

영화 데이터를 이용한 Hybrid 추천 시스템 구현

뭐 볼지 알려주는 아이들 (3조)

황인범, 정한솔, 이경현, 이세원

서비스 산업 데이터를 활용한 빅데이터 분석 실무 - 멀티캠퍼스

임정섭 강사님

목차

1	프로젝트 배경	프로젝트의 주제, 개요, 구조, 목적, 기대효과	p. 3
2	팀 구성	프로젝트 팀 구성 및 개개인의 주도적 역할	p. 7
3	프로젝트 절차	프로젝트 기획, 아이디어를 도출 및 활동 내용	p. 8
4	프로젝트 결과	프로젝트 결과물 도출 과정	p. 9
5	느낀 점	경험한 성찰이나 반성, 성과, 자신의 경력 계획	p. 33

1. 프로젝트 배경_주제 및 목적

• 주제 및 목적

- Kaggle에서 제공되는 사용자 데이터와 영화 메타 데이터를 기반으로 다양한 알고리즘과 변수들을 적용해 Hybrid 추천 시스템을 구현하고, 그 시스템의 결과를 시각화하여 비교, 분석을 통해 사용자에게 양질의 추천 시스템을 제공한다.
- 이를 통해 사용자의 만족감을 높이고 궁극적으로 해당 영상 플랫폼의 충성 고객층을 형성하는데 그 목적이 있다.



1. 프로젝트 배경_개요

• 프로젝트 개요

훈련 내용과의 연관성

- 해당 프로젝트는 분석 도구 Python을 기반으로 진행
- 내부 패키지 Pandas 및 Numpy를 사용하여 교육 과정에서 학습했던 분석 도구 R의 함수, 통계적 개념을 적극 활용
- 기존 학습했던 데이터의 타입, 포맷 변환을 자유자재로 활용
- 시각화 과정에서 이전 학습했던 bar plot, ggplot등의 개념을 기반으로 Power BI에 적용 및 응용하여 분석 결과 및 차이점을 더욱 명확하고 효과적으로 전달 할 수 있는 시각화를 구현

개발 환경

- 좀 더 유연하고 원활한 환경에서 빅데이터를 다루기 위해 교육과정에서 제공되는 AWS 서버 및 Jupyter Notebook 활용
- 간편한 접근, 코드 작성 및 수정을 위해 Google에서 제공되는 Colaboratory를 활용
- 강사님 및 매니저님의 적극적인 지원 아래 Zoom 영상 회의를 통해 원활한 의사 소통, Screen Share 기능을 바탕으로 서로의 코드를 공유하며 팀워크 효율성 극대화

1. 프로젝트 배경_구조



1

사전 조사 및 일정 정리

주제 선정, 문제 정의 및 분석 후
목표 수립

이후 각자 리서치를 통하여 추천
시스템 종류, 특징 등에 대한 **전반
적인 이해도**를 높임

브레인스토밍을 통해 추후 프로젝
트에 사용될 **프로그램을 선정**하고
대략적인 전체 **일정 기획 및 조율**



2

사용자 기반 시스템 구현

사용자 평점 기반 시스템 코드를
기반으로 하여 추가/삭제/변형
의 수정 단계를 거쳐 확보하고 있
는 영화 데이터에 맞게 적용

수정한 코드 (제목 input 추천)에
개인화 추천 알고리즘을 더하여

제목 추천 + 개인화 추천

두가지 방법으로 상황에 맞게 영
화를 추천 받을 수 있도록 구현



3

Hybrid 시스템 구현

각자 맡은 변수
(‘평점’, ‘출시일’, ‘인기도’, ‘언어’)를
기존 알고리즘에 적용시켜 **Hybrid
시스템을 구현**하고, 이후 개개인이
만든 코드들을 통합하여 하나의 완성
된 Hybrid 추천 시스템 창작

완성된 코드 내에서 각자 **예측 알고
리즘** (SVD, Slope-one, NMF, ALS)
을 맡아 기존 알고리즘과 비교할 수
있는 코드를 작성하고 이후 예측 알
고리즘들의 **정확도를 비교 분석**



4

시각화 구현

비교 분석한 결과를 다양한 **그래프
및 차트**로 보여주는 최종 과정으로,
보고서 작성을 맡은 이세원 제외 각
개인이 시각화 주제를 전담하여
Power BI에서 동적 그래프로 구현

여러 예측 알고리즘에 따른 추천 결
과의 변화와 변수의 영향에 따른 추
천의 결과의 변화를 효과적으로 시각
화하고 전달하며 이를 통해 추천시스
템의 이해도를 높인다

1. 프로젝트 배경_기대효과

다양한 변수들 적용
여러 예측 알고리즘 비교 분석
개인화 + 제목 추천
효과적인 시각화
다양한 결과

새로운 알고리즘

기대 효과

하나의 변수만 적용
하나의 예측 알고리즘 적용
제목으로만 추천
시각화 X
하나의 결과
비효과적인 전달

기존 알고리즘

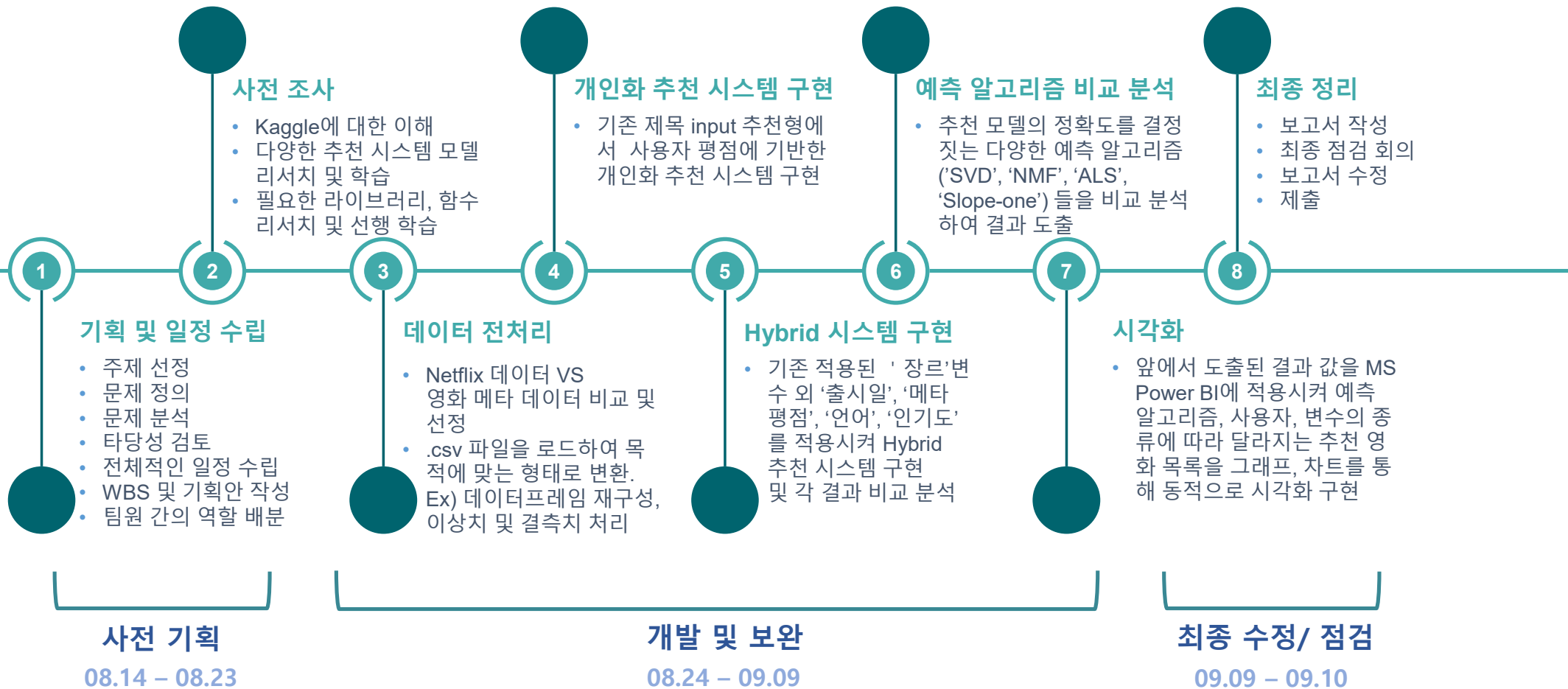


2. 팀 구성

훈련생 명	역할
황인범 (팀 리더)	<ul style="list-style-type: none"> ▪ 사용자 평점 및 콘텐츠 기반 Hybrid 추천 시스템 선행 학습 및 제목 추천, 개인화 추천 알고리즘 구현 ▪ Hybrid 추천 시스템에 사용된 변수 중 "release_date (출시일)" 을 맡아 적용 ▪ 예측 알고리즘 중 "SVD"을 담당하여 학습 및 분석, 적용 ▪ "유저별 영화 추천 결과"를 Power BI의 그래프 및 차트를 이용해 효과적으로 시각화 ▪ 보고서 작성
정한솔 (팀원)	<ul style="list-style-type: none"> ▪ 사용자 평점 및 콘텐츠 기반 Hybrid 추천 시스템 선행 학습 및 제목 추천, 개인화 추천 알고리즘 구현 ▪ Hybrid 추천 시스템에 사용된 변수 중 "language(언어)" 를 맡아 적용 ▪ 예측 알고리즘 중 "NMF"를 담당하여 학습 및 분석, 적용 ▪ "유저가 본 영화 데이터의 변화에 따라 달라지는 영화 추천 알고리즘"을 Power BI로 시각화 진행 ▪ 보고서 작성
이경현 (팀원)	<ul style="list-style-type: none"> ▪ 사용자 평점 및 콘텐츠 기반 Hybrid 추천 시스템 선행 학습 및 제목 추천, 개인화 추천 알고리즘 구현 ▪ Hybrid 추천 시스템에 사용된 변수 중 "rating(평점)" 을 맡아 적용 ▪ 예측 알고리즘 중 "ALS"를 담당하여 학습 및 분석, 적용 ▪ "변수 가중치, 알고리즘에 따른 영화 추천 결과"를 Power BI로 시각화 진행 ▪ 보고서 작성
이세원 (팀원)	<ul style="list-style-type: none"> ▪ 사용자 평점 및 콘텐츠 기반 Hybrid 추천 시스템 선행 학습 및 제목 추천, 개인화 추천 알고리즘 구현 ▪ Hybrid 추천 시스템에 사용된 변수 중 "popularity(인기도)"를 맡아 적용 ▪ 예측 알고리즘 중 "Slope-one"을 담당하여 학습 및 분석, 적용 ▪ "알고리즘별 영화 추천 결과"를 Power BI의 그래프 및 차트를 이용해 효과적으로 시각화 ▪ 보고서 작성

3. 프로젝트 절차

총 개발기간: 08.14 - 09.10 (4주)

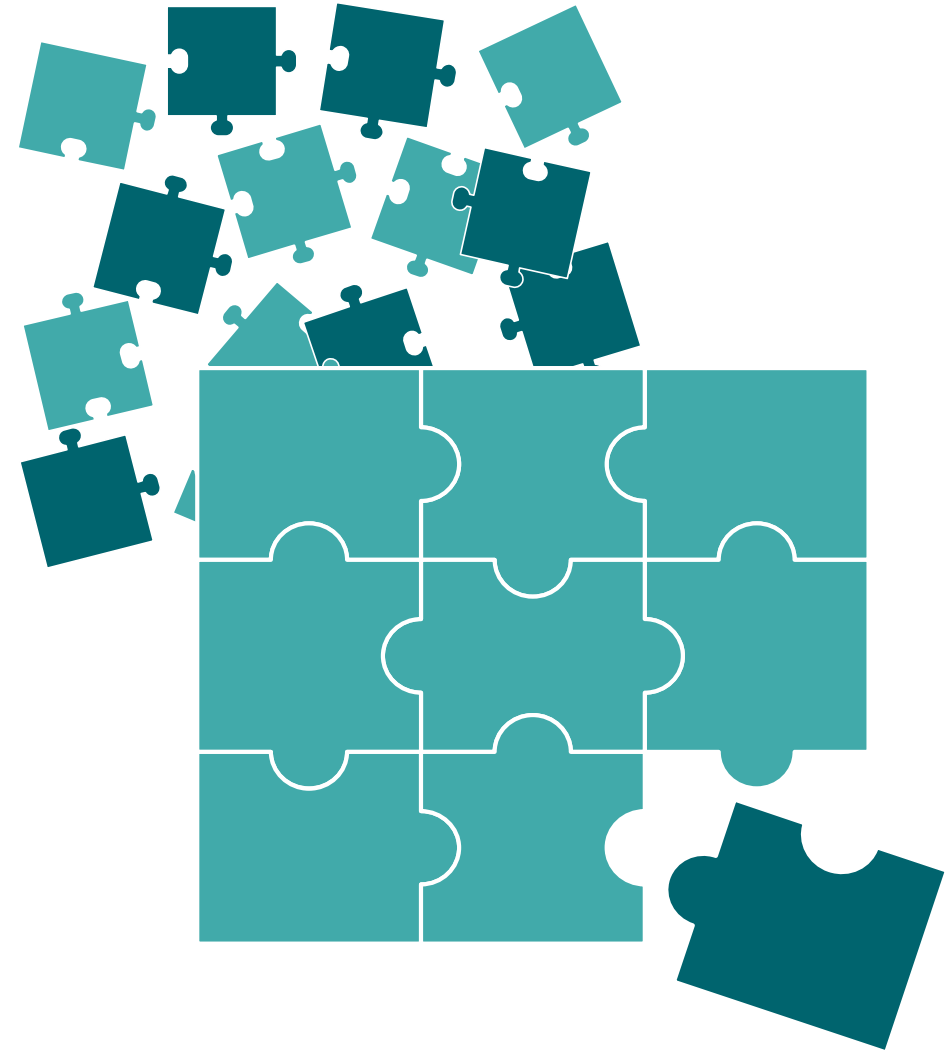


프로젝트 수행 결과

4. 프로젝트 수행 결과_기획 및 사전조사

정보 과잉의 시대

- 정보 기술의 발달로 정보 생산성이 급성장하고 있으며 이러한 현상 속에서 소비자는 자신이 원하는 정보를 찾는데 종종 어려움을 겪고 정보의 정확도 또한 떨어진다.
- SNS 사용자 10명 중 3명이 SNS 피로증후군을 경험했다고 답했고, 사생활이 불특정 다수에게 노출되는 것이 싫다(34.1%)거나 흥미와 관심이 떨어졌다(43.9%)는 이들도 많았다. (한국일보, 2017)¹⁾
- 기업의 입장에선 정보 과잉의 시대에 얼마만큼 정확하고 알맞는 정보를 취할 수 있는지가 관건 (조선비즈, 2013)²⁾



1) 한국일보, 2017, <http://m.koreatimes.com/article/20170831/1073950>

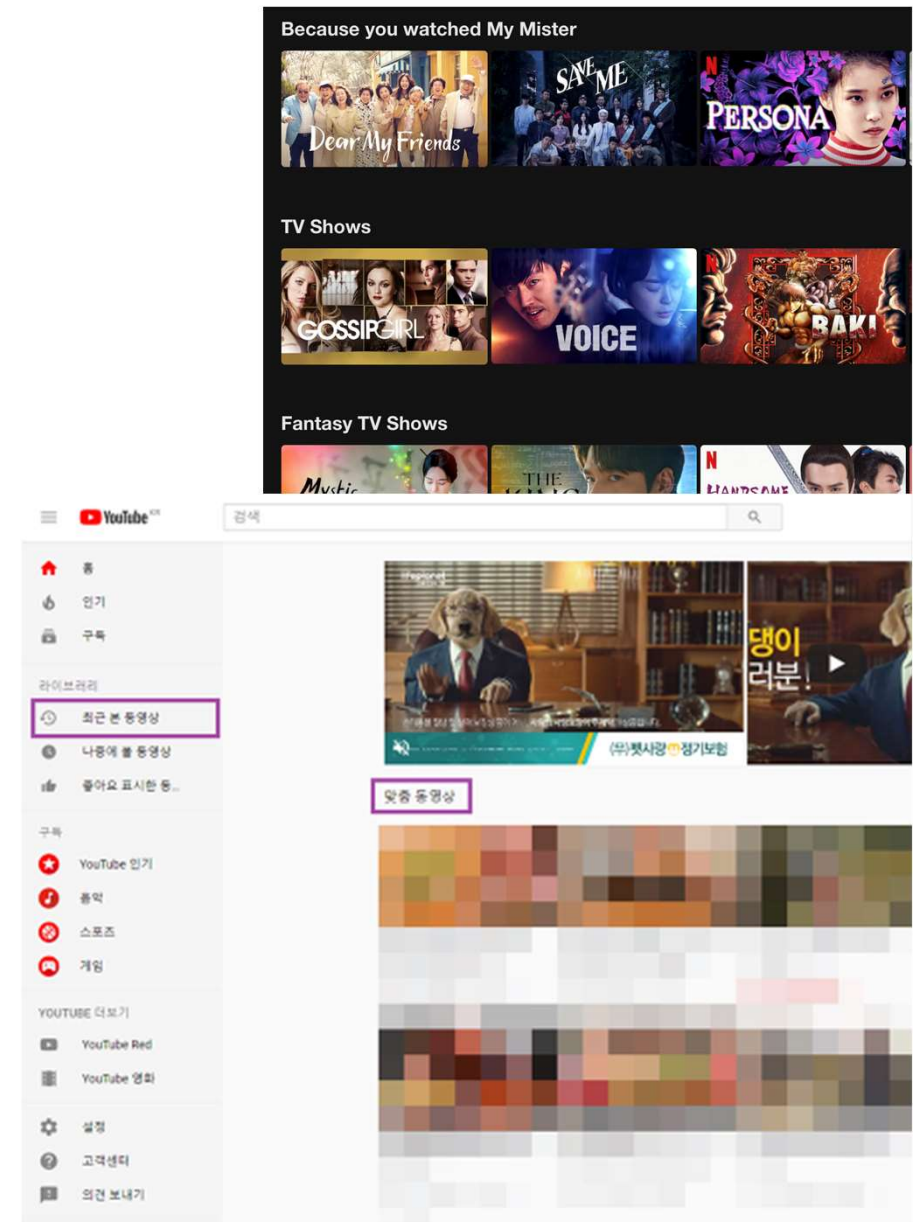
2) 조선비즈, 2013, https://biz.chosun.com/site/data/html_dir/2013/12/06/2013120601254.html

4. 프로젝트 수행 결과_ 기획 및 사전조사

추천 시스템이란?

- 추천 시스템이란 사용자의 취향이나 선호의 기반으로 정보를 선별하여 사용자에게 적합한 특정 항목을 선택(information filtering)하여 제공하는 시스템을 일컫는다. (정인용, 양새동 & 정회경, 2015)¹⁾
- 추천 시스템에는 협업 필터링, 콘텐츠 기반 필터링, 그리고 Hybrid 필터링이 있다.

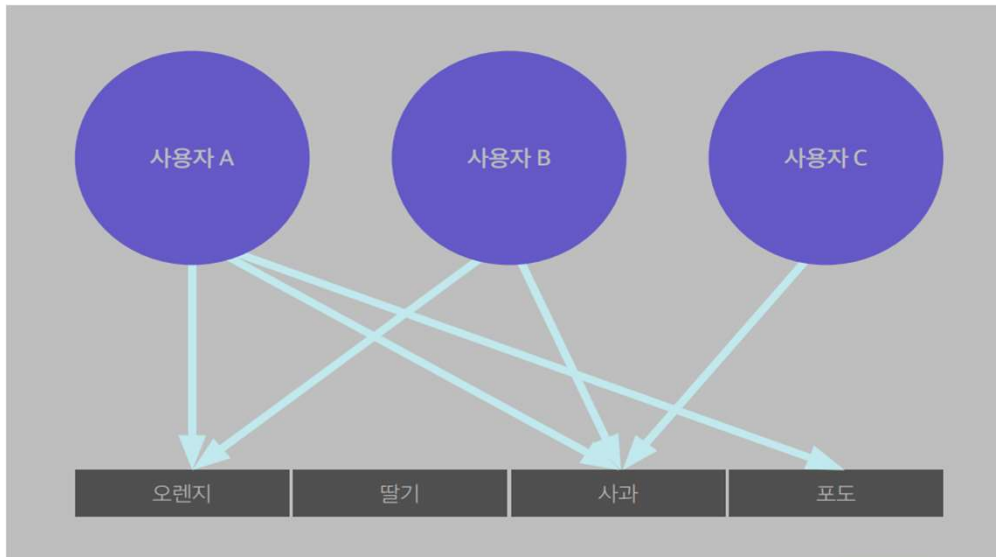
1) 정인용, 양새동, 정회경, "A Study on Movies Recommendation System of Hybrid Filtering-Based", *Journal of the Korea Institute of Information and Communication Engineering* Vol. 19, No. 1, January 2015, pp. 113-118



4. 프로젝트 수행 결과_ 기획 및 사전조사

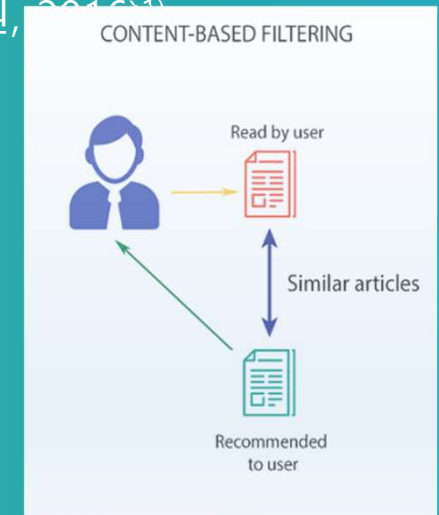
협업 필터링

- 대규모의 기존 사용자 행동 정보를 분석하여 해당 사용자와 비슷한 성향의 사용자들이 기존에 좋아했던 항목을 추천하는 기술이다.
- 예를 들어, 사용자 A와 사용자 B가 사과와 오렌지를 같이 구매했으므로, 사과를 구매한 사용자 C에게 오렌지를 추천한다.



콘텐츠 기반 필터링

- 협업 필터링이 사용자의 행동 기록을 이용하는 반면, 콘텐츠 기반 필터링은 항목 자체를 분석하여 추천을 구현한다. 예를 들어 음악을 추천하기 위해 음악 자체를 분석하여 유사한 음악을 추천하는 방식이다. (서봉원, 2016:1)



1) 서봉원, "콘텐츠 추천 알고리즘의 진화", *Broadcasting Trend & Insight* Vol. 5, No. 1, April 2016, pp. 19-24

4. 프로젝트 수행 결과_ 기획 및 사전조사

협업 필터링 모델의 단점

1. 콜드 스타트 (Cold Start)

다중 사용자들이 기존 아이템에 대해 평가한 선호도를 기반으로 다음 아이템을 추천하는데, 만약 새로운 아이템이 추가 되었을 때 이 아이템이 사용자들로부터 평가를 받기까지 즉, 다음 추천을 위한 충분한 데이터가 쌓일 때 까지 오랜 시간이 필요하다.

2. 롱테일 (Long Tail)

선택권이 많다 해도 사용자들은 소수 인기 항목들에 편향되는 단점이 있다. 그래서 데이터는 점점 더 양극화 되고 결국 비교적 좋은 아이템이 있어도 추천을 할 수 있는 기반 데이터가 적어 문제를 야기한다.

콘텐츠 기반 필터링 모델의 단점

3. 다양한 형식 추천 어려움

모델 특성상 콘텐츠들을 분석하고 특징들을 카테고리화 시켜 데이터로 저장하고 이를 추천에 사용한다. 하지만 다양한 형식의 아이템이 추가되면 상이한 형식의 아이템들의 특징을 통일시키기 어렵다.

예를 들어, 비디오와 사진은 다른 아이템 형식으로 서로 다른 특징들을 가지고 있고 얻을 수 있는 정보가 달라 프로파일 구성이 어렵다.

(서봉원, 2016)¹⁾

4. 낮은 정확도

협업 필터링과 비교 했을 때 더 낮은 정확도를 보임



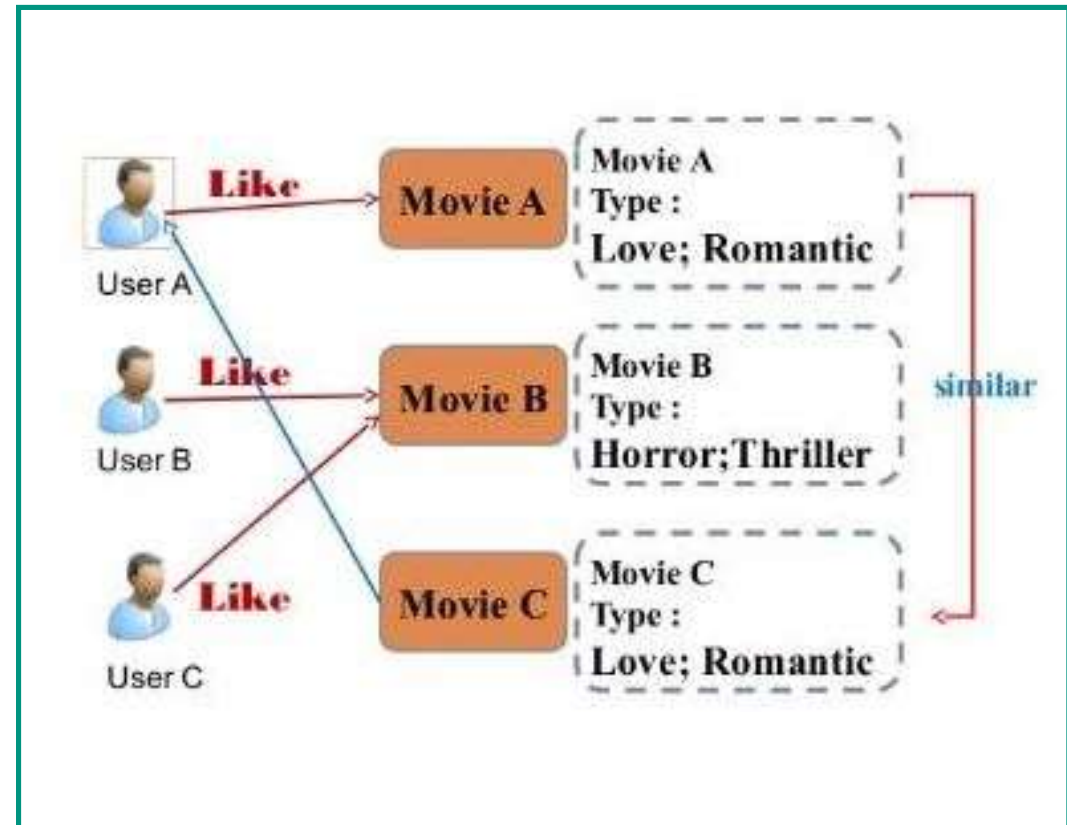
1) 서봉원, "콘텐츠 추천 알고리즘의 진화", *Broadcasting Trend & Insight* Vol. 5, No. 1, April 2016, pp. 19-24

4. 프로젝트 수행 결과_ 기획 및 사전조사

대안

하이브리드 (Hybrid) 필터링 모델

- 위 두 모델의 단점들을 최소화 하기 위해서 고안된 방법이 바로 Hybrid 필터링이다.
- 사용자 평점 기반 데이터에 콘텐츠 기반 데이터 (ex. 특징들)의 가중치를 주고 두개를 합산하여 좀 더 정확한 결과를 낼 수 있는 모델이다.
- 콜드 스타트를 방지하기 위해 신규 아이템에 대해서는 콘텐츠 기반 가중치가 보완 역할을 하고, 반대로 콘텐츠 기반 필터링의 단점은 협업 필터링 모델로 상쇄하는 상호보완적인 알고리즘 모델이라고 할 수 있다.

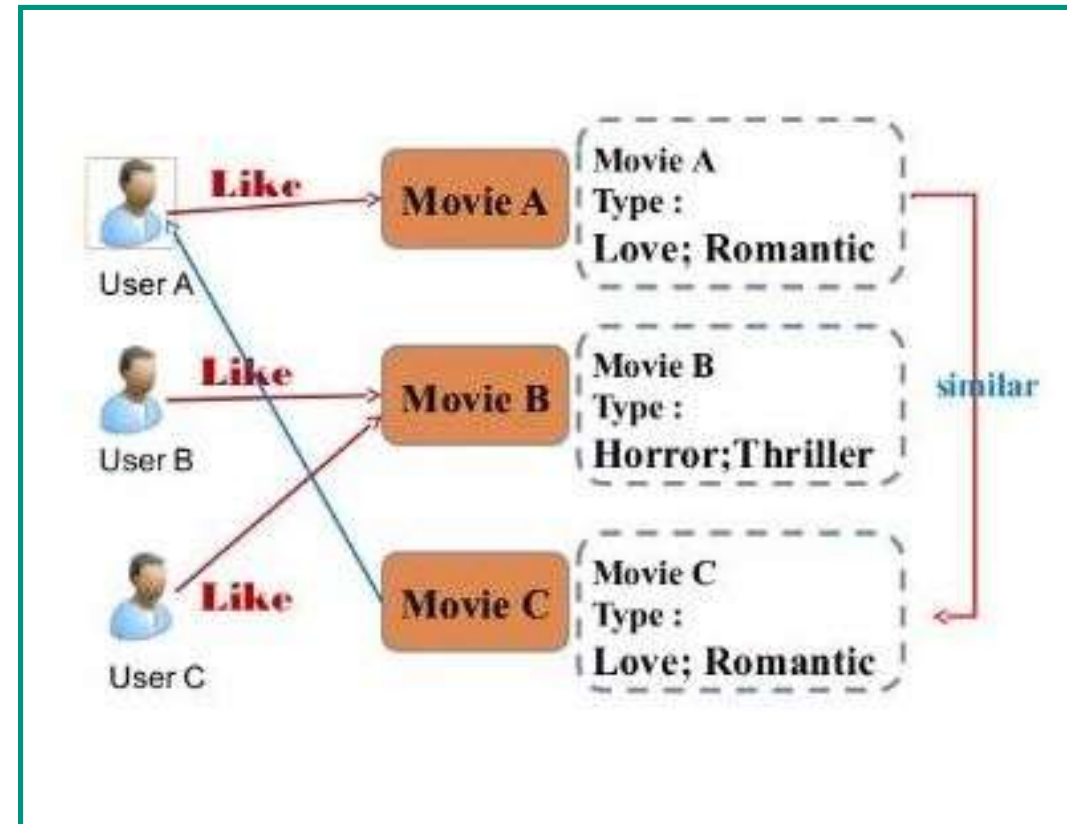


4. 프로젝트 수행 결과_ 기획 및 사전조사

목표

향상된 성능의 Hybrid 추천 시스템 구현

- 이러한 Hybrid 모델을 더욱 정교화 시키기 위해, 사용자 평점 데이터에 영화 메타 데이터의 다양한 변수들의 가중치를 적용시키고, 다양한 예측 알고리즘 또한 비교 분석하여 Hybrid 추천 시스템의 성능을 극대화 시키는 것을 목표로 함
- 궁극적으로, 앞서 제기했던 정보 과잉의 문제를 해결하기 위해 개개인에게 적합한 양질의 정보를 제공하고, 기업의 관점에서 소비자를 좀 더 정확히 이해하고 맞춤 상품을 추천함으로써 인해 수익의 극대화



4. 프로젝트 수행 결과_ 데이터 전처리

Netflix Prize 데이터 VS 영화 메타 데이터

데이터셋에 따른 방향 설정

- 초기에 Netflix Prize 데이터를 사용할 것으로 계획 했었지만 이 데이터셋은 사용자 평점 기반이 중심이 되고 하이브리드에 사용 할 수 있는 콘텐츠 기반 요소가 '장르' 하나뿐
- 따라서 해당 데이터셋을 가지고 우리가 애초에 계획했던 향상된 Hybrid 시스템을 구현할 수 있을지 의문이었고, 따라서 8월 24일 회의를 통해 장단점을 논의한 후 더 많은 콘텐츠 기반 요소가 있는 영화 메타 데이터셋으로 변경

8월 24일 월요일 회의록

1. Netflix Prize Data

내용 사용자의 Rating 데이터 기반 추천 시스템

장점 모델링 및 구성이 이미 완료되어 있음, 개인 맞춤 추천 시스템

단점 데이터셋에 콘텐츠 데이터가 없어 Hybrid System 적용 어려움

2. The Movies Dataset

내용 사용자 Rating 및 콘텐츠의 변수들 (Casting, Director, Origin, etc) 기반 추천시스템

장점 다양한 양질의 콘텐츠 변수들이 제공됨, 따라서 높은 정확도 기대

단점 개인 맞춤 추천 시스템 구현

결론: 2번을 기반으로 Genre 데이터를 통해 하이브리드 시스템 구현.
여유가 된다면 기타 다른 변수들 더 붙여 정확도 Up

일정

4. 프로젝트 수행 결과_ 데이터 전처리

데이터 타입 변환 및 재구성

- csv 파일을 불러와서 오른쪽과 같이 구조를 파악한 후 원하는 컬럼의 값들만 가져와 재구성
- movieId 컬럼 이름 명명 및 기타 값들의 타입을 적합하게 변경
- Genre 컬럼은 기존 정리가 되어있지 않은 모습이었는데 Parsing을 통해 재정리

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	popularity
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...}	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Family'}]	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	21.94
1	False		NaN	65000000		8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	17.01
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Family'}]		15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	11.7
3	False		NaN	16000000		31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	3.85
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Family'}]		11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	8.38

```
[ ] meta = meta[['id', 'original_title', 'genres', 'release_date', 'popularity', 'original_language', 'poster_path']]
meta = meta.rename(columns={'id': 'movieId'})
meta.movieId = pd.to_numeric(meta.movieId, errors='coerce')
meta.popularity = pd.to_numeric(meta.popularity, errors='coerce') # popularity를 문자열에서 숫자형으로 변환!

# meta.head()
```

```
def parse_genres(genres_str):
    genres = json.loads(genres_str.replace('\\', ''))

    genres_list = []
    for g in genres:
        genres_list.append(g['name'])

    return genres_list
```

```
meta['genres'] = meta['genres'].apply(parse_genres)
```

4. 프로젝트 수행 결과_개인화 추천 모델 구현

개인화 추천 구현

- 기존 코드 방식은 최종 결과물에 '제목'을 넣어야 새로운 영화들을 추천해주는 형식이었는데 팀은 User ID가 Input으로 들어갔을 때 영화를 추천해주는 방식이 더욱 유의미하다고 판단하여 개인화 추천 시스템을 구현
- User ID와 Rating에 따라 영화 추천 결과값이 다르게 도출

```
print(user_df665)
print(user_df664)
```

	original_title	...	Estimate_Score
5295	5 Card Stud	...	4.816690
33913	Der Tunnel	...	4.464185
11922	License to Wed	...	4.436658
6910	Birdman of Alcatraz	...	4.434383
938	The 39 Steps	...	4.395770
...
3293	L'Ours	...	2.205307
19332	舊縁の葬列	...	2.168298
11993	Abraham	...	2.127045
11324	Dr. Cyclops	...	2.053258
1391	Kolja	...	1.998790

[43759 rows x 6 columns]

	original_title	...	Estimate_Score
4020	The Million Dollar Hotel	...	4.530457
700	Dead Man	...	4.511589
6141	The Good Thief	...	4.504925
11566	Lonely Hearts	...	4.485913
6910	Birdman of Alcatraz	...	4.482592
...
81	Antonia	...	2.557509
3293	L'Ours	...	2.542781
11815	28 Weeks Later	...	2.488051
11993	Abraham	...	2.317700
11324	Dr. Cyclops	...	2.224762

[43759 rows x 6 columns]

유저에 따른 개인 영화 추천

```
def user_difference(data, usernumber, rating, moviedata, dropdata, reader, svd):
    df = data
    df_user = df[(df['userId'] == usernumber) & (df['rating'] == rating)]
    df_user = df_user.set_index('movieId')
    df_user = df_user.join(moviedata[['original_title']])
    print(df_user)

    user_release_ratio_list = user_release_ratio(df, usernumber) # 유저의 년도 비율을 가져온다.

    user_df = moviedata.copy()
    user_df = user_df[~user_df['movieId'].isin(dropdata)]
    data1 = Dataset.load_from_df(df[['userId', 'movieId', 'rating']], reader)
    trainset = data1.build_full_trainset()
    svd.fit(trainset)
    user_df['Estimate_Score'] = user_df['movieId'].apply(lambda x: svd.predict(usernumber, x).est)
    user_df = user_df.drop('movieId', axis=1)
    user_df = user_df.sort_values('Estimate_Score', ascending=False)
    print(user_df.head(10))

    return user_df

user_df665 = user_difference(df, 665, 5, meta, drop_movie_list, reader, svd)
user_df664 = user_difference(df, 664, 5, meta, drop_movie_list, reader, svd)
```

4. 프로젝트 수행 결과_Hybrid 추천 모델 구현

Hybrid 추천 모델 구현

- 사용자 기반 필터링과 영화 메타 데이터의 변수들의 가중치를 더하여 좀 더 정확하고 유의미한 결과값을 도출

예시)

- 오른쪽 예시와 같이, 함수를 만들어 그 안에서 Loop 구문을 돌려 정의한 범주 내로 조건을 만족하는 값들을 넣어줌
- 마지막에 "특정 범주값 / 총합" 하여 특정 범주값의 전체 대비 비율을 구하고 그 비율이 가중치가 됨

```
[ ] def user_release_ratio(df, usernumber):
    user_df = df[df['userId'] == usernumber]
    meta2 = pd.read_csv('../content/drive/My Drive/data/the-movies-dataset/movies_metadata.csv', low_memory=False)
    value_meta = meta2[['id', 'original_title', 'release_date', 'genres']]

    value_meta = value_meta.rename(columns={'id': 'movieId'})
    value_meta.movieId = pd.to_numeric(value_meta.movieId, errors='coerce')
    value_meta = value_meta.dropna(axis=0)
    value_meta = value_meta.reset_index()
    merge_data = pd.merge(user_df, value_meta, on='movieId', how='left')
    merge_data = merge_data.dropna(axis=0)
    merge_data = merge_data.reset_index()

    release_date_list = {'1900':0, '1950':0, '1960':0, '1970':0, '1980':0, '1990':0, '2000':0, '2010':0, '2020':0}
    for i in range(0, len(merge_data)):
        if int(merge_data['release_date'].loc[i][0:4]) <= 1900:
            release_date_list["1900"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 1950:
            release_date_list["1950"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 1960:
            release_date_list["1960"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 1970:
            release_date_list["1970"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 1980:
            release_date_list["1980"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 1990:
            release_date_list["1990"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 2000:
            release_date_list["2000"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 2010:
            release_date_list["2010"] += 1
        elif int(merge_data['release_date'].loc[i][0:4]) <= 2020:
            release_date_list["2020"] += 1

    release_date_list

    sum = 0
    for i in release_date_list:
        sum += release_date_list[i]

    release_date_rate = []
    for i in release_date_list:
        if release_date_list[i] == 0:
            continue
        release_date_list[i] = round((release_date_list[i]/sum), 3)
    return release_date_list
```

4. 프로젝트 수행 결과_Hybrid 추천 모델 구현

Hybrid 추천 모델 구현

예시)

- 이렇게 생성된 가중치를 각 범주에 할당하는 작업을 거침
- 예시와 같은 작업을 각 변수마다 거치게 됨

```
[ ] def Estimate_Score_sum1(user_df, user_release_ratio_list):
    user_df = user_df.dropna(axis=0)
    for i in range(0, len(user_df)):
        if int(user_df.iloc[i]['release_date'][0:4]) <= 1900:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['1900']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 1950:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['1950']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 1960:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['1960']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 1970:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['1970']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 1980:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['1980']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 1990:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['1990']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 2000:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['2000']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 2010:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['2010']
        elif int(user_df.iloc[i]['release_date'][0:4]) <= 2020:
            user_df['Estimate_Score'].loc[user_df.index[i]] = user_df.iloc[i]['Estimate_Score'] + user_release_ratio_list['2020']
    return user_df
```

4. 프로젝트 수행 결과_Hybrid 추천 모델 구현

Hybrid 추천 모델 구현

- 각 변수의 가중치를 더한 결과값들을 따로 도출하여 비교 분석

변수 별 영화 추천

```
def variable_weight(data, usernumber, rating, moviedata, dropdata, reader, algo):
    df = data
    df_user = df[(df['userId'] == usernumber) & (df['rating'] == rating)]
    df_user = df_user.set_index('movieId')
    df_user = df_user.join(moviedata)[['original_title']]
    # print(df_user)

    user_release_ratio_list = user_release_ratio(df, usernumber) # 유저의 년도 비율을 가져온다.
    # print(user_release_ratio_list)
    user_pop_ratio_list = user_pop_ratio(df, usernumber) # 유저의 popularity 비율을 가져온다.
    # print(user_pop_ratio_list)
    user_language_ratio_list = user_language_ratio(df, usernumber) # 유저의 language 비율을 가져온다.
    # print(user_language_ratio_list)

    user_df = moviedata.copy()
    user_df = user_df[~user_df['movieId'].isin(dropdata)]
    datal = Dataset.load_from_df([['userId', 'movieId', 'rating']], reader)
    trainset = datal.build_full_trainset()
    algo.fit(trainset)
    user_df['Estimate_Score'] = user_df['movieId'].apply(lambda x: algo.predict(usernumber, x).est)
    user_df = user_df.drop('movieId', axis=1)
    user_df = user_df.sort_values('Estimate_Score', ascending=False)
    # print(user_df.head(10))

    user_df_sum = Estimate_Score_sum1(user_df, user_release_ratio_list)
    user_df_sum_release = user_df_sum.sort_values('Estimate_Score', ascending=False)
    print("개봉일 별 가중치")
    print(user_df_sum_release.head(10))

    user_df_sum = Estimate_Score_sum2(user_df, user_pop_ratio_list)
    user_df_sum_pop = user_df_sum.sort_values('Estimate_Score', ascending=False)
    print("인기 별 가중치")
    print(user_df_sum_pop.head(10))

    user_df_sum = Estimate_Score_sum3(user_df, user_language_ratio_list)
    user_df_sum_lang = user_df_sum.sort_values('Estimate_Score', ascending=False)
    print("언어 별 가중치")
    print(user_df_sum_lang.head(10))

    return user_df_sum_release, user_df_sum_pop, user_df_sum_lang
user_df_sum_release, user_df_sum_pop, user_df_sum_lang = variable_weight(df, 665, 5, meta, drop_movie_list, reader, svd)
```

```
[ ] print(user_df_sum_release.head(10))
print(user_df_sum_pop.head(10))
print(user_df_sum_lang.head(10))
print(user_df_sum_vote.head(10))
```

```
10089      original_title ... Estimate_Score
24036      Madagascar    ...      4.809054
Der rote Elvis ...      4.742484
5253      Enough        ...      4.685655
11922      License to Wed ...      4.658529
11355      Flags of Our Fathers ...      4.652800
22347      Miffo         ...      4.636811
40113      Local Color   ...      4.632677
4020      The Million Dollar Hotel ...      4.621149
11566      Lonely Hearts ...      4.583051
12216      The Golden Compass ...      4.531655

[10 rows x 6 columns]

      original_title ... Estimate_Score
4020  The Million Dollar Hotel ...      4.610149
6836  Broken Blossoms    ...      4.589656
24036  Der rote Elvis     ...      4.575484
2649  The Thomas Crown Affair ...      4.560433
5253  Enough             ...      4.553655
10089  Madagascar        ...      4.540054
534    Sleepless in Seattle ...      4.498078
11922  License to Wed     ...      4.496529
11355  Flags of Our Fathers ...      4.490800
286    Once Were Warriors ...      4.489941

[10 rows x 6 columns]

      original_title ... Estimate_Score
10089  Madagascar    ...      5.289054
4020  The Million Dollar Hotel ...      5.222149
2649  The Thomas Crown Affair ...      5.202433
6836  Broken Blossoms ...      5.201656
5253  Enough         ...      5.165655
11922  License to Wed ...      5.138529
11355  Flags of Our Fathers ...      5.132800
3058  Galaxy Quest   ...      5.127713
534    Sleepless in Seattle ...      5.120078
40113  Local Color    ...      5.112677

[10 rows x 6 columns]

      original_title ... Estimate_Score
259    Once Were Warriors ...      4.735607
4808    5 Card Stud       ...      4.665998
476    Sleepless in Seattle ...      4.644566
35822  Confession of a Child of the Century ...      4.523984
1043    To Kill a Mockingbird ...      4.498238
2363  The Thomas Crown Affair ...      4.477621
1173    Batman Returns    ...      4.452596
10925  License to Wed     ...      4.448770
2361  The Sixth Sense     ...      4.411268
10507  The Good Shepherd ...      4.395469

[10 rows x 6 columns]
```


4. 프로젝트 수행 결과_예측 알고리즘

예측 알고리즘 비교 분석

- SVD를 포함하여 NMF, ALS, Slope-one 예측 알고리즘의 성능 및 정확도를 각각 비교 분석

예시)

- 오른쪽 예시와 같이, 직접 함수를 만들거나 함수식을 교차 검증 (Cross Validation)에 적용하고 결과 비교

```
[ ] def ALS(df_p):
    R = np.array(df_p) # ALS하기 위해 배열이 필요해서 배열로 바꿔준다.
    R = np.nan_to_num(R) #nan_to_num : nan 값을 0으로 바꾸기 위한 함수
    r_lambda = 40 # 블로그에서 언급하길 논문에서 가장 좋은 결과를 냈다는 것을 가져다 쓴다.
    nf = 200
    alpha = 40

    R = np.array(df_p) # ALS하기 위해 배열이 필요해서 배열로 바꿔준다.
    R = np.nan_to_num(R) #nan_to_num : nan 값을 0으로 바꾸기 위한 함수

    nu = R.shape[0] # nu는 userId의 개수
    ni = R.shape[1] # ni는 movieId의 개수

    # initialize X and Y with very small values
    X = np.random.rand(nu, nf) * 0.01 # np.random.rand : 0 - 1의 균일분포 표준정규분포 난수를 matrix array(m(행),n(열))로 반환
    Y = np.random.rand(ni, nf) * 0.01

    P = np.copy(R)
    P[P > 0] = 1 #위의 사용자 평점들을 1로 바꿔줘야 하기 때문에 0보다 크면 1로 다 바꿔준다.
    C = 1 + alpha * R

    predict_errors = []
    confidence_errors = []
    regularization_list = []
    total_losses = []
    for i in range(2):
        if i != 0:
            yT = np.transpose(Y) # np.transpose 행과 열을 바꾸는 함수
            for u in range(nu):
                Cu = np.diag(C[u]) # np.diag 대각행렬 만드는 함수
                yT_Cu_y = np.matmul(np.matmul(yT, Cu), Y) # np.matmul 두 배열의 행렬곱
                liy = np.dot(r_lambda, np.identity(nf)) # np.dot 두 배열의 내적곱, np.identity 2차원의 정방단위행렬 객체를 반환
                yT_Cu_pu = np.matmul(np.matmul(yT, Cu), P[u])
                X[u] = np.linalg.solve(yT_Cu_y + liy, yT_Cu_pu) # np.linalg.solve 연립방정식 해 풀기.

            xT = np.transpose(X)
            for i in range(ni):
                Ci = np.diag(C[i, :])
                xT_Ci_x = np.matmul(np.matmul(xT, Ci), X)
                lix = np.dot(r_lambda, np.identity(nf))
                xT_Ci_pi = np.matmul(np.matmul(xT, Ci), P[i, :])
                Y[i] = np.linalg.solve(xT_Ci_x + lix, xT_Ci_pi)

            predict = np.matmul(X, np.transpose(Y))
            predict_error = np.square(P - predict)
            confidence_error = np.sum(C * predict_error)
            regularization = r_lambda * (np.sum(np.square(X)) + np.sum(np.square(Y)))
            total_loss = confidence_error + regularization
            # predict_error, confidence_error, regularization, total_loss = loss_function(C, P, predict, X, Y, r_lambda)

            predict_errors.append(predict_error)
            confidence_errors.append(confidence_error)
            regularization_list.append(regularization)
            total_losses.append(total_loss)
            # predict_error, confidence_error, regularization, total_loss = loss_function(C, P, predict, X, Y, r_lambda)

            predict_errors.append(predict_error)
            confidence_errors.append(confidence_error)
            regularization_list.append(regularization)
            total_losses.append(total_loss)

    predict = np.matmul(X, np.transpose(Y))
    print('final predict')
    print([predict])

    return predict

[ ] reader = Reader()
```

4. 프로젝트 수행 결과_ 예측 알고리즘

예측 알고리즘 비교 분석

- SVD를 포함하여 NMF, ALS, Slope-one 예측 알고리즘의 성능 및 정확도를 각각 비교 분석

예시)

- 오른쪽 예시와 같이, 직접 함수를 만들거나 함수 식을 교차 검증 (Cross Validation)에 적용하고 결과 비교

알고리즘 별 cross_validate

```
[ ] data = Dataset.load_from_df(df[['userId', 'movieId', 'rating']], reader)
    svd = SVD()
    slope = SlopeOne()
    nmf = NMF()

    bsl_options = {'method': 'als',
                  'n_epochs': 5,
                  'reg_u': 12,
                  'reg_i': 5
                  }
    als = BaselineOnly(bsl_options=bsl_options)

    als_result = cross_validate(als, data, measures=['RMSE', 'MAE'], cv=5, verbose=False) #evaluate 대신 cross_validate
    slope_result = cross_validate(slope, data, measures=['RMSE', 'MAE'], cv=5, verbose=False) #evaluate 대신 cross_validate
    svd_result = cross_validate(svd, data, cv = 5, measures=['RMSE', 'MAE']) #evaluate 대신 cross_validate사용
    nmf_result = cross_validate(nmf, data, measures=['RMSE', 'MAE'], cv=5, verbose=False) #evaluate 대신 cross_validate
```

알고리즘에 따른 RMSE, MAE 비교

```
[ ] print(slope_result)
    print(svd_result)
    print(nmf_result)
    print(als_result)
```

```
{'test_rmse': array([0.87971907, 0.87101856, 0.88009223, 0.87136654, 0.88427013]), 'test_mae': array([0.67634766, 0.67026
{'test_rmse': array([0.87459235, 0.87518767, 0.85694145, 0.86820855, 0.86525299]), 'test_mae': array([0.67414103, 0.67293
{'test_rmse': array([0.90292033, 0.88657769, 0.904553 , 0.89707099, 0.88480504]), 'test_mae': array([0.69390337, 0.68080
{'test_rmse': array([0.84550772, 0.86423789, 0.86138027, 0.85378715, 0.86219043]), 'test_mae': array([0.65171743, 0.66573
```

4. 프로젝트 수행 결과_ 예측 알고리즘

예측 알고리즘 구현

- 결과적으로 UserRec3에 사용자 ID와 Rating을 넣으면 각기 다른 알고리즘들로 다양한 결과를 추천받고 비교 해볼 수 있음
- 이러한 결과들을 Power BI에서 시각화로 전달

```
def userRec3(data, usernumber, rating, moviedata, dropdata, reader, algo):  
    df = data  
    df_user = df[(df['userId'] == usernumber) & (df['rating'] == rating)]  
    df_user = df_user.set_index('movieId')  
    df_user = df_user.join(moviedata['original_title'])  
    # print(df_user)  
  
    user_release_ratio_list = user_release_ratio(df, usernumber) # 유저의 년도 비율을 가져온다.  
    # print(user_release_ratio_list)  
    user_pop_ratio_list = user_pop_ratio(df, usernumber) # 유저의 popularity 비율을 가져온다.  
    # print(user_pop_ratio_list)  
    user_language_ratio_list = user_language_ratio(df, usernumber) # 유저의 language 비율을 가져온다.  
    # print(user_language_ratio_list)  
  
    user_df = moviedata.copy()  
    user_df = user_df[~user_df['movieId'].isin(dropdata)]  
    datal = Dataset.load_from_df(df[['userId', 'movieId', 'rating']], reader)  
    trainset = datal.build_full_trainset()  
    algo.fit(trainset)  
    user_df['Estimate_Score'] = user_df['movieId'].apply(lambda x: algo.predict(usernumber, x).est)  
    # user_df = user_df.drop('movieId', axis = 1)  
    user_df = user_df.sort_values('Estimate_Score', ascending=False)  
    # print(user_df.head(10))  
  
    # user_df_sum = Estimate_Score_sum1(user_df, user_release_ratio_list)  
    # # user_df_sum = user_df_sum.sort_values('Estimate_Score', ascending=False)  
    # # print(user_df_sum.head(10))  
  
    # user_df_sum = Estimate_Score_sum2(user_df_sum, user_pop_ratio_list)  
    # # user_df_sum = user_df_sum.sort_values('Estimate_Score', ascending=False)  
    # # print(user_df_sum.head(10))  
  
    return user_df  
  
als_recomend = userRec3(df, 665, 5, meta, drop_movie_list, reader, als)  
svd_recomend = userRec3(df, 665, 5, meta, drop_movie_list, reader, svd)  
slope_recomend = userRec3(df, 665, 5, meta, drop_movie_list, reader, slope)  
nmf_recomend = userRec3(df, 665, 5, meta, drop_movie_list, reader, nmf)  
  
print(als_recomend)  
print(svd_recomend)  
print(slope_recomend)  
print(nmf_recomend)
```


4. 프로젝트 수행 결과_ 영화 포스터

TMDB Open API를 통해 영화 추천 포스터 추가

- 시각화 전 단계에서 TMDB에서 제공하고 있는 API를 기반으로 영화에 맞는 포스터를 가져옴
- 이후 이루어질 시각화 단계에서 좀 더 효과적이고 명확하게 결과를 전달 할 수 있고, 일반적인 결과 시각화와 비교했을 때 사용자들의 흥미를 불러일으킬 수 있음

```
def display_recommendations(data):
    api_key = '7api_key=8a98ba13f2855e4a5c4aae1af3e81974'
    url = 'https://api.themoviedb.org/3/movie/'
    urls = []
    start = 0
    data['imgUrl'] = ''
    for i in range(len(data)):
        mov_num = str(data['movieId'].iloc[i])
        JSONcontent = requests.get(url + mov_num + api_key)
        response = JSONcontent.json()
        base_url = 'https://image.tmdb.org/t/p/original'
        img_url = response['poster_path']
        if img_url != None:
            image_url = base_url + img_url
            urls.append(image_url)
            data['imgUrl'].iloc[i] = urls[i]
            print(data['imgUrl'].iloc[i])
        else:
            image_url = "None"
            urls.append(image_url)
            data['imgUrl'].iloc[i] = 'None'
            print(data['imgUrl'].iloc[i])

    images = ''
    for link in urls:
        if link != '':
            images += "<img style='width: 150px; margin: 1px; float: left; border: 1px solid black;' src='%s'" % link
    display(HTML(images))
    return data


user_df665 = user_df665.head(12) #위에서 받은 데이터 정리.
user_df665_poster = display_recommendations(user_df665)
user_df665_poster2 = user_df665_poster.to_csv('../content/drive/My Drive/user_df665_poster.csv')
```

/usr/local/lib/python3.6/dist-packages/pandas/core/indexing.py:671: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

self._setitem_with_indexer(indexer, value)

<https://image.tmdb.org/t/p/original/cwlesW1KorPiJavR6G2Obv0uy9r.jpg>
<https://image.tmdb.org/t/p/original/2ksp2aL8x5Axj1F1tyE2U0CkR.jpg>
<https://image.tmdb.org/t/p/original/2A3x0YbWdndknp0Ktaho1s2Iy.jpg>
<https://image.tmdb.org/t/p/original/1nd48sytYc26hy92g8NNU0p3maf.jpg>
<https://image.tmdb.org/t/p/original/w0y2mNCiiHdyo5Klguq6S28Ftn.jpg>
<https://image.tmdb.org/t/p/original/h6aVbUaiJB3tLxrhvZxX2013h.jpg>
<https://image.tmdb.org/t/p/original/B4Cw8Yein3D8Aau0EidFvYaXln.jpg>
<https://image.tmdb.org/t/p/original/5yqs1MVlqIdY5adC5jF33d7i.jpg>
<https://image.tmdb.org/t/p/original/iD2oCrOUckjcU8WtDH2g70iU4fA.jpg>
None
<https://image.tmdb.org/t/p/original/fjD0Bm7DiEzRR0ph0Tiace8oyNK.jpg>
<https://image.tmdb.org/t/p/original/sqe1RROEdYxud970x68AC3jBMV.jpg>



4. 프로젝트 수행 결과_ 시각화

Power BI를 이용한 시각화

- 위 작성했던 코드들을 바탕으로 Power BI 사용해 효과적으로 시각화를 진행

Power BI 선정 이유

1. 다양한 데이터를 업로드하거나 실시간으로 연동할 수 있는 다양한 종류의 데이터 커넥터

- 데이터 종류는 크게 파일, 서버, 데이터베이스, 온라인 서버 데이터로 구분되며, 기타 웹사이트와 빅데이터 연동도 지원
- 커넥터의 종류와 사이즈에 상관없이 데이터를 가져오는 데에 비용이 발생하지 않는 점이 다른 솔루션 대비 파워 BI의 가장 큰 장점 (주 경쟁사인 태블로(Tableau)의 경우 라이선스 구매에 따라 활용할 수 있는 데이터 종류 상이)

2. 빠르고 쉬운 시각화 구현

- 다양한 시각화 프리셋을 갖추고 있고 이 셋들을 사용하여 빠르고 깔끔하게 시각화 가능

3. 고급 인사이트

- 크게 수학적 분석과 시각적 분석 두가지의 고급 인사이트 제공. 또한, 시각적 분석 관련하여서 최근에 마이크로소프트가 Power BI에 인공지능(AI)을 접목하는 시도 중,

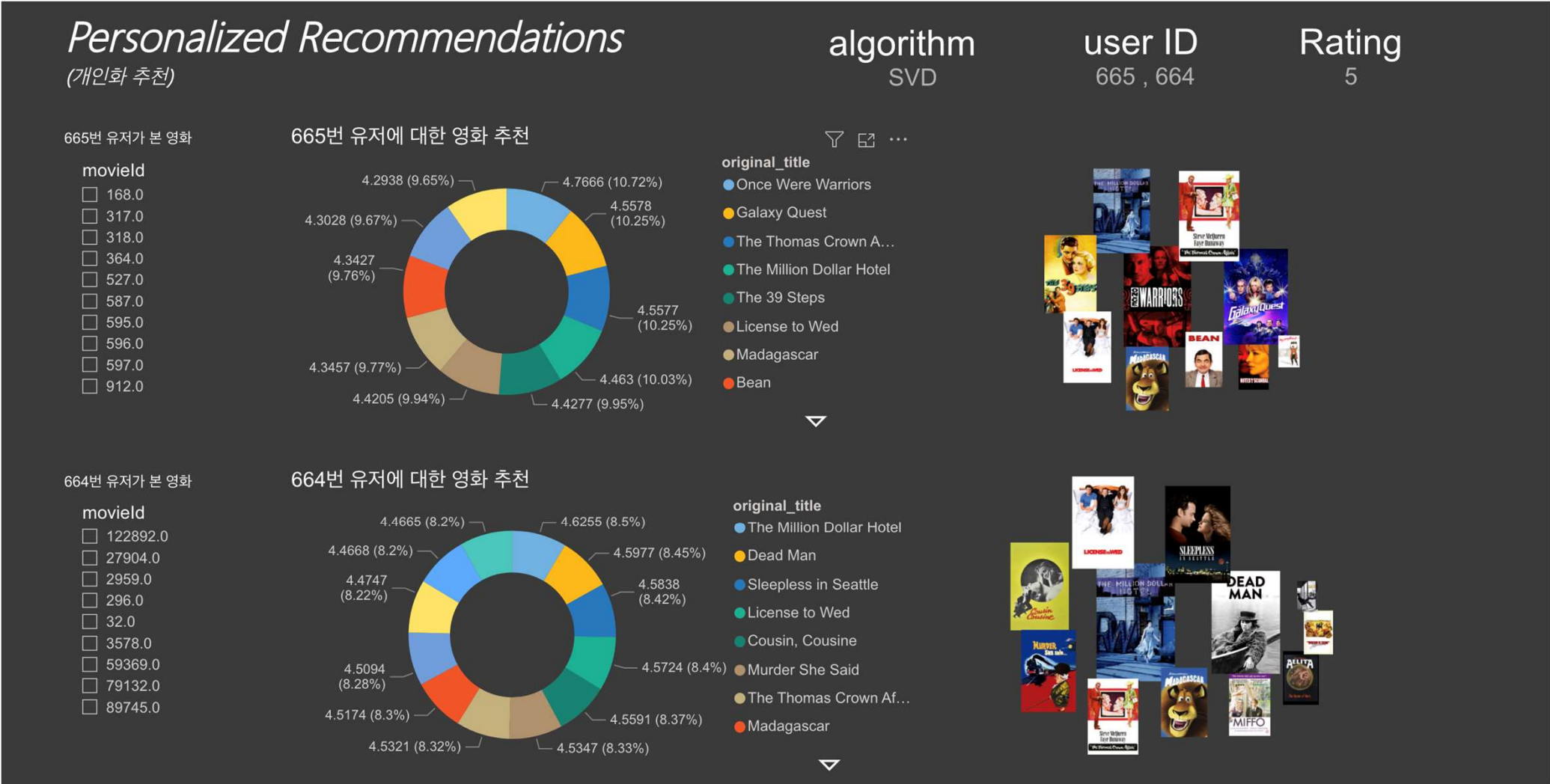
Ex) '빠른 인사이트 찾기(Quick Insight)' 기능

위와 같은 이유로 추후에도 현업에서 Power BI를 프로젝트에 접목시키고 효과적 시각화 할 수 있기 때문에 사전 학습 및 숙련 차원에서 Power BI를 선정함



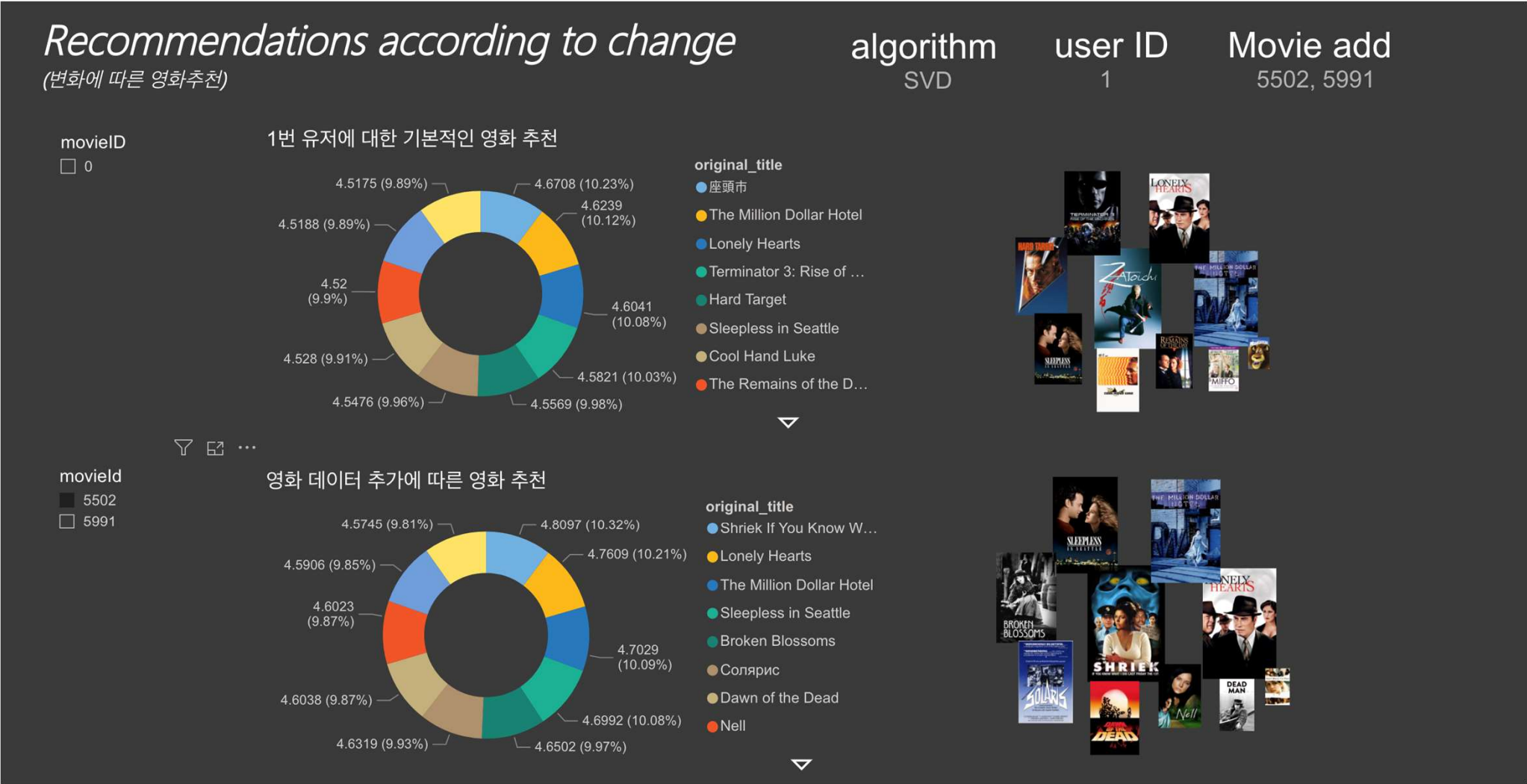
4. 프로젝트 수행 결과_ 시각화

결과 1. 개인화 추천



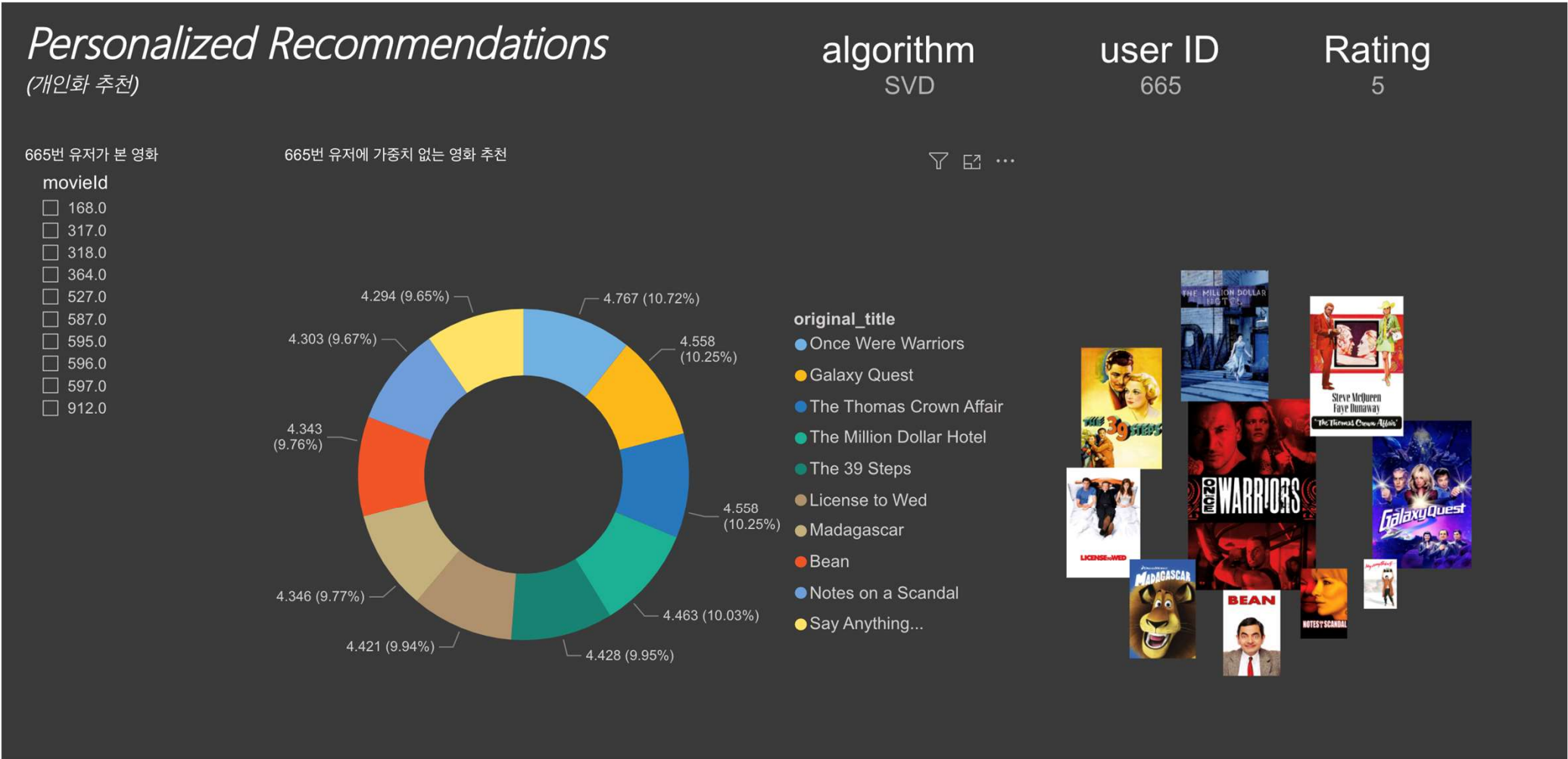
4. 프로젝트 수행 결과_ 시각화

결과 2. 개인화 추천



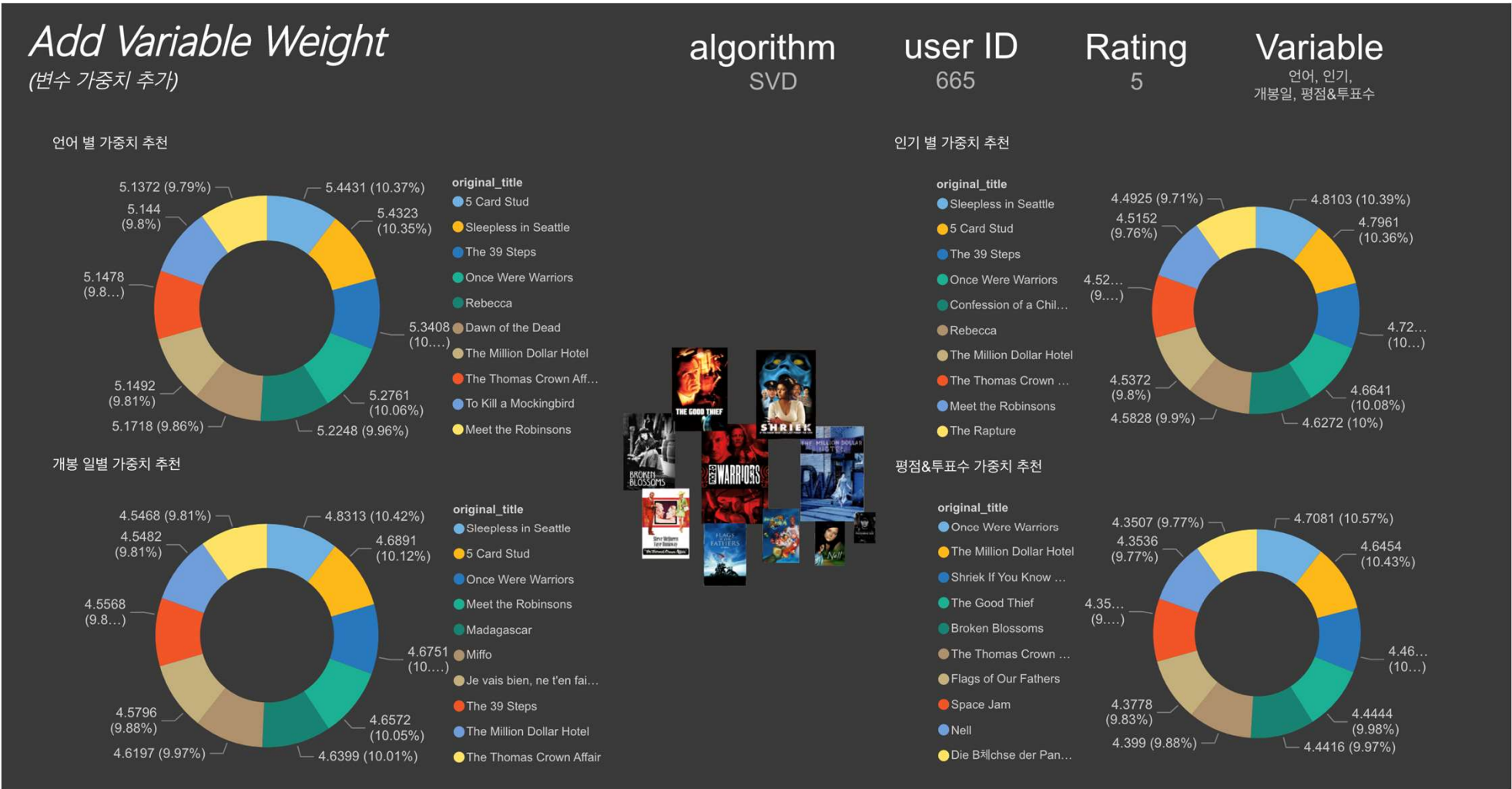
4. 프로젝트 수행 결과_ 시각화

결과 3-1. 개인화 추천 (665번 유저에게 가중치 없는 영화 추천)



4. 프로젝트 수행 결과_ 시각화

결과 3-2. 변수 가중치 추가



4. 프로젝트 수행 결과_시각화

결과 4. 알고리즘 추천

Algorithm Recommendation

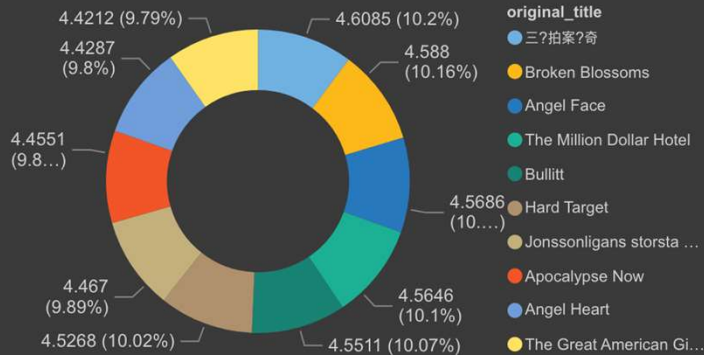
(알고리즘 추천)

algorithm
NMF, SLOPE, ALS, SVD

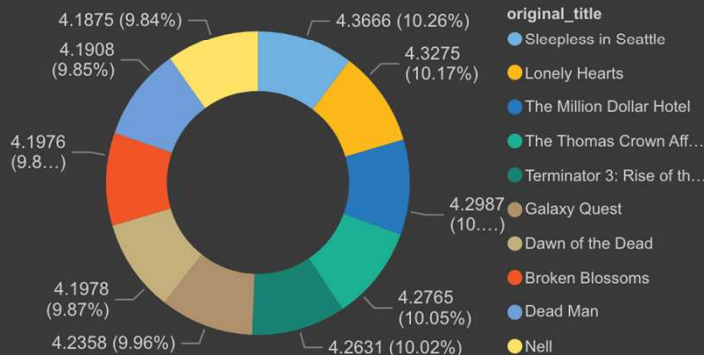
user ID
665

Rating
5

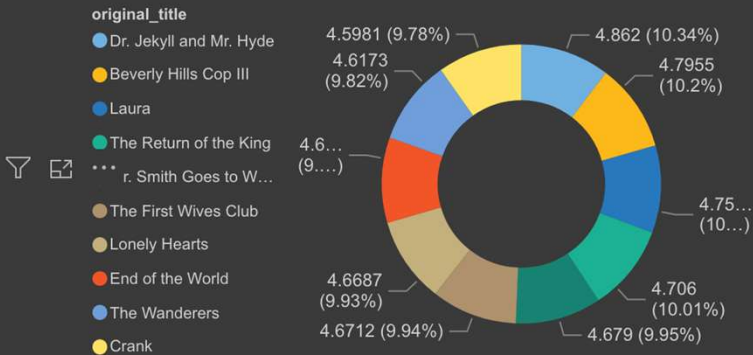
NMF 알고리즘 추천



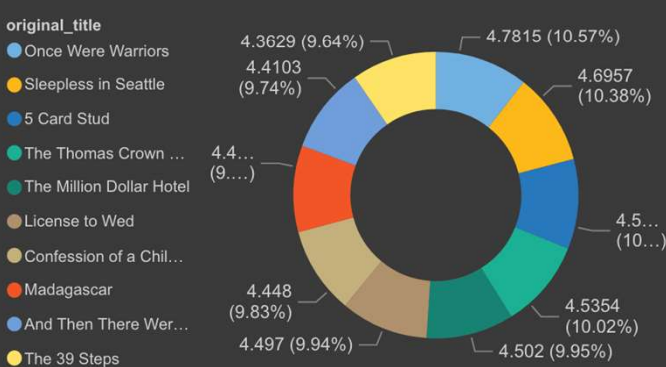
ALS 알고리즘 추천



SLOPE 알고리즘 추천

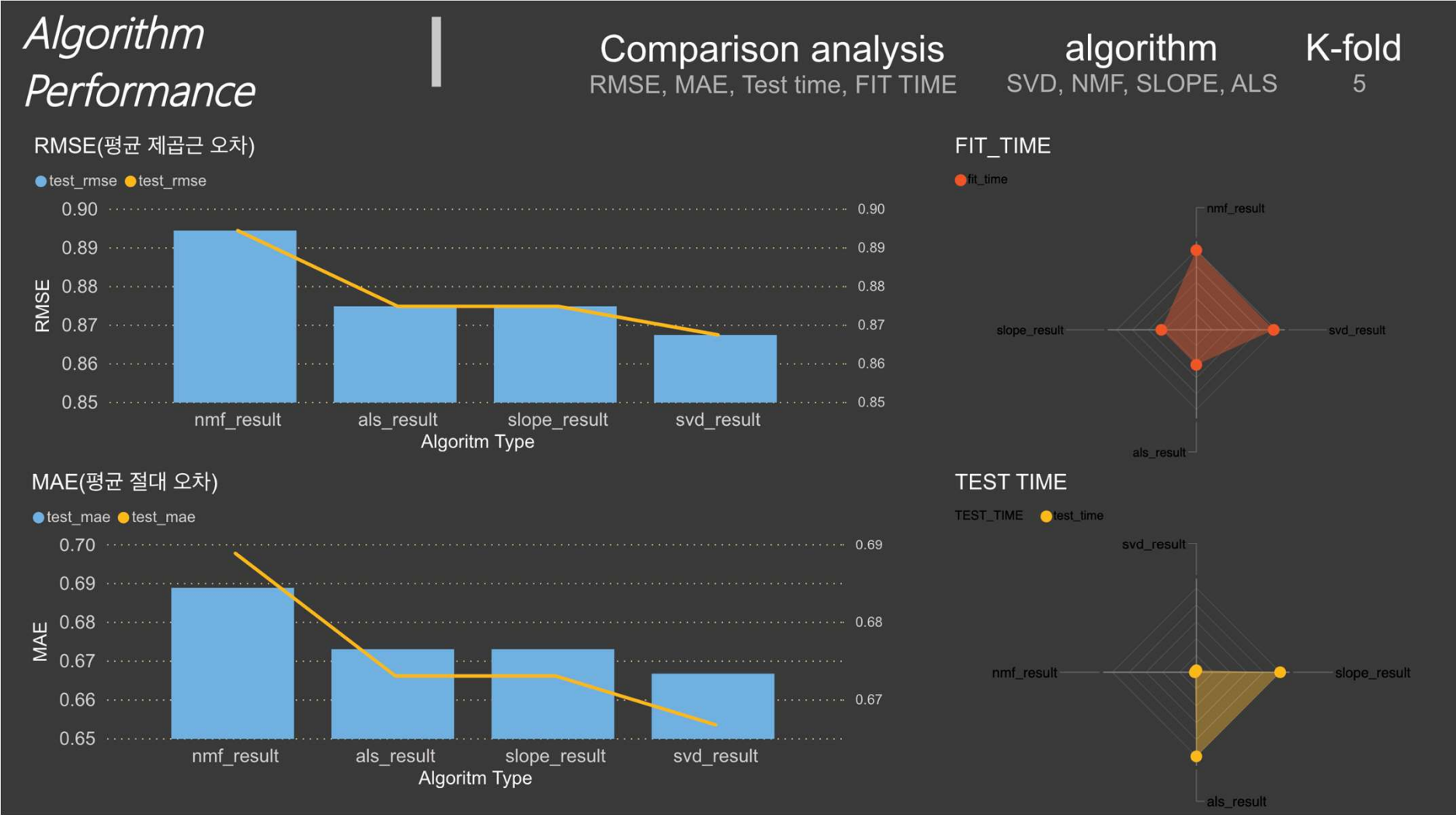


SVD 알고리즘 추천



4. 프로젝트 수행 결과_시각화

결과 5. 알고리즘 별 성능 차이



5. 느낀 점

구분	내용
프로젝트 수행상 어려움 극복 사례	<ul style="list-style-type: none"> 모두에게 생소한 추천 시스템이라는 개념과 Pandas, Numpy, Power BI와 같은 도구들을 사용하여 프로젝트를 진행했다. 특히 하이브리드 추천 모델 및 알고리즘 부분은 개인이 전부 감당하기에 무리가 있는 부분이었다. 따라서, 이 어려움을 극복하기 위해 각자의 상황과 능력에 적합한 역할이 배분 되었고, 메신저 및 Zoom 소회의실을 이용한 원활한 커뮤니케이션을 통해 서로 어려움을 공유하고 해결책을 찾아나가는 팀워크를 보였다.
프로젝트에서 잘한 부분	<ul style="list-style-type: none"> 팀원간의 팀워크가 잘 맞았고 업무 역할과 일정 분배가 잘 되었다. 또한, 열정을 가지고 각자가 맡은 임무를 끝까지 해내려 노력하였다. 학습한 내용 또한 서로 공유하며 시너지를 발휘했다.
프로젝트에서 아쉬운 부분	<ul style="list-style-type: none"> 데이터의 크기가 너무 커서 표본으로 진행한 점이 아쉽다. 심지어 제공된 AWS에 적용시켰을 때도 프로그램이 멈추는 현상도 발생했다. 또한 최신 데이터가 아니라는 아쉬움도 있다. 프로젝트를 진행하는 동시에 알고리즘 및 통계 수학적 지식을 학습했기에 오랜 시간이 소요됐다.

5. 느낀 점

구분	내용
<p>프로젝트를 통한 진로설계, 취업분야 탐색 및 결정 등 도움</p>	<ul style="list-style-type: none"> • 현업에서 팀워크를 활용해 프로젝트를 수행하는 것을 간접적으로 경험 할 수 있어 좋았고, 프로젝트 기간 데이터 분석 학습을 통해 이 산업에 대한 이해도가 높아질 수 있었다. 또한, 더욱 흥미롭고 진로 설계에 있어 큰 동기부여가 되었다. • 더 나아가 부족한 점을 깨닫고 앞으로의 공부 계획에 긍정적인 영향을 미쳤다. 추후 대학원 진학 혹은 취업 사이에서 결정을 내리는데 조금이나마 도움이 될 것 같다
<p>기타</p>	<ul style="list-style-type: none"> • 떠오르는 분야인 만큼 준비만 잘하면 길은 많다고 생각한다. 사용자 입장에서 접했던 추천 시스템을 실제로 구현해보니 복잡한 과정이 많고 알고리즘마다 예측 값이 다른 점을 관찰했다. 다음 기회에는 유튜브와 같은 영상을 분석하여 다양한 추천 변수를 추가해보고 싶다. • 세미 프로젝트 진행을 통해 생각했던 것보다 훨씬 더 많은 것을 얻게 됐다. 스스로 많이 부족하다고 생각했지만 조금이나마 더 자신감을 얻을 수 있었고, 뛰어난 팀원들을 보며 강한 동기부여가 되었고 같이 열심히 할 수 있었다. 이러한 코드들을 보면 시작하기 전부터 혼란스럽고 겁부터 먹었는데 하나하나 스스로 검색하고 주석을 달며 이해하는 접근 방식 등 프로젝트 내용 뿐만 아니라 팀원들에게 자세에 대한 배움도 많이 얻었다. • 소중한 기회를 제공해주신 강사님께 정말 감사합니다. ☺

감사합니다

