Kaggle Notebook

Netflix Visualizations, Recommendation, EDA

Netflix 영화 및 TV 프로그램 데이터와 IMDb(인터넷 영화 데이터베이스)에서 제공하는 평점, 리뷰 데이터를 이용하여 IMDb 와 Netflix의 관계를 알아보고, 최종적으로 Content Based 추천 시스템을 구축

사용된 데이터

- Netflix_titles.csv
- IMDb rattings.csv
- IMDb movies.csv
- Books.csv

https://www.kaggle.com/niharika41298/netflix-visualizations-recommendation-eda/notebook

Data

```
netflix_overall=pd.read_csv("netflix_titles.csv")
netflix_overall.head()
```

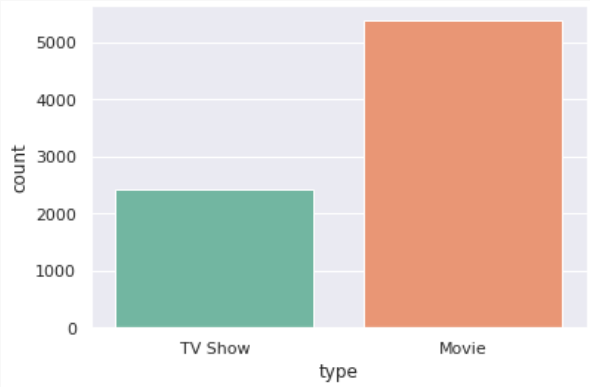
	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R...	Brazil	August 14, 2020	2020	TV-MA	4 Seasons	International TV Shows, TV Dramas, TV Sci-Fi &...	In a future where the elite inhabit an island ...
1	s2	Movie	7:19	Jorge Michel Grau	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	Mexico	December 23, 2016	2016	TV-MA	93 min	Dramas, International Movies	After a devastating earthquake hits Mexico Cit...
2	s3	Movie	23:59	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	December 20, 2018	2011	R	78 min	Horror Movies, International Movies	When an army recruit is found dead, his fellow...
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	November 16, 2017	2009	PG-13	80 min	Action & Adventure, Independent Movies, Sci-Fi...	In a postapocalyptic world, rag-doll robots hi...
4	s5	Movie	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	United States	January 1, 2020	2008	PG-13	123 min	Dramas	A brilliant group of students become card-coun...

Netflix_titles.csv
- show_id: 영화/tv 프로그램에 대한 고유 ID
- type: Identifier – 영화/tv 프로그램
- title: 영화/tv 프로그램의 제목
- director: 영화 감독
- Cast: 영화/tv 프로그램에 참여한 배우들
- Country: 영화/tv 프로그램 제작 국가
- Date_added: 넷플릭스에 추가된 날짜
- release_year: 영화/tv 프로그램 실제 개봉한 연도
- Rating: 영화/tv 프로그램의 TV등급
- Duration: 총 기간 – 러닝 타임(분) or 시즌의 수
- listed_in: 장르
- Description: 영화/tv 프로그램 요약 설명

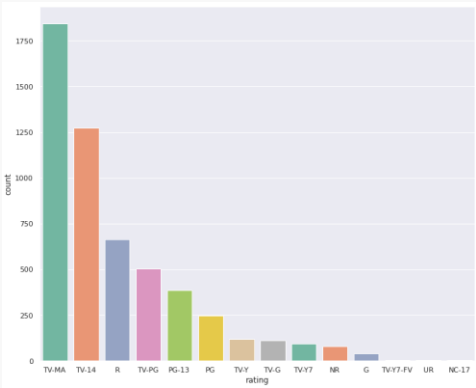
IMDb rattings.csv
- Title → Title
- Year → Release Year
- Genre → Genre

IMDb movies.csv
- weighted_average_vote → Rating

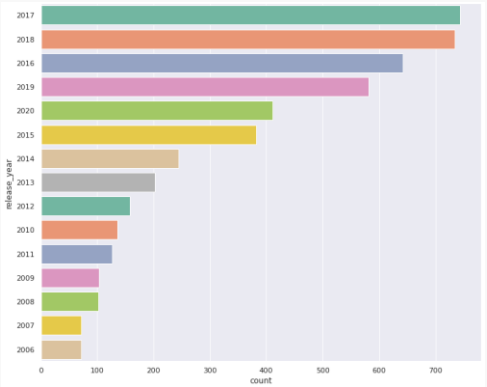
EDA



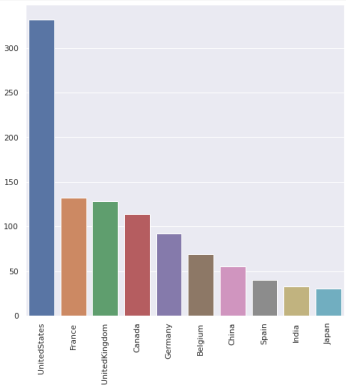
Movies vs TV Shows



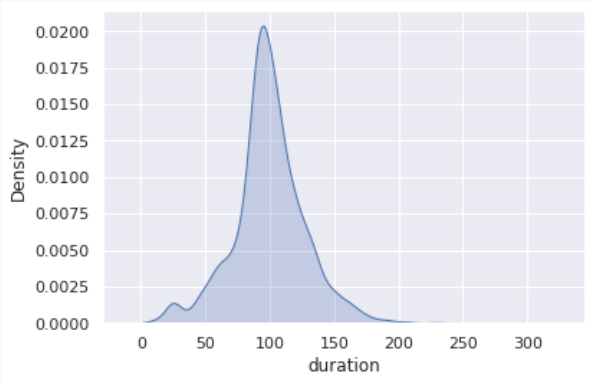
Movie ratings



Year wise



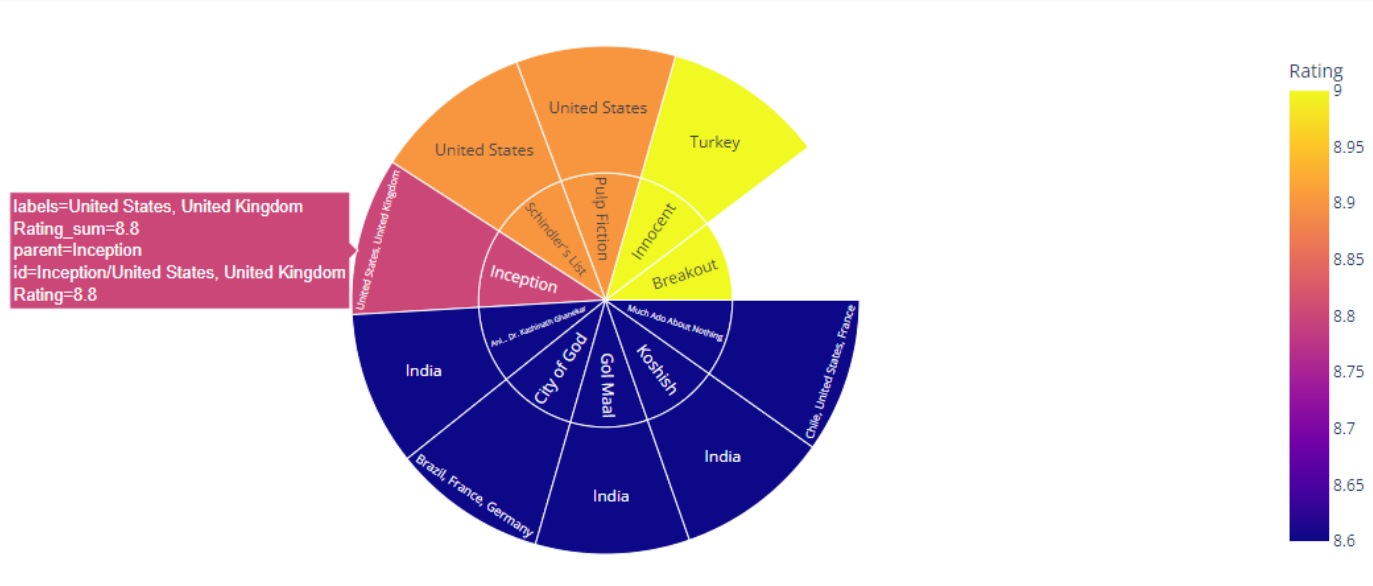
Top 10 movie content creating countries



duration of movies

- Content를 가장 많이 만든 나라, 시즌의 수가 가장 많은 tv 프로그램, 가장 최근에 출시한 content, 가장 오래된 content 등 다양한 EDA를 진행
- NA 값에 대해 fillna()를 이용 → 범주형 변수는 'Unknown', 수치형 변수는 '0'의 값을 넣어 진행

Visualizations Plotly를 이용하여 시각화



파이차트

```
# Top rated 10 movies on Netflix are:
import plotly.express as px
top_rated=joint_data[0:10] # IMDb 데이터를 joint한 데이터
fig =px.sunburst(
    top_rated,
    path=['title','country'],
    values='Rating',
    color='Rating')
fig.show()
```

- IMDb의 등급데이터와 넷플릭스 데이터의 내부 조인을 수행하여 IMDb에서 등급이 지정된 콘텐츠와 넷플릭스에서 시청할 수 있는 콘텐츠 중 가장 좋은 등급이 매겨진 top 10을 알아봄
- 마우스를 오버하면 등급, 국가, 제목, 고유 id 정보 제공

Visualizations Plotly를 이용하여 시각화

트리맵

```
# Content in France are:
import plotly.express as px
netflix_fr=netflix_overall[netflix_overall['country']=='France']
nannef=netflix_fr.dropna()
fig = px.treemap(nannef, path=['country','director'],
                 color='director', hover_data=['director','title'],
                 color_continuous_scale='Purples')
fig.show()
```

- 프랑스의 감독별 content count를 표현
- 자료상 영화가 극단적으로 많고 적은 감독의 분포는 보이지 않음
- 마우스를 오버하면 감독, 제목, 보조감독, 영화수, 국가, 고유 id 제공

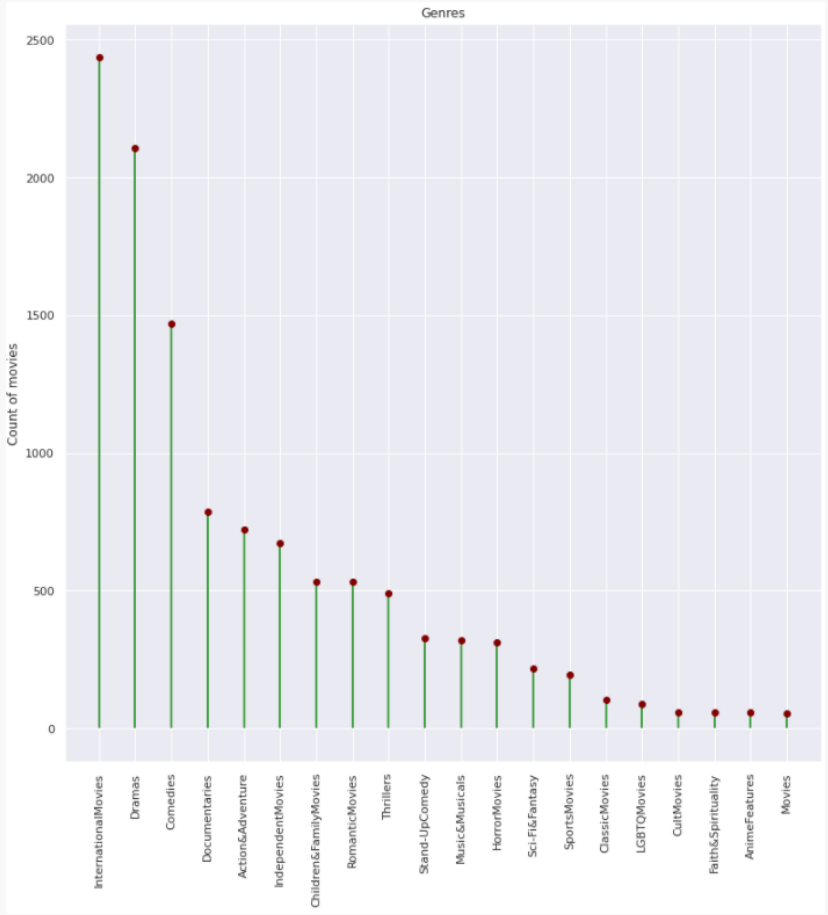
Visualizations

WordCloud

```
# WordCloud for Genres are:
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from PIL import Image

text = list(set(gen))
plt.rcParams['figure.figsize'] = (13, 13)

#assigning shape to the word cloud
mask = np.array(Image.open('star.png'))
wordcloud = WordCloud(max_words=1000000,background_color="white",
                      mask=mask).generate(str(text))

plt.imshow(wordcloud,interpolation="bilinear")
plt.axis("off")
plt.show()
```

- 콘텐츠 장르를 워드클라우드를 통해 표현
- 워드클라우드는 png 이미지를 이용하여 원하는 모양으로 만들 수 있음

Visualizations

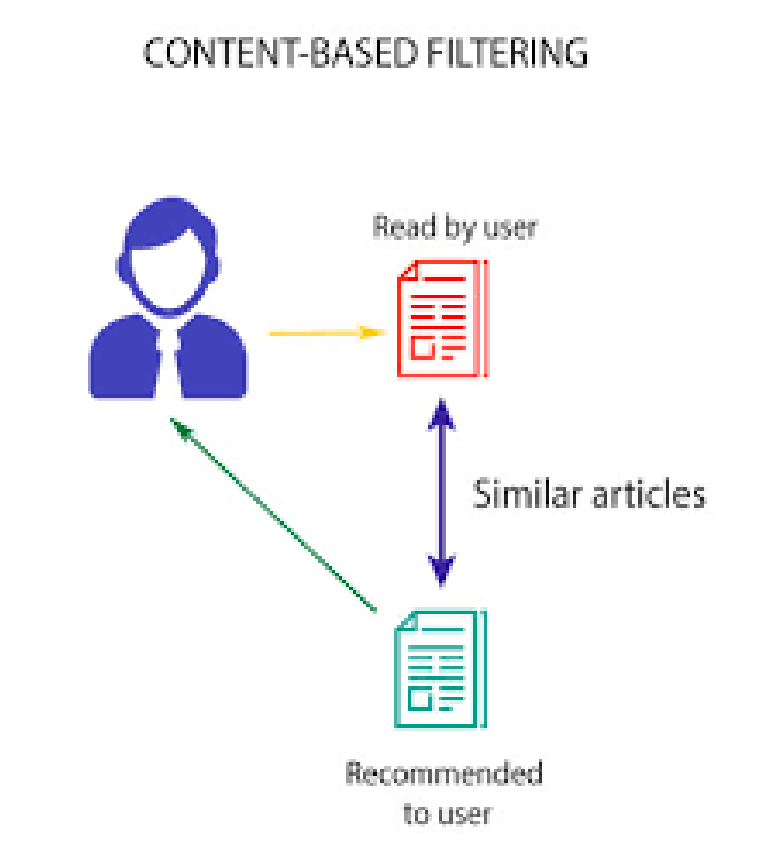


Lollipop plot

```
# Genres vs their count on Netflix are:
g={k: v for k, v in sorted(g.items(), key=lambda item: item[1], reverse= True)}

fig, ax = plt.subplots()

fig = plt.figure(figsize = (10,10))
x=list(g.keys())
y=list(g.values())
ax.vlines(x, ymin=0, ymax=y, color='green')
ax.plot(x,y, "o", color='maroon')
ax.set_xticklabels(x, rotation = 90)
ax.set_ylabel("Count of movies")
# set a title
ax.set_title("Genres");
```

- 장르별 contents count가 많은 순으로 정렬하여 시각화

Recommendation System



사용자가 이전에 구매한 상품 중에서 좋아한 상품과 유사한 상품들을 추천하는 방법

```
# Recommendation System (Content Based):
from sklearn.feature_extraction.text import TfidfVectorizer

# 영어 불용어 제거
tfidf = TfidfVectorizer(stop_words='english')

# NaN을 빈 문자열로 바꿈
netflix_overall['description'] = netflix_overall['description'].fillna('')

# 데이터를 fitting 하고 transforming 하여 TF-IDF matrix 구성
tfidf_matrix = tfidf.fit_transform(netflix_overall['description'])

# tfidf_matrix shape
tfidf_matrix.shape # (7787, 17905)
```

Description 기반

- TF-IDF(Term Frequency-Inverse Document Frequency)

특정 문서 내에 특정 단어가 얼마나 자주 등장하는 지를 의미하는 단어 빈도(TF)와 전체 문서에서 특정 단어가 얼마나 자주 등장하는지를 의미하는 역문서 빈도(DF)를 통해 "다른 문서에는 등장하지 않지만 특정 문서에서만 자주 등장하는 단어"를 찾아, 문서 내 단어의 가중치를 계산하는 방법
→ 문서의 핵심어 추출, 문서들 사이의 유사도 계산, 검색 결과의 중요도 결정에 활용 등

단점: 메모리 문제

Recommendation System

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

코사인 유사도 함수

- 유사도 함수는 자신과 가장 비슷한 콘텐츠를 찾아줄 때 사용됨
- 벡터의 크기가 중요하지 않은 경우에 거리를 특정하기 위한 메트릭으로 사용
- 즉, 문서 내에서 단어가 얼마나 나왔는지 비율을 확인함

단점: 벡터의 크기가 중요한 경우에 대해서는 잘 작동하지 않음

```python
# Import linear_kernel
from sklearn.metrics.pairwise import linear_kernel

# 코사인 유사도 행렬 계산
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

indices = pd.Series(netflix_overall.index, index=netflix_overall['title']).drop_duplicates()

def get_recommendations(title, cosine_sim=cosine_sim):
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return netflix_overall['title'].iloc[movie_indices]
```

```
[ ] get_recommendations('Peaky Blinders')

4692                    Our Godfather
4358                    My Stupid Boss
1807                               Don
6344                          The Fear
3219    Jonathan Strange & Mr Norrell
4953                Power Rangers Zeo
6783                        The Prison
6950                        The Tudors
6236                    The Con Is On
6585    The Legend of Michael Mishra
Name: title, dtype: object
```

Recommendation System

Content based filtering on multiple metrics

Content based filtering on the following factors:

- Title
- Cast
- Director
- Listed in
- Description

요소를 더 추가하여 진행

```
# count vectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(filledna['soup'])

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

CountVectorizer

- 각 문서에서 해당 단어가 나타나는 횟수를 Count 하여 피처로 사용

```
filledna=filledna.reset_index()
indices = pd.Series(filledna.index, index=filledna['title'])

def get_recommendations_new(title, cosine_sim=cosine_sim):
    title=title.replace(' ','').lower()
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return netflix_overall['title'].iloc[movie_indices]
```

```
[ ] get_recommendations_new('Peaky Blinders', cosine_sim2)

2419                                  Giri / Haji
6374                The Frankenstein Chronicles
6693                     The Murder Detectives
3692                                    Loaded
3412                                Kiss Me First
2616                                Happy Valley
2381                                     Get Even
2845    How to Live Mortgage Free with Sarah Beeny
2886                                I AM A KILLER
3013                      Inside the Criminal Mind
Name: title, dtype: object
```