 | In Our Garden | (no genres listed) | 4.000000 | NaN |
| **25659** | 125958 | Stephen Fry In America - New World | (no genres listed) | 3.394737 | NaN |
| **...** | ... | ... | ... | ... | ... |
| **58711** | 207714 | Tales of Found Footage | (no genres listed) | 3.000000 | NaN |
| **58743** | 207884 | Enduring Destiny | (no genres listed) | 5.000000 | NaN |
| **58911** | 208597 | Punk the Capital: Building a Sound Movement | Documentary | 5.000000 | NaN |
| **58951** | 208763 | Yosemite: The Fate of Heaven | (no genres listed) | 2.500000 | NaN |
| **59004** | 208973 | The Falklands War: The Untold Story | (no genres listed) | 3.500000 | NaN |

368 rows × 5 columns

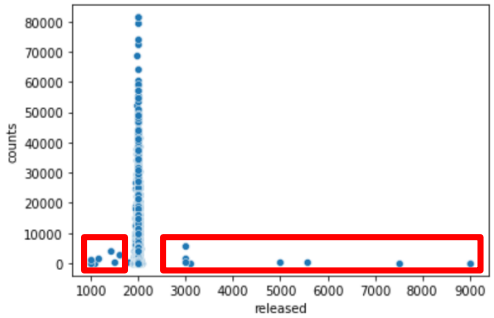
### 2. 유효한 연도값을 가진 데이터만 추출

1차 가공된 값에는 비정상적인 값이 포함되어 있음.

```
import seaborn as sns
sns.scatterplot(x='released', y='counts', data = movies_prac)
```

executed in 902ms, finished 17:30:53 2021-09-01

```
<AxesSubplot:xlabel='released', ylabel='counts'>
```



# 3. 분석& 구현 - (3)

## 기능 1. 자체 알고리즘에 의한 영화추천

MOVIES RATINGS

```
# 결측치 제거(NaN값 제거)
movies_4 = movies_1.dropna()

# 유효한 연도값만 추출
movies_4['released'] = movies_4['released'].astype('int64')
movies_4 = movies_4[(movies_4['released']>=1895) & (movies_4['released']<=2019)]

1. 추천 알고리즘(value값 계산식0 생성
movies_4['value'] = movies_4['rating']*4/5
                    + movies_4['counts']/movies_4['counts'].max()*4
                    + (movies_4['released']-movies_4['released'].min())
                      /(movies_4['released'].max()-movies_4['released'].min())*2

2. value값을 기준으로 정렬
movies_4 = movies_4.sort_values('value',ascending=False, ignore_index=True)
Movies_4
```

| | movieId | title | genres | rating | released | counts | value |
|---|---|---|---|---|---|---|---|
| 0 | 318 | Shawshank Redemption, The (1994) | Crime\|Drama | 4.413576 | 1994 | 81482 | 11.808185 |
| 1 | 296 | Pulp Fiction (1994) | Comedy\|Crime\|Drama\|Thriller | 4.188912 | 1994 | 79672 | 11.472466 |
| 2 | 356 | Forrest Gump (1994) | Comedy\|Drama\|Romance\|War | 4.048011 | 1994 | 81491 | 11.443173 |
| 3 | 2571 | Matrix, The (1999) | Action\|Sci-Fi\|Thriller | 4.154099 | 1999 | 72674 | 11.129248 |
| 4 | 593 | Silence of the Lambs, The (1991) | Crime\|Horror\|Thriller | 4.151342 | 1991 | 74127 | 11.022093 |
| 5 | 2959 | Fight Club (1999) | Action\|Crime\|Drama\|Thriller | 4.228311 | 1999 | 58773 | 10.350543 |
| 6 | 527 | Schindler's List (1993) | Drama\|War | 4.247579 | 1993 | 60411 | 10.325152 |
| 7 | 260 | Star Wars: Episode IV - A New Hope (1977) | Action\|Adventure\|Sci-Fi | 4.120189 | 1977 | 68717 | 10.320292 |
| 8 | 50 | Usual Suspects, The (1995) | Crime\|Mystery\|Thriller | 4.284353 | 1995 | 55366 | 10.100770 |
| 9 | 4993 | Lord of the Rings: The Fellowship of the Ring,... | Adventure\|Fantasy | 4.091189 | 2001 | 55736 | 10.075469 |

1.추천영화 | 2. 연도별 추천영화 | 3. 인기영화 | 4. 고평점 영화 | 5. 실시간 트렌드(관객평) | 9. 종료

메뉴선택

# 3. 분석& 구현 - (4)

## 기능 2. 카테고리별 영화조회 서비스

MOVIES RATINGS

# 1. 결측치 제거(NaN값 제거)
# 2. 유효한 연도값을 가진 데이터만 추출
# 3. 추천 알고리즘 (value값 계산식) 생성
# 4. value값을 기준으로 정렬
# 5. 사용자로부터 장르와 연도 입력받고 조건에 맞는 데이터 조회

```
Movies_5 = movies_5.loc[movies_5['genres'].str.contains(gen, case=False)
                        & (movies_5['released'] == year)]
Movies_5
```

| | movieId | title | genres | rating | counts | released | value |
|---|---|---|---|---|---|---|---|
| 0 | 55820 | No Country for Old Men (2007) | Crime\|Drama | 4.026145 | 18474 | 2007 | 5.934167 |
| 1 | 50872 | Ratatouille (2007) | Animation\|Children\|Drama | 3.811140 | 19157 | 2007 | 5.795688 |
| 2 | 56367 | Juno (2007) | Comedy\|Drama\|Romance | 3.732847 | 18306 | 2007 | 5.691283 |
| 3 | 54001 | Harry Potter and the Order of the Phoenix (2007) | Adventure\|Drama\|Fantasy\|IMAX | 3.768890 | 14346 | 2007 | 5.525740 |
| 4 | 56782 | There Will Be Blood (2007) | Drama\|Western | 3.982726 | 10044 | 2007 | 5.485644 |
| 5 | 55247 | Into the Wild (2007) | Action\|Adventure\|Drama | 3.875194 | 10328 | 2007 | 5.413558 |
| 6 | 55765 | American Gangster (2007) | Crime\|Drama\|Thriller | 3.793536 | 7379 | 2007 | 5.203480 |
| 7 | 51540 | Zodiac (2007) | Crime\|Drama\|Thriller | 3.744267 | 7936 | 2007 | 5.191405 |
| 8 | 55908 | Man from Earth, The (2007) | Drama\|Sci-Fi | 3.959690 | 3225 | 2007 | 5.132503 |
| 9 | 54997 | 3:10 to Yuma (2007) | Action\|Crime\|Drama\|Western | 3.731641 | 6659 | 2007 | 5.118623 |

# 3. 분석& 구현 - (5)


## 5. 실시간 트렌드 시각화(관객평)

### 1. 실시간 동적 웹크롤링

Chromedriver를 활용하여 '1~원하는 페이지'까지 마우스 클릭

```
for i in range(1,20):
    page_btn =
    dv.find_element_by_css_selector('div.paging a:nth-
                                    child(' + str(i) + ') > span')
    page_btn.click()
```

⋮

# 정규식을 활용한 페이지 이동

```
crude = re.findall(r'<span class="st_on".+?<a href="#"',
contents, re.DOTALL)
```

⋮

# 각 페이지에서 정규식을 활용한 관객평 수집

```
for c in crude :
    ready = re.search(r'<br>(.+)WnWtWtWtWnWtWt
                      WtWnWtWtWtWnWtWtWtWtWnWtWt
                      WtWtWnWtWtWtWtWnWtWtWtWt
                                    <a href="#"', c)
    ready_to_print += ready.group(1)
```

⋮

### 2. 자연어 처리

```
word_list = kkma.pos('%r' % target)
word_list = [t[0] for t in word_list if t[1]=='NNG']
text = ' '.join(word_list)
```

⋮

### 3. 워드클라우드(from wordcloud import WordCloud)

```
불용어 = set(['영화', '연기', '배우'])
wc = WordCloud(background_color = 'white',
    max_words=300,
    font_path = 'C:/Windows/Fonts/한컴산뜻돋움/HanSantteutDotumBold.ttf',
    relative_scaling = 0.5,
    stopwords = 불용어,
    mask=mask)
```

⋮

# 3. 분석& 구현

## 5. 실시간 트렌드 시각화(관객평)

# 4. 평가

## 1. 자체 알고리즘에 의한 영화추천

- 평점(4) + 관객평가횟수(=인기도)(4) + 조정값(2)에 의해 산출된 최종값임에도 결국 관객평가횟수라는 단일 요소에 큰 영향을 받아 기능 3번과 차별화된 영화목록을 제공하지 못하는 한계를 지님
  -> 보다 다양한 요소들이 고려된 지표를 개발, 알고리즘을 강화함으로써 많은 이용자들이 숨겨진 자신의 선호를 발견할 수 있는 서비스가 될 수  있도록 개선

## 2. 카테고리별 영화조회 서비스

- 자체 제공 영화순위가 연도별 + 장르별 두 가지 요소만을 기준으로 조회가 가능함
  -> 연도 범위, 수상여부, 감독, 출연배우 등 이용자가 각기 다른 정보를 가지고 원하는 영화를 검색할 수 있도록 자유도가 향상된 기능으로 개선

## 3. 수집데 데이터의 입체적 활용

- Tags, Genome_Tags 혹은 ratings의 timestamp 등과 같이 이미 수집된 데이터를 적극 활용할 경우 이용자들이 시청한 영화간 연관분석, 영화평(tags)을 통한 시청 후의 느낀점 및 아이디어 을 추출할 수 있다 그에 따라 보다 입체적 분석이 가능해지고 풍성한 서비스를 구성할 수 있을 것으로 관측됨.

# THANK


조선
조선
스토리
한국
최근
차
승원
정민
황
황정민
사람
마블
상견
스크린
최고
영상미
타령
중국
고무
마
자체
평점
알바
기대
안
액션
신

# YOU