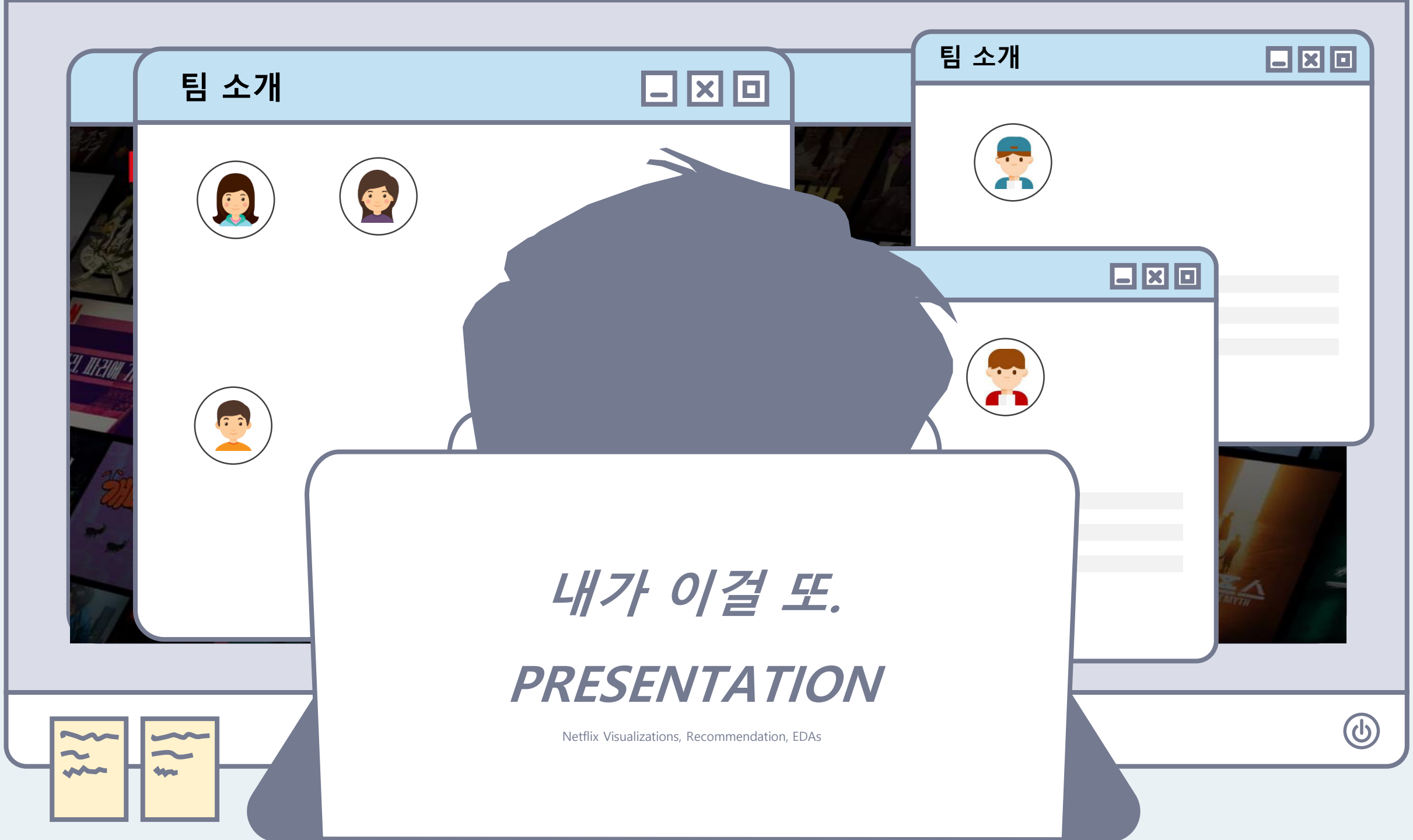

팀 소개
팀 소개
내가 이걸 또.
PRESENTATION
Netflix Visualizations, Recommendation, EDAs

목차

기획의도

- 신종 코로나바이러스 감염(코로나19) 여파로 사람들이 집에서 보내는 시간이 늘면서, 지난 1년간 온라인 스트리밍 서비스 수요가 대폭 증가함
- 대표적인 OTT 서비스 플랫폼인 NETFLIX에 등단코자 하는 신인 감독, 시나리오 작가, 영상 콘텐츠 제작사 등 관련 업계 종사자들에게 어떠한 형태, 어떤 장르의 영상을 제작하면 좋을지 방향성 제시

Netflix

주제 및 타겟

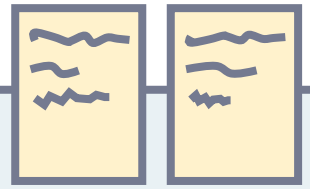
✓ **주제**

- 영상 제작자들을 위한 추천 시스템

✓ **타겟**

- Netflix에 영상을 업로드할 신입 영상 감독 및 시나리오 작가

데이터 소개



STEP1. 데이터 소개

Data_Info

변수이름	설명	개수
show_id	구분자	7787
type	Movie / TV Show	7787
title	제목	7787
director	감독	5398
cast	배우	7069
country	나라	7280
date_added	넷플릭스 업로드 날짜	7777
release_year	출시연도	7787
rating	등급	7780
duration	상영시간	7787
listed_in	장르	7787
description	줄거리	7787

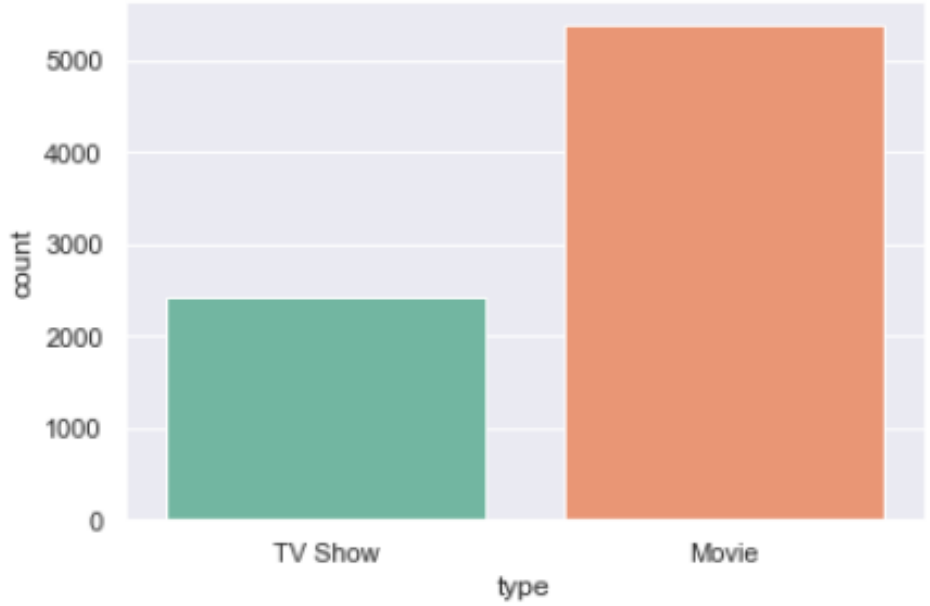
- ✓ Netflix에서 서비스 되고 있는 콘텐츠 데이터
- ✓ Last update : 2021-01-19
- ✓ Netflix 전문 조사 사이트 Fixable에서 수집된 자료
- ✓ 2018년, Netfilx의 TV 프로그램 수가 2010년 이후로 3배 가까이 증가했다는 흥미로운 보고서 발표
- ✓ 스트리밍 서비스의 영화 수는 2010년 이후 2,000 편 이상 감소한 반면, TV 프로그램 수는 3배 정도 증가

데이터 탐색 및 시각화

STEP2. 탐색 및 시각화

TV Show vs 영화

```
: 1  # TV vs 영화
  2  sns.set(style="darkgrid")
  3  ax = sns.countplot(x="type", data=netflix_data1, palette="Set2")
```



PPT PRESENTATION

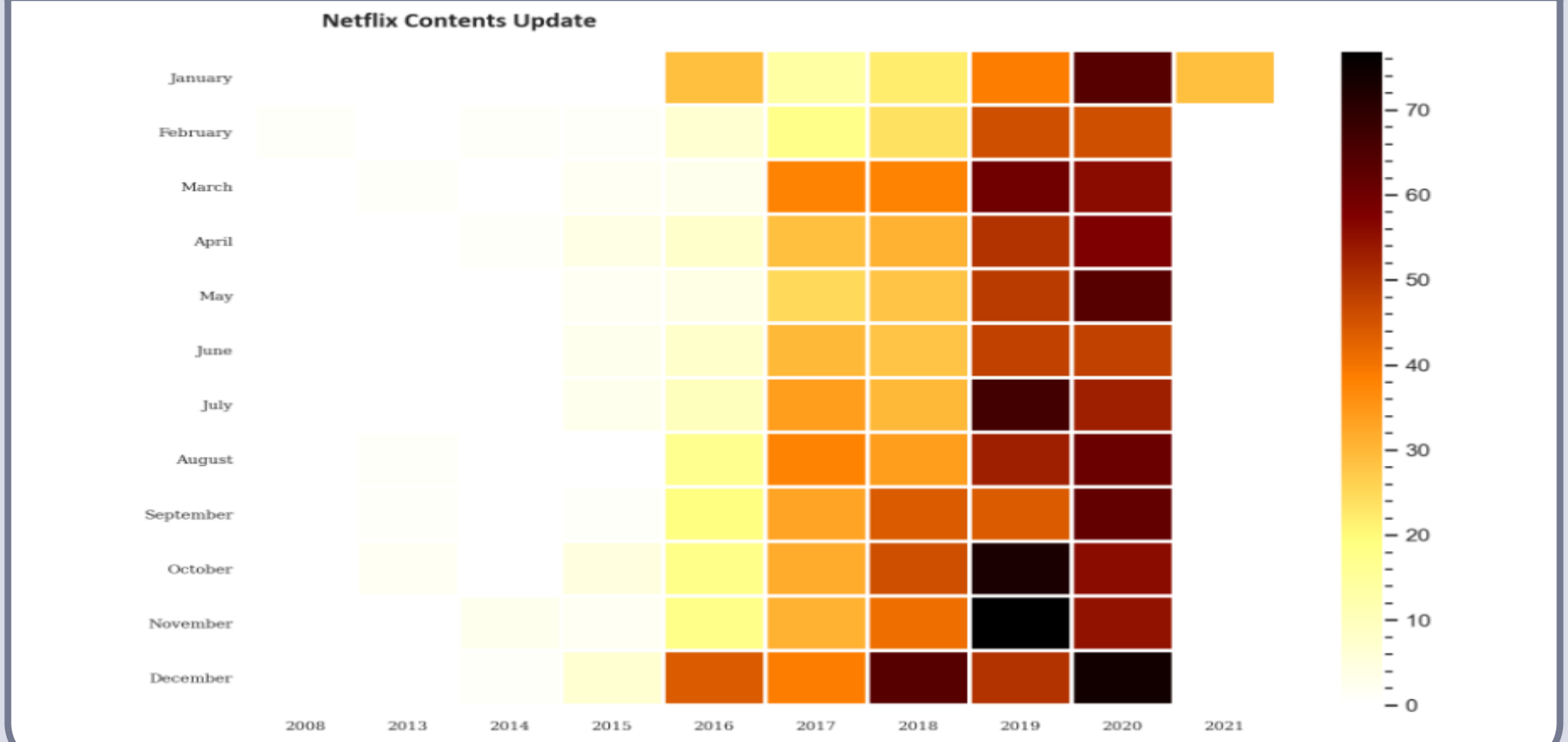
STEP2. 탐색 및 시각화

국가별콘텐츠 업로드 수

	country
United States	799
India	701
United Kingdom	107
Canada	56
Philippines	50
Spain	40
South Korea	36
Indonesia	35
France	33
United Kingdom, United States	31
Australia	30

STEP2. 탐색 및 시각화

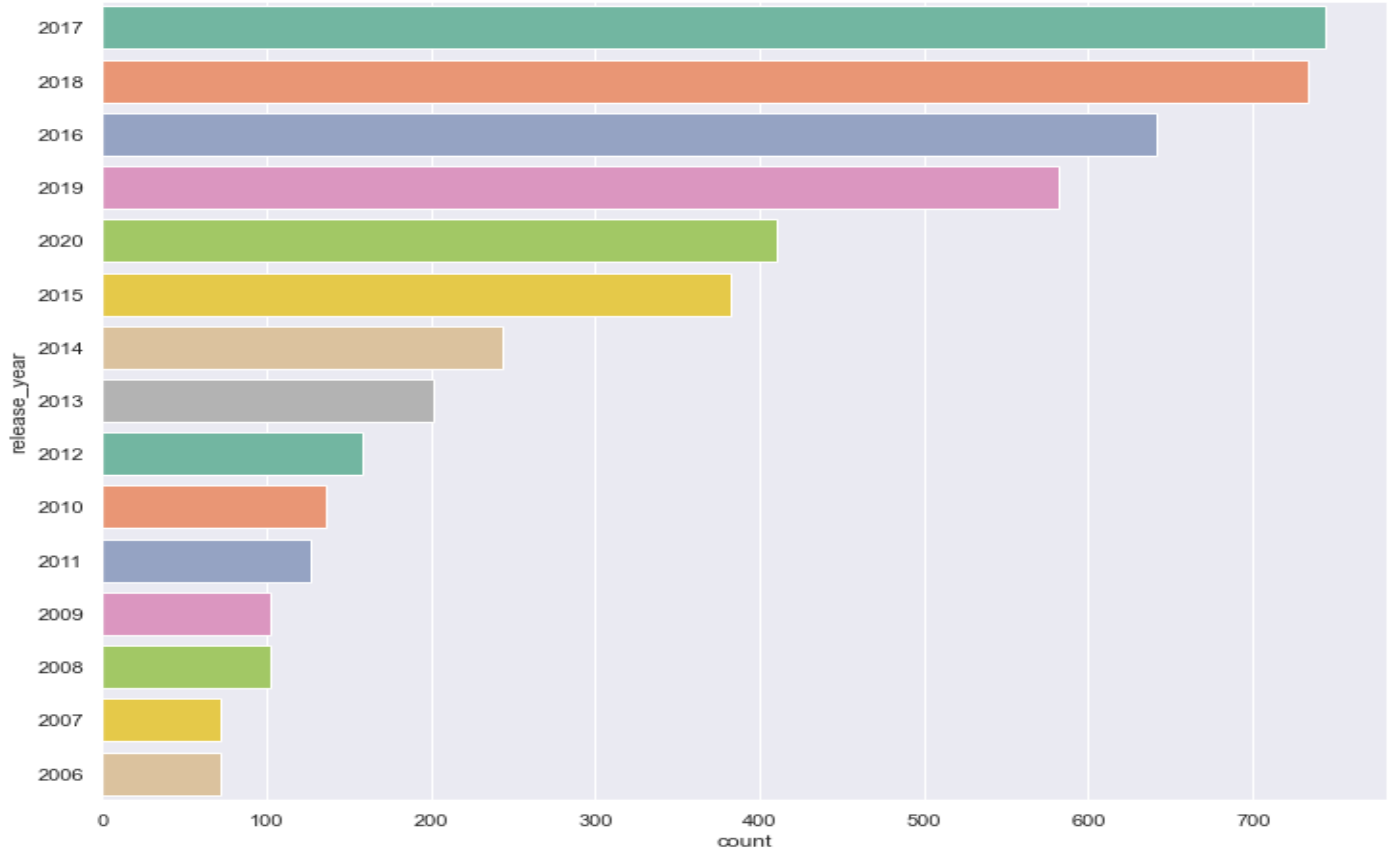
월별 컨텐츠 업데이트 비율 (Heatmap)



PPT PRESENTATION

STEP2. 탐색 및 시각화

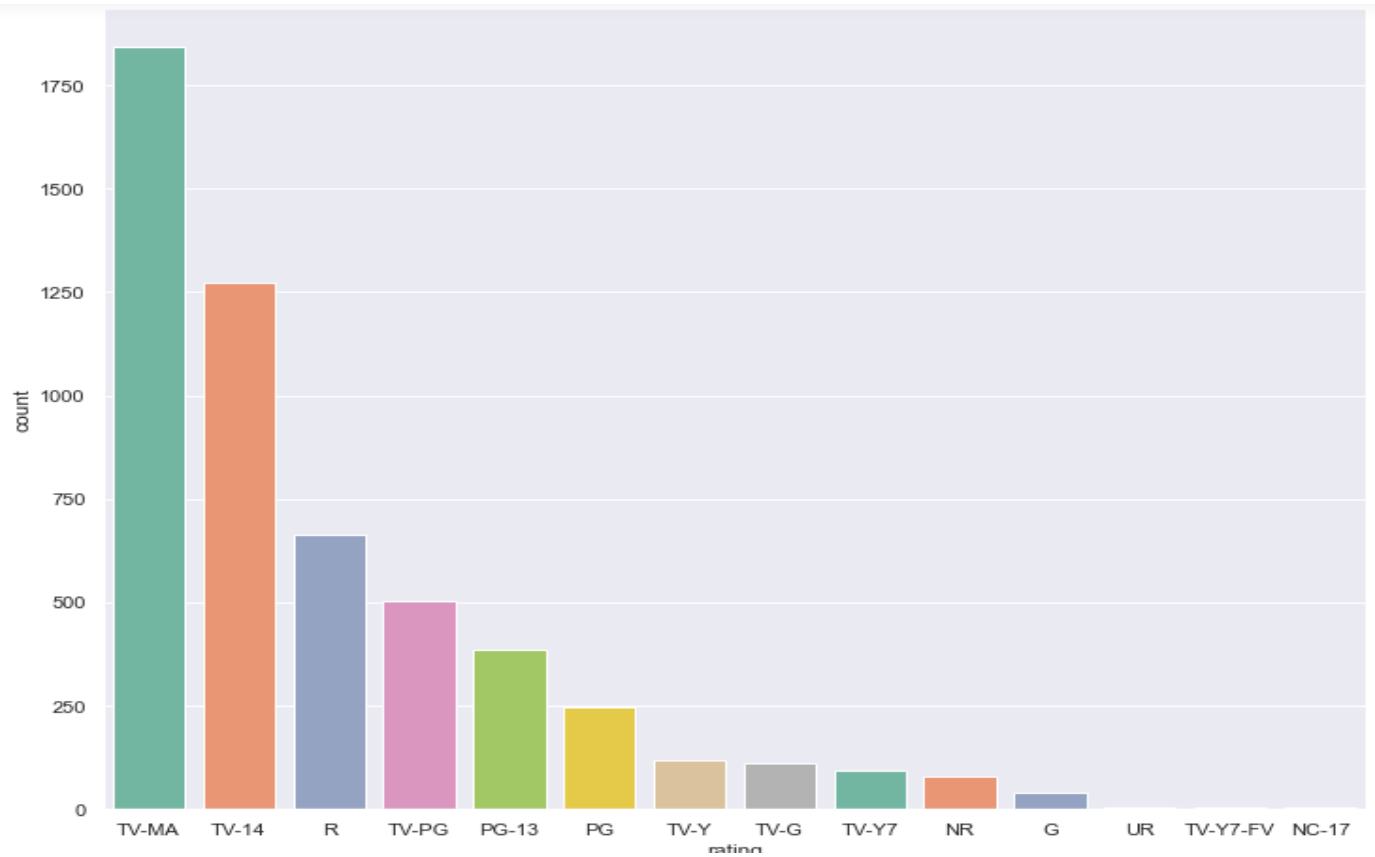
가장 많은 영화가 개봉된 연도순



PPT PRESENTATION

STEP2. 탐색 및 시각화

프로그램 등급별 제작 수



PPT PRESENTATION

STEP2. 탐색 및 시각화

프로그램별 등급 확인

```
# 프로그램 등급
print(netflix_shows['rating'])
```

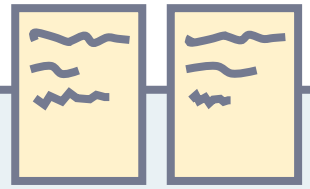
```
0       TV-MA
5       TV-MA
11      TV-MA
12      TV-MA
16      TV-14
        ...
7767    TV-PG
7775    TV-Y7
7777    TV-Y7
7779    TV-MA
7785    TV-PG
Name: rating, Length: 2410, dtype: object
```

TV 프로그램 등급(미국)

TV-Y	영유아를 위한 프로그램	
TV-Y7	7세 이상의 어린이를 위한 프로그램	
	FV	매우 경미한 가상의 폭력 묘사가 있음.
TV-G	모든 연령이 시청할 수 있는 프로그램. 다만 어린이를 대상으로 하지는 않았다.	
TV-PG	어린이가 시청하려면 보호자 지도가 권장되는 프로그램	
	D	성적으로 부적절한 언어가 약하게 사용됨
	L	욕설, 비속어 등이 약하게 사용됨. 예를 들어 crap, ass, Hell, Damn 등의 약한 욕설이나 잔인함을 암시하는 대사 등이 있다.
	S	성적인 묘사가 약간 있음. 네모바지 스폰지밥처럼 엉덩이가 묘사되는 정도?
	V	폭력적인 묘사가 약간 있음. 의도적인 유혈효과가 나오거나 모방위험성이 있는 현실적인 폭행묘사 및 약한 살인묘사 및 사망의 묘사 총기난사등이 나오면 최소가 TV-PG이다.
TV-14	14세 미만의 어린이 혹은 청소년이 시청하려면 보호자 지도가 권장되는 프로그램	
	D	성적으로 부적절한 언어가 사용됨. "Penis", "Vagina" 등.
	L	욕설, 비속어 등이 사용됨. "Ass", "Bitch", "Piss", "Shit" 수준. 혹은, 매우 심한 욕설을 문맥상 충분히 유추 가능하게 검열한 정도[23]
	S	성적인 묘사가 있음. 베드신이나 신체 노출이 있을 수 있으나 그리 심하지 않은 경우.
	V	폭력적인 묘사가 있음. 특히 피가 나오면 무조건 TV-14이다.
TV-MA	17세 미만의 어린이 혹은 청소년한테 부적절한 프로그램.[24]	
	L	심한 욕설이 있음. 혹은, "fuck", "cunt" 등 심한 욕설의 짧은 검열이나 검열이 없음.
	S	심하게 성적인 묘사가 있음. 노골적인 베드신이나 신체 노출 등. 유료채널에서는[25] 유두나 성기 노출도 있을 수 있다.
	V	매우 폭력적인 묘사가 있음. 총상을 매우 자세하고, 길게 묘사한 경우.

출처:
https://namu.wiki/w/%EC%98%81%EC%83%81%EB%AC%BC%20%EB%93%B1%EA%B8%89%20%EC%A0%9C%EB%8F%84/%EB%AF%B8%EA%B5%AD

데이터 전처리



STEP3. 데이터 전처리

결측치 확인 및 처리

```
show_id         0
type            0
title           0
director     2389
cast          718
country       507
date_added     10
release_year    0
rating          7
duration        0
listed_in       0
description     0
dtype: int64
```

결측치는 전부 직접 검색을 통해 추가했으며, 그럼에도 값이 존재하지 않는 데이터들은 0으로 처리

STEP3. 데이터 전처리

2. 추가 및 제거 변수

제거한 변수	
show_id	구분자
title	제목
director	감독
cast	배우
date_added	넷플릭스 업로드 날짜
description	줄거리

추가한 변수	
All_star	평점

3. 최종 사용 변수

변수이름	설명
type	Movie / TV Show
country	나라
release_year	출시연도
rating	등급
duration	상영시간
listed_in	장르
All_star	평점

Y값 : All_Star(평점)

STEP3. 데이터 전처리

IMDb 평점 데이터 크롤링

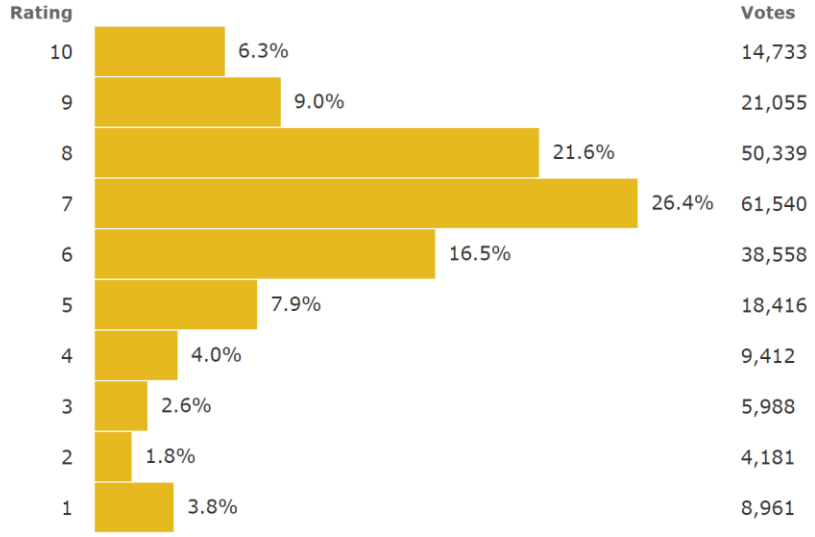
Us (II) (2019)

User Ratings

★ 6.8 ☆ Rate

IMDb Users

233,183 IMDb users have given a weighted average vote of 6.8 / 10



Rating By Demographic

	All Ages	<18	18-29	30-44	45+
All	6.8 233,183	7.2 300	7.1 49,059	6.7 76,294	6.4 18,448
Males	6.8 125,452	7.1 191	7.1 35,760	6.7 61,714	6.4 14,935
Females	6.9 29,035	7.8 46	7.1 10,311	6.8 12,377	6.5 2,951

Top 1000 Voters	US Users	Non-US Users
6.5 491	7.0 32,196	6.7 80,348

PPT PRESENTATION

STEP2. 탐색 및 시각화

IMDs 별점 분석

```
imdb_ratings=pd.read_csv('data/IMDb ratings.csv',usecols=['weighted_average_vote'])
imdb_titles=pd.read_csv('data/IMDb movies.csv', usecols=['title','year','genre'])

ratings = pd.DataFrame({'Title':imdb_titles.title,
                        'Release Year':imdb_titles.year,
                        'Rating': imdb_ratings.weighted_average_vote,
                        'Genre':imdb_titles.genre})

ratings.drop_duplicates(subset=['Title','Release Year','Rating'], inplace=True)
ratings.shape
```

```
C:₩Users₩cecil₩anaconda3₩lib₩site-packages₩IPython₩core₩interactiveshell.py:3071: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

```
(85852, 4)
```

```
ratings.dropna()
joint_data=ratings.merge(netflix_data1,left_on='Title',right_on='title',how='inner')
joint_data=joint_data.sort_values(by='Rating', ascending=False)
```

STEP3. 데이터 전처리

1. 제목에 붙어있던 공백문자(₩u200b) 삭제

```
title_c = list(np.array(data["title"].tolist()))
h=[]
for i in range(0, len(title_c)):
    if "₩u200b" in title_c[i]:
        title_c[i] = title_c[i].replace("₩u200b",'')
        h.append(title_c[i])
title_c
```

2. 고유번호 파악을 위한 html 주소 저장

```
# 평점조회  URL를 들어가기 위해 필요한 제목의 고유번호를 얻기 위한 작업
# 제목을 검색한 주소를 가져온다.
html=[]
for i in tqdm_notebook(range(0, len(title_c))):
    element = driver.find_element_by_name('q')
    element.send_keys(title_c[i])
    element.submit()
    juso = driver.current_url
    html.append(juso)
```

3. 제대로 입력되지 않은 주소 처리

```
# 응답을 받지 못하여 주소가 제대로 입력되지 않은 것들 처리
for i in range(0, len(html)):
    if "https://www.imdb.com/find?q=&ref_=nv_sr_sm" in html[i]:
        print(title_c[i])
```

```
html[3] = "https://www.imdb.com/find?q=9&ref_=nv_sr_sm"
html[11] = "https://www.imdb.com/find?q=1983&ref_=nv_sr_sm"
html[799] = "https://www.imdb.com/find?q=Becoming&ref_=nv_sr_sm"
html[1244] = "https://www.imdb.com/find?q=Catching+the+Sun&ref_=nv_sr_sm"
html[1480] = "https://www.imdb.com/find?q=Concrete+Football&ref_=nv_sr_sm"
html[1540] = "https://www.imdb.com/find?q=Criminal%3A+Spain&ref_=nv_sr_sm"
html[2731] = "https://www.imdb.com/find?q=High+Score&ref_=nv_sr_sm"
html[5692] = "https://www.imdb.com/find?q=Sofia+the+First&ref_=nv_sr_sm"
html[6187] = "https://www.imdb.com/find?q=The+Burial+of+Kojo&ref_=nv_sr_sm"
```

STEP3. 데이터 전처리

4. 작품 고유번호 크롤링

```
▼<div class="findSection">
  ▶<h3 class="findSectionHeader">…</h3>
  ▼<table class="findList">
    ▼<tbody>
      ▼<tr class="findResult odd">
        ▶<td class="primary_photo">…</td>
        ▼<td class="result_text">
          <a href="/title/tt4922804/?ref_=fn_al_tt_1">3%</a> == $
          " (2016) (TV Series) "
        </td>
      </tr>
```

5. 작품 고유번호 a에 담기

```
# 영화 고유번호 크롤링
a = []
for i in tqdm_notebook(range(0,len(html))):
    try :
        url = html[i]
        page = urlopen(url)
        soup = BeautifulSoup(page, "html.parser")
        title= list(soup.find_all("td", class_="result_text")[0])
        title1 = str(title[1]).split("/")
        title2 = title1[2]
        a.append(title2)
    except IndexError:
        a.append(" ")

len(a)
```

6. 사람의 고유번호 작품 고유번호로 수정

```
# 사람이름으로 검색된 것들 영화제목으로 수정

for i in tqdm_notebook(range(0, len(s_a))):
    try :
        url = s_a[i]
        page = urlopen(url)
        soup = BeautifulSoup(page, "html.parser")
        titles_a= list(soup.find_all("td", class_="result_text"))
        titles_a = str(titles_a).split("/")
        for j in range(0, len(titles_a)):
            if """<td class="result_text">""" in titles_a[j-2] and "title" in titles_a[j-1]:
                a[i] = titles_a[j]
                break

    except ValueError:
        pass
```

PPT PRESENTATION

STEP3. 데이터 전처리

7. All_star 변수 생성

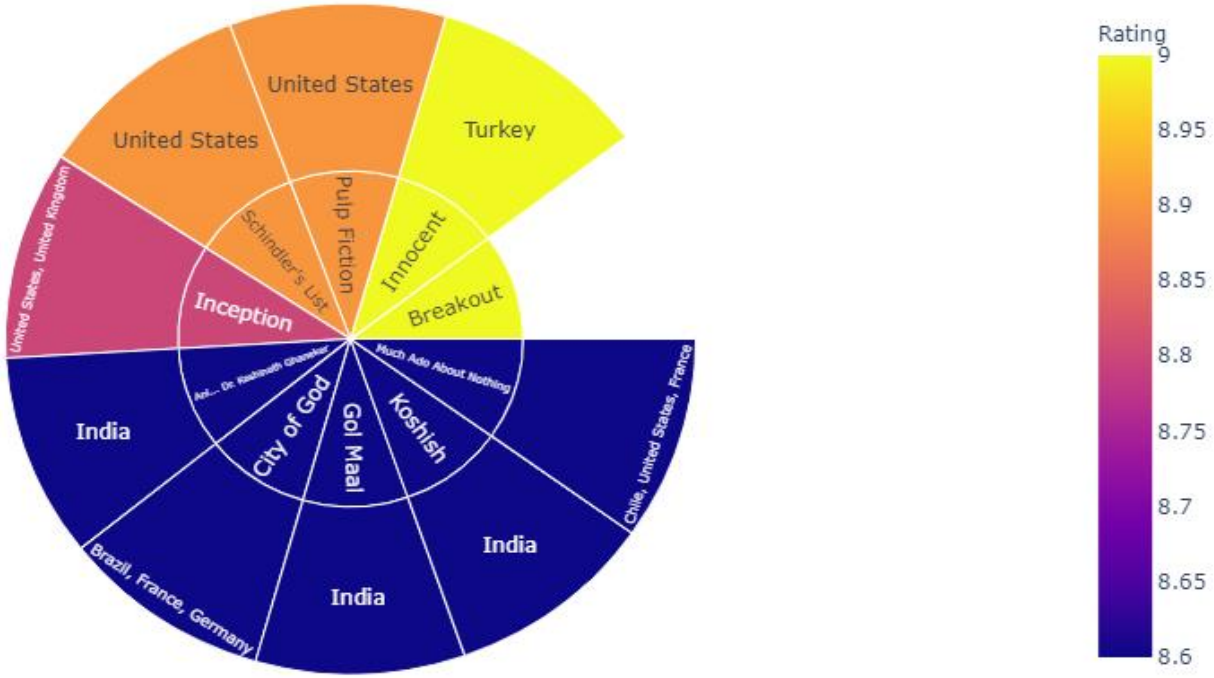
```
All_star=[]
def append_0(a):
    All_star.append(a)
```

```
# 평점 조회
for i in tqdm_notebook(range(0, len(a))):
#for i in tqdm_notebook(range(0, 20)):
    if a[i] == " ":
        append_0(0)
        continue
    try :
        url = "https://www.imdb.com/title/{}/ratings?ref_=tt_ov_rt".format(a[i])
        page = urlopen(url)
        soup = BeautifulSoup(page, "html.parser")
        star = list([soup.find_all("td", "ratingTable")[n].get_text() for n in range(0, 18)])
        for j in range(0, len(star)):
            star[j] = star[j].replace('₩n', '').split()
            if  "-" in star[j]:
                star[j] = 0, 0
        star = np.array(star).flatten().tolist()
        All_star.append(star[0])
    except (IndexError):
        append_0(0)
```

- ✓ 고유번호를 평점주소에 입력하여 평점을 크롤링
- ✓ 크롤링 한 평점은 'All_star' 라는 변수에 추가
- ✓ 만약 평점사이트가 없는 작품은 0점으로 처리

STEP2. 탐색 및 시각화

상위 10개 컨텐츠 그래프



PPT PRESENTATION

STEP3. 데이터 전처리

Country 변수 전처리

```
#  data1 : 나라이름이 누락되어있는 것을 수작업으로 찾아 넣은 파일
data1 = pd.read_csv("C:/finalProject/country_final1.csv")
```

```
data["country"] = data1["country"]
```

```
# 0 : United States
# 1 : India
# 2 : United Kingdom
# 3 : Japan
# 4 : South Korea
# 5 : Canada
# 6 : Spain
# 7 : France
# 8 : 공동 제작
# 9 : 상위8개 국가를 제외한 나머지 국가

country2 = dict(data["country"].value_counts())
country_item = list(country2.items())

for i in range(0, len(data["country"])):
    if "," in str(data["country"][i]):
        data["country"][i] = 8  # 공동제작
        continue
    for j in range(0, 7):
        if data["country"][i] == country_item[j][0]:
            data["country"][i] = j
            break
    else :
        data["country"][i] = 9 # 상위 8개국가를 제외한 나머지 국가
```

Country 데이터 중에서 누락되어 있는 데이터들은 수작업으로 찾아 넣음

상위 8개 국가들과 공동제작, 그 외의 나라들로 분류

0 : United States
1 : India
2 : United Kingdom
3 : Japan
4 : South Korea
5 : Canada
6 : Spain
7 : France
8 : 공동 제작
9 : 상위8개 국가를 제외한 나머지 국가

STEP3. 데이터 전처리

Rating 변수 전처리

Rating	분류	범주화
18세 미만	TV-MA, R, NC-17	0
14세 미만	TV-14, PG-13	1
어린이 시청 시 보호자 지도 필요	TV-PG, TV-G, PG	2
7세 미만	TV-Y7-FV, TV-Y7	3
영유아	TY-Y	4
등급 없음	NR, UR	5
전체관람	나머지	6

STEP3. 데이터 전처리

Duration 변수 전처리 : TV Show Season

```
# duration 에서 Season으로 있는것을 1화당 평균 분으로 교체하기 위해 Seasons의 title 찾기
season=[]
for i in range(0, len(data3["duration"])):
    if "Seasons" in data3["duration"][i] or "Season" in data3["duration"][i]:
        season.append(data3["title"][i])
    elif "min" in data3["duration"][i]:
        season.append("")
    else :
        season.append("")
season
```

```
# 위에서 구한 고유번호를 통해 1화당 평균 시간 크롤링

for i in tqdm_notebook(range(0, len(season))):
    if a[i] == " ":
        time_min.append("주소가 없습니다.")
        continue
    try :
        if season[i] != "":
            url = "https://www.imdb.com/title/{}/?ref_=fn_al_tt_1".format(a[i])
            page = urlopen(url)
            soup = BeautifulSoup(page, "html.parser")
            time = soup.find("div", "subtext")
            time_str = str(time)
            time_list = list(time)
            time_split = str(time_list).split(">")
            time_split = str(time_list).replace('₩n', '').split()
            if "datetime" in time_str :
                for j in range(0, len(time_split)):
                    if "datetime" in time_split[j-1] and "min" not in time_split[j+1]:
                        time_min.append(time_split[j])
                    elif "datetime" in time_split[j-1] and "min" in time_split[j+1] :
                        time_min.append(time_split[j]+time_split[j+1])
            elif "datetime" not in time_str:
                time_min.append("없음")

        else:
            time_min.append("")
    except IndexError:
        time_min.append(data["title"][i])

time_min
```

```
# 데이터 duration 교체
for i in range(0, len(data3["duration"])):
    if time_min[i] != "":
        data3["duration"][i] = time_min[i]
data3
```

STEP3. 데이터 전처리

Duration 변수 전처리

```
for i in range(0, len(data3["duration"])):
    if "h" in data3["duration"][i]:
        if "min" in data3["duration"][i]:
            k = data3["duration"][i]
            x = k.split("h")
            y = x[1].split("min")
            z = (int(x[0])*60) + int(y[0])
            data3["duration"][i] = str(z)+"min"
        elif "min" not in data3["duration"][i]:
            k = data3["duration"][i]
            x = k.split("h")
            z = (int(x[0])*60)
            data3["duration"][i] = str(z)+"min"
```

시간을 60분 단위로 환산하여 분 단위로 통합
min 단위 제거

Type 변수 전처리

```
for i in range(0, len(data["type"])):
    if data["type"][i] == "Movie":
        data["type"][i] = 0
    elif data["type"][i] == "TV Show":
        data["type"][i]= 1
```

Movie는 0, TV Show는 1로 변환

STEP3. 데이터 전처리

listed_in 변수 Genre 변수로 전처리 : 가변수화

International TV Shows	TV Dramas	TV Sci-Fi & Fantasy	Dramas	...	TV Shows	Classic Movies	Cult Movies	TV Horror	Stand-Up Comedy & Talk Shows	Teen TV Shows	Stand-Up Comedy	Anime Features	TV Thrillers	Classic & Cult TV
1	1	1	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	...	0	0	0	0	0	0	0	0	0	0


Data ML

STEP3. 데이터 전처리

평점(All_Star) 변수 전처리

```
count    7787.000000
mean        6.269192
std         1.765881
min         0.000000
25%         5.700000
50%         6.600000
75%         7.400000
max         9.700000
Name: All_star, dtype: float64
```

Summary를 통해 평점 변수의 기준을 중앙값 6.5로 설정

전처리코드

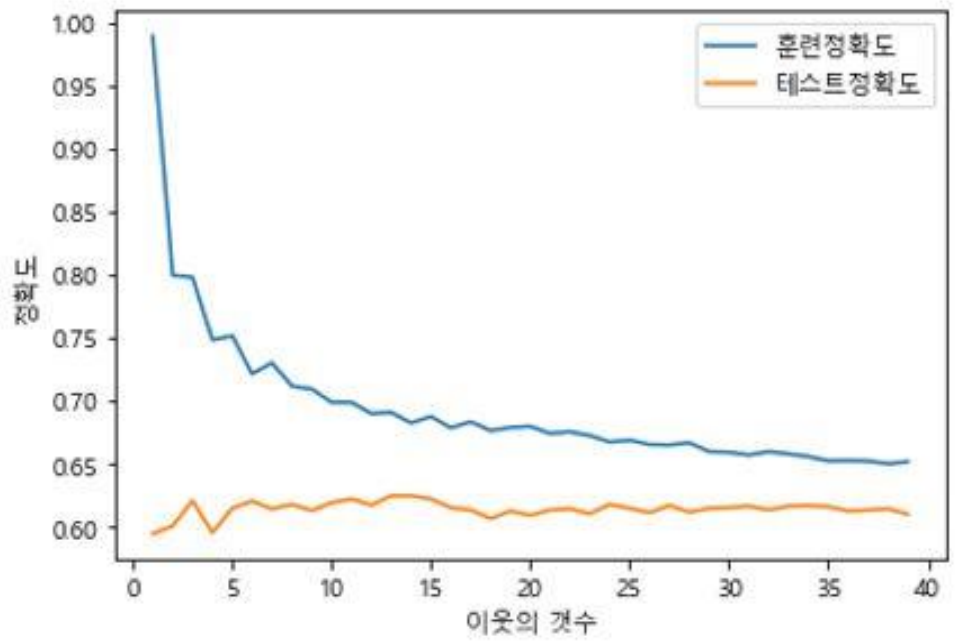
```
# 총평점 6.5점이하는 0으로 나머지는 1로
for i in range(0, len(data["All_star"])):
    if data["All_star"][i] <= 6.5:
        data["All_star"][i] = 0
    elif data["All_star"][i] > 6.5:
        data["All_star"][i] = 1
data
```

```
data = data.astype({"All_star":"int"})
```

총 평점 6.5점 이하는 0으로
나머지는 1로 바꾼 후 int형으로 타입 변경

STEP4. Machine Learning : KNN

KNN(K - Nearest neighbors)



- Min-Max Normalization을 통해 정규화
- 7:3비율로 train, test 셋 분리
- 1~40까지 임의의 이웃의 개수로 학습 시킨 모델의 정확도 확인

STEP4. Machine Learning : KNN

최적의 KNN 모델

최적의 하이퍼 파라미터
{'n_neighbors': 17, 'p': 1}
최고 예측 정확도 : 0.6572481033293622

훈련 정확도

0.7018348623853211

테스트 정확도

0.6422764227642277

정오분류표

	precision	recall	f1-score	support
0	0.66	0.59	0.62	1168
1	0.63	0.69	0.66	1169
accuracy			0.64	2337
macro avg	0.64	0.64	0.64	2337
weighted avg	0.64	0.64	0.64	2337

STEP4. Machine Learning : 로지스틱 회귀분석

로지스틱 회귀 모델

```
([['Penalty : 0.0000,   훈련 점수 : 0.509,   테스트 점수 : 0.534'],
  ['Penalty : 0.0000,   훈련 점수 : 0.509,   테스트 점수 : 0.534'],
  ['Penalty : 0.0001,   훈련 점수 : 0.545,   테스트 점수 : 0.567'],
  ['Penalty : 0.0010,   훈련 점수 : 0.652,   테스트 점수 : 0.645'],
  ['Penalty : 0.0100,   훈련 점수 : 0.660,   테스트 점수 : 0.659'],
  ['Penalty : 0.1000,   훈련 점수 : 0.672,   테스트 점수 : 0.672']],
 [['Penalty : 0.00,   훈련 점수 : 0.509,   테스트 점수 : 0.534'],
  ['Penalty : 0.00,   훈련 점수 : 0.509,   테스트 점수 : 0.534'],
  ['Penalty : 0.00,   훈련 점수 : 0.545,   테스트 점수 : 0.567'],
  ['Penalty : 0.00,   훈련 점수 : 0.652,   테스트 점수 : 0.645'],
  ['Penalty : 0.01,   훈련 점수 : 0.660,   테스트 점수 : 0.659'],
  ['Penalty : 0.10,   훈련 점수 : 0.672,   테스트 점수 : 0.672']])
```

- Min-Max Normalization을 통해 정규화
- 7:3비율로 train, test 셋 분리
- 각각의 하이퍼 파라미터에 따른 훈련 점수와 테스트 점수 비교 후 최적의 파라미터 결정

STEP4. Machine Learning : 로지스틱 회귀분석

최적의 로지스틱 회귀 모델

분석 방법 : [LogisticRegression]

Penalty=0.001의 테스트 정확도	0.6407659392383398
Penalty=0.01의 테스트 정확도	0.6589259734702609

- Penalty=0.01의 정확도가 약간 더 높은 결과를 보여주기 때문에 최적의 파라미터로 결정

정오분류표

	precision	recall	f1-score	support
0	0.70	0.61	0.65	1152
1	0.66	0.74	0.70	1185
accuracy			0.68	2337
macro avg	0.68	0.68	0.68	2337
weighted avg	0.68	0.68	0.68	2337

STEP4. Machine Learning : SVM

최적의 파라미터 찾기 (Grid Search)

```
{'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}
```

: Grid Search를 이용하여 잠재적 Parameter들의
후보군들의 조합 중 가장 Best 조합을 찾아줍니다.

최적의 파라미터 적용 정확도

훈련 결과 : 0.7526605504587156
테스트 결과 : 0.6705177578091 57

17757809157

	ecision	recall	f1-score	support
	.68	0.65	0.67	1168
	.67	0.69	0.68	1169
			0.67	2337
	.67	0.67	0.67	2337
	.67	0.67	0.67	2337

PPT PRESENTATION

STEP4. Machine Learning : SVM

최적의 파라미터 찾기 (Grid Search)

{'C': 1000, 'gamma': 0.1, 'ke

: Grid Search를 이용하여 잠재적 Pa
후보군들의 조합 중 가장 Best 조합

최적의 파라미터 적용 정확도

훈련 결과 : 0.75
테스트 결과 : 0.

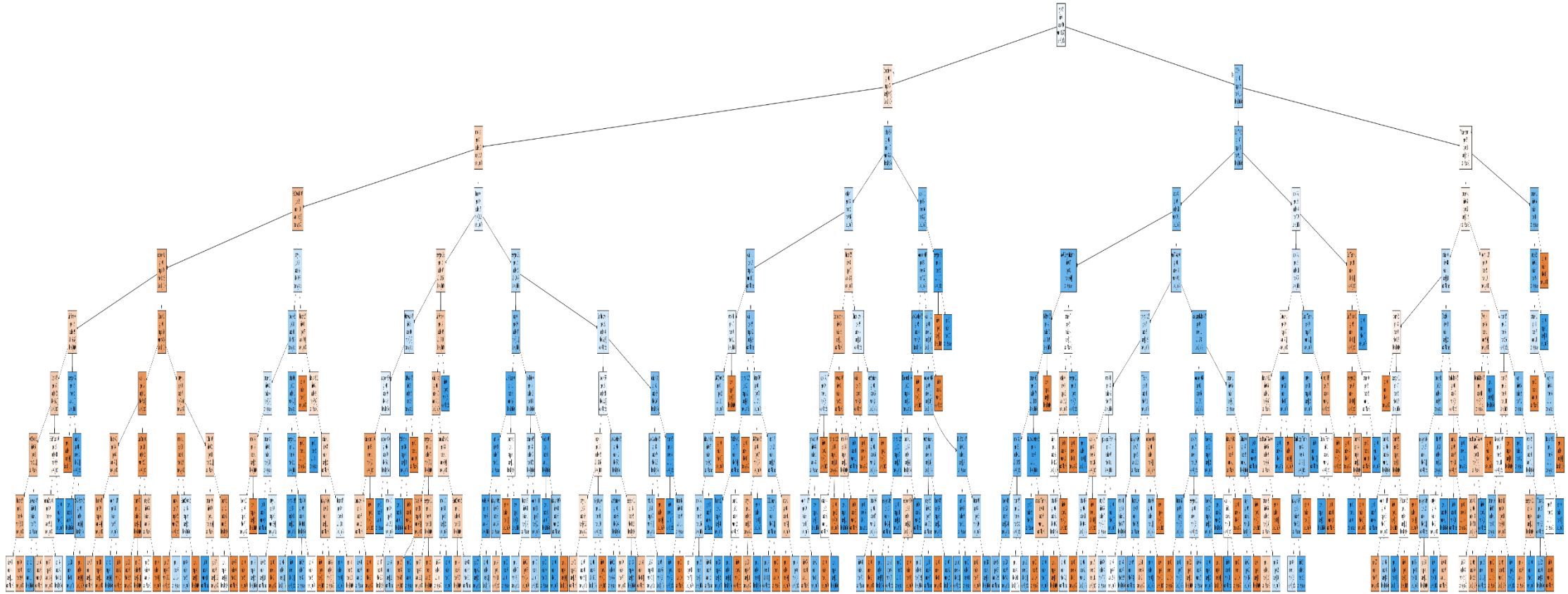
정오분류표

정확도 : 0.6705177578091 57

	precision	recall	f1-score	support
class 0	0.68	0.65	0.67	1168
class 1	0.67	0.69	0.68	1169
accuracy			0.67	2337
macro avg	0.67	0.67	0.67	2337
weighted avg	0.67	0.67	0.67	2337

STEP4. Machine Learning : 의사결정나무

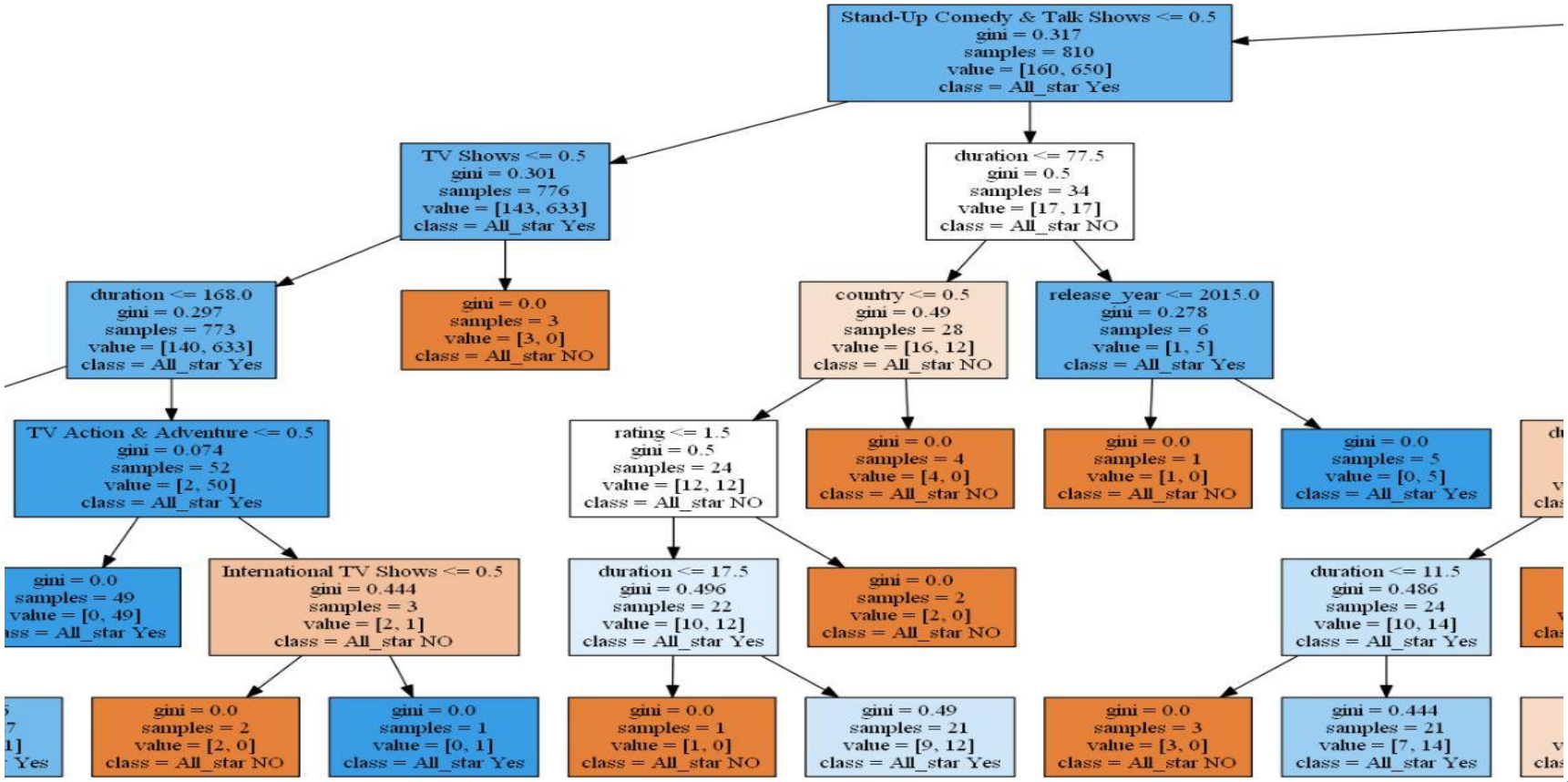
최적의 의사결정나무 그림



PPT PRESENTATION

STEP4. Machine Learning : 의사결정나무

최적의 의사결정나무 확대 그림



뿌리 설명

- 분류기준
- Gini
- Samples
- Value
- class

STEP4. Machine Learning : 의사결정나무

성능평가

훈련 정확도 : 0.7563302752293578
테스트 정확도 : 0.6829268292682927

정오분류표

	precision	recall	f1-score	support
class 0	0.66	0.70	0.68	1125
class 1	0.70	0.67	0.69	1212
accuracy			0.68	2337
macro avg	0.68	0.68	0.68	2337
weighted avg	0.68	0.68	0.68	2337

결론&한계점

STEP5. 결론

모델링 결과

1. KNN, 로지스틱 회귀모델, SVM, 의사결정나무. 이 4가지 분류모델의 테스트 정확도 측면에서 의사결정나무가 별다른 차이 없이 높은 수치를 보였고, 정오분류표에서도 좋은 수치를 보여줬기 때문의 최선의 모델로 정하였습니다.

2. 또한 의사결정나무는 다른 모델들과는 다르게 새로운 데이터에 적합 시키기가 매우 쉽고, 해석이 용이하기 때문에 사용자가 쉽게 이해할 수 있다는 장점이 있습니다.

3. 결과적으로 의사결정나무 모델을 기준을 결정하는데 중요한 역할을 하는 상위3개 변수는 영상길이, Movie인지 TV Show인지, 출시연도 였습니다.

STEP6. 한계점

윤정

- ✓ 데이터가 특수해서 추가 데이터 확보에 어려웠다는 점
- ✓ 시간 부족으로 인해 이용 가치 있는 변수를 사용하지 못한 점

병우

- ✓ 더 많은 독립변수가 있었더라면 더 좋은 결과가 나올 수 있었을 거 같은데 아쉬웠다.
- ✓ 시간이 더 있었더라면 description 변수도 활용할 수 있었을텐데 시간이 부족하여 사용하지 못했던 것이 아쉬웠다.

태영

- ✓ 비슷한 종류의 데이터를 얻기 어려워 다른 데이터와의 비교분석을 하지 못한 점
- ✓ 상품화를 위한 플랫폼을 구축할 수 있었으면 좋았을 것 같다.
- ✓ 역시, 시간부족

STEP6. 한계점

태동

- ✓ 로우 데이터의 양이 적어서 해당 데이터로만 반복 학습을 할 경우, 과적합의 우려도 있어 다양한 방법의 분석을 시도하지 못하였다.
- ✓ 평점을 연령제한별 구분을 나눠 분석을 해보거나 혹은 연도별에 따른 평점 변화 등을 추가적으로 분석을 해보았으면 좋았을 것 같다는 아쉬움이 남았습니다.

민준

- ✓ 변수가 결측치도 많고 좋지 않아서 의결조율은 물론 분석하기 어려웠다.

참고자료

- https://www.kaggle.com/shivamb/netflix-shows/metadata
- https://namu.wiki/w/%EC%98%81%EC%83%81%EB%AC%BC%20%EB%93%B1%EA%B8%89%20%EC%A0%9C%EB%8F%84/%EB%AF%B8%EA%B5%AD
- https://en.wikipedia.org/wiki/Television_content_rating_system

감사합니다.

내가 이걸 또.

PRESENTATION

Netflix Visualizations, Recommendation, EDAs