TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN CUỐI KỲ

HỌC PHẦN: KHOA HỌC DỮ LIỆU

TÊN ĐỀ TÀI

DỰ ĐOÁN ĐÁNH GIÁ PHIM

HỌ VÀ TÊN SINH VIÊN	LỚP HỌC PHẦN	ĐIỂM BẢO VỆ
Nguyễn Tri An	19N13	
Đặng Công Toàn	19N13	
Nguyễn Đức Nghĩa	19N13	

ĐÀ NẴNG, 06/2022

TÓM TẮT

Bài toán đặt ra: Thực hiện thu thập dữ liệu về các bộ phim và dự đoán đánh giá của một bộ phim trên thang 1 đến 10 của bộ phim đó.

Phương pháp giải quyết vấn đề:

- Crawl dữ liệu các phim và trích xuất các dữ liệu cần thiết để có thể dự đoán bộ phim có điểm số bao nhiêu.
- Minh họa dữ liệu để xem các đặt trưng của dữ liệu như thế nào để phục vụ công việc tiếp theo.
- Phân tích các đặc trưng dựa EDA và thực hiện các Feature Engineering với các hành động như bổ sung đặc trưng bị trống, biển đổi (Label hóa) dữ liệu về dạng số để thực hiện bài toán dự đoán.
- Dự đoán điểm bộ phim dựa trên các đặc trưng mà bộ phim đó có.

Kết quả đạt được: Điểm đánh giá một bộ phim và các độ lệch trong việc dự đoán dựa trên các metrics từ đó rút ra được nhận xét về mô hình dự đoán điểm trên dữ liệu phim.

BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên thực hiện	Các nhiệm vụ	Tự đánh giá theo 3 mức (Đã hoàn thành/Chưa hoàn thành/Không triển khai)
Đặng Công Toàn	- Crawl dữ liệu - EDA	Đã hoàn thành
Nguyễn Tri An	-Tiền xử lý dữ liệu -Linear Regression	Đã hoàn thành
Nguyễn Đức Nghĩa	-K Nearest Neighbor -Bagging Regression	Đã hoàn thành

MỤC LỤC

DANH MỤC HÌNH ẢNH

1. Giới thiệu

Vấn đề bài toán đặt ra:

- Thực hiện viêc Crawl dữ liệu từ một nguồn trang web trên mạng (cụ thể ở đây là dữ liệu về các bộ phim).
- Sau đó thực hiện việc xử lí và biến đổi trực quan hóa dữ liệu để có thể tìm ra các quy luật tìm ẩn có trong dữ liệu mà ta đã trực tiếp crawl về.
- Thực hiển các bước Feature Engineering cho dữ liệu phim mà ta có được.
- Cuối cùng là dùng dữ liệu để thực hiện mục tiêu bài toán đặc ra đó là dự đoán đánh giá bộ phim (cụ thể là điểm đánh giá theo thang 0 tới 10) dựa trên hai mô hình/ thuật toán để xem hiểu năng dự đoán ra sao và các đặc trưng của dữ liệu có hỗ trợ tốt cho bài toán dự đoán hay không.

Giải pháp tổng quan cần thiết:

- Đối việc Crawl dữ liệu ta sử dụng các giải pháp để crawl dữ liệu phổ biến ngày nay cụ thể như beautifulsoup, API với Postman, Scrapy, … Trong bài toán crawl phim nhóm sử dụng beautifulsoup là công cụ chính để thu thập dữ liệu.

Hình 1: Web Scraping with BeautifulSoup and Python

- Minh họa EDA được minh họa thông qua các xử lí dữ liệu ở mức cơ bản và trực quan hóa dữ liệu đó thông qua các công cụ như seaborn hay matplotlib.

Hình 2: Công cụ trực quan hóa dữ liệu Seaborn.

- Bước feature engineering để tạo thêm đặc trưng mới, biến đổi các đặc trưng dạng chữ (Tag phim, Thể loại phim, Studio sản xuất phim, giới hạn độ tuổi, …) và thực hiện xử lí thêm dữ liệu trống vào các cột thiếu trong đặc trưng đó bằng một trong các hình thức như random hoặc thêm theo một quy luật mình nhận ra trong qua trình minh họa EDA.
- Thực hiện sử dụng các công cụ trong bộ thư viện sklearn cung cấp cho ta nhiều API để giải quyết các bài toán về học máy, AI do đó đối với bài toán dự đoán thang phim từ 0 tới 10 các công cụ Regression thuộc bộ công cụ trên hỗ trợ tốt cho bài toán dự đoán (Cụ thể Linear Regression, KNN Regression, Bagging Regression).

Hình 3: Công cụ giải quyết bài toán dự đoán sklearn

2. Thu thập và mô tả dữ liệu

2.1. Thu thập dữ liệu

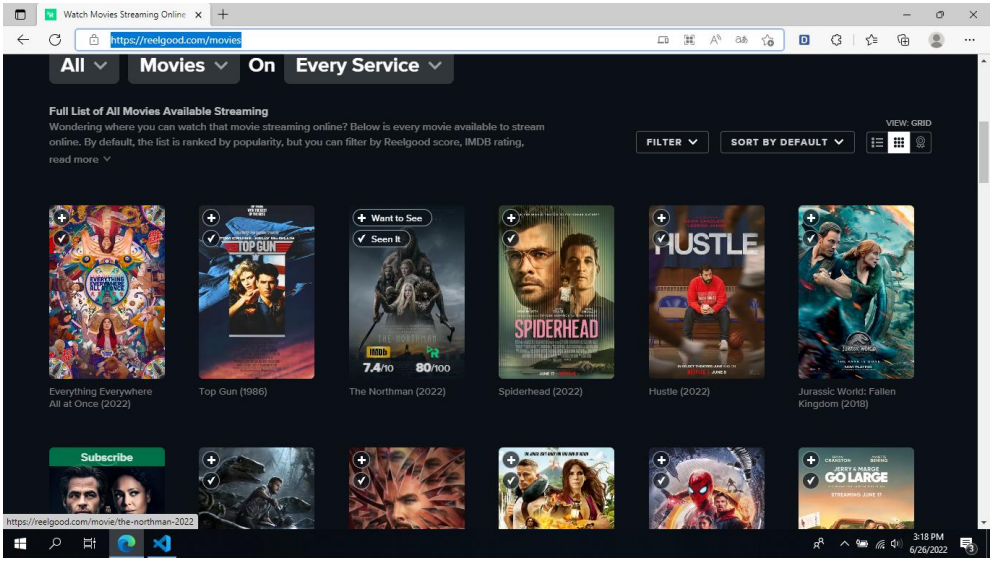
Nguồn dữ liệu: Watch Movies Streaming Online - Top 50 | Reelgood

Công cụ thu thập:

- BeautifulSoup (sử dụng để trích xuất dữ liệu trên trang web).
- Requests (sử dụng để liên kết đến trang web cần thu thập dữ liệu).

Các bước thu thập dữ liệu:

- Đầu tiên chúng ta sẽ tiến hành lấy link của các bộ phim. Chúng ta sẽ không sử dụng trang này để lấy dữ liệu vì nó chỉ có tên phim và điểm rating, thiếu rất nhiều thông tin cần thiết khác



Hình 4 : Trang web thực hiện crawl dữ liệu

```
#link of films in reelgood.com
for i in range(0, 5500, 50):
    URL= "https://reelgood.com/movies?offset=" + str(i)
    page = requests.get(URL)
    soup = BeautifulSoup(page.content, "html.parser")

    f = open("/kaggle/working/movies_netflix.txt", "a")
    for link in soup.select('[itemprop=itemListElement] [itemprop=url]'):
        data = link.get('content')
        f.write(data)
        f.write("\n")
```

Hình 5: Lấy các link đến trang detail của các bộ phim

Đối với trang reelgood thì nó tiến hành phân trang (mỗi trang chứa 50 phim) và được đánh dấu bằng tham số offset. Do đó chúng ta sẽ lặp từ 0 – 5500 với mục đích lấy được đường link đến trang detail của 5500 bộ phim.

– Tiếp đến chúng ta sẽ tiếp tục tạo định dạng mẫu cho mỗi bộ phim:

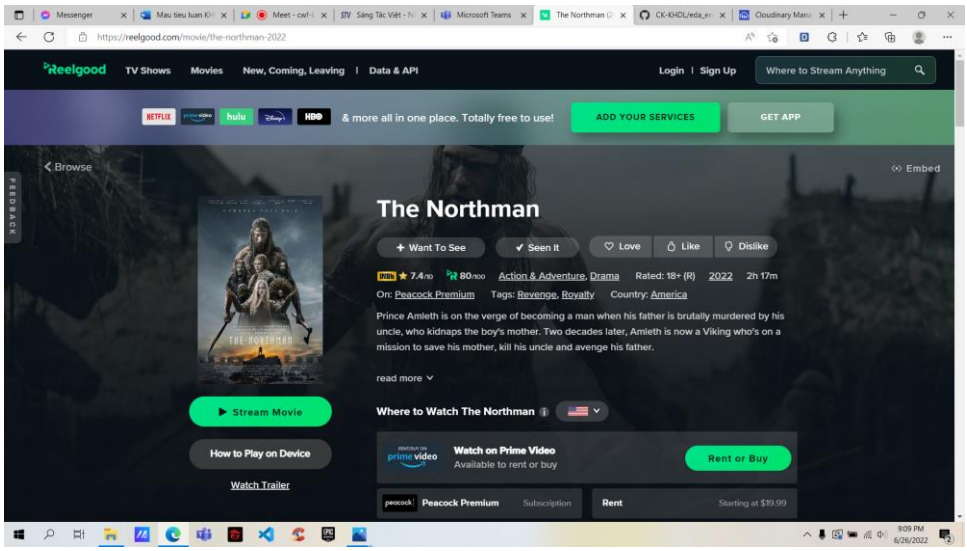
```
filmObj = {
      "Title": None,
      "IMDB_Rating": None,
      "Reelgood_Rating": None,
      "Genres": [],
      "Year": None,
      "Tags": [],
      "Country": None,
      "Source": None,
      "Rated": None,
      "Duration": None,
      "Description": None,
      "Director": None,
      "Employees": [],
}
```

Hình 6: Định dạng dữ liệu của từng bộ phim

– Tiến hành lấy dữ liệu bằng cách request tới link của các bộ phim trong tập chúng ta vừa lấy được lúc nãy

```
with open('/kaggle/input/link-films-crawl/movies_netflix.txt') as f:
    links = [line.rstrip() for line in f]
```

– Với mỗi đường link của phim, chúng ta sẽ sử dụng thư viện requests để lấy nội dung của trang đó và dùng BeautifulSoup để trích xuất dữ liệu



Hình 7: Trang detail của từng phim

Đối với reelgood thì các thông tin mà chúng ta cần sẽ được gói trong các tag html và được đánh class cho mỗi tag đó. Ở đây có một số class mà chúng ta sẽ sử dụng để trích xuất dữ liệu:

- Title: **css-of585f e14injhv7**
- IMDB Rating: **css-xmin1q ey4ir3j3**
- Các loại dữ liệu khác (thể loại, độ tuổi, năm, thời lượng,…) được gói trong thẻ **a** và phân biệt với nhau bởi các url

  Ví dụ: năm - /year/2022

  thể loại - /movies/genre/action-and-adventure

  …

Chúng ta chỉ cần dựa vào từ khoá thì sẽ lấy được các loại thông tin mà chúng ta cần như year, genre, source...

```
arr_film = []
for i in range(0, len(links)):
  link = links[i]
  film = copy.deepcopy(filmObj)

  try:
    with requests.get(link) as page:
      soup = BeautifulSoup(page.content, 'html.parser')
      film['Title'] = soup.find(class_="css-of585f e14injhv7")
      if film['Title'] != None:
        film['Title'] = film['Title'].get_text()

      div_info = soup.find(class_="css-1su90nd ey4ir3j0")

      # GET SCORE
      film['IMDB_Rating'] = div_info.find(class_="css-xmin1q ey4ir3j3").get_text()
      film['Reelgood_Rating'] = div_info.find(class_="css-xmin1q ey4ir3j8")
      if film['Reelgood_Rating'] != None:
        film['Reelgood_Rating'] = film['Reelgood_Rating'].get_text()

      # GET INFO BY TAG A
      list_info = div_info.find_all(class_="css-10wrqt0")
      for info in list_info:
        type_info = info.get("href")
        if('/genre' in type_info):
          film['Genres'].append(info.get_text())
        elif '/year' in type_info:
          film['Year'] = info.get_text()
        elif '/list' in type_info:
          film['Tags'].append(info.get_text())
        elif '/origin' in type_info:
          film['Country'] = info.get_text()
        elif '/source' in type_info:
          film['Source'] = info.get_text()
```

```
      # GET INFO BY TAG META
      film['Rated'] = div_info.find('meta', itemprop='contentRating')
      if film['Rated'] != None:
        film['Rated'] = film['Rated'].get("content", None)
      film['Duration'] = div_info.find('meta', itemprop='duration')
      if film['Duration'] != None:
        film['Duration'] = film['Duration'].get("content", None)

      # GET DESCRIPTION
      film['Description'] = soup.find('p', itemprop="description").get_text()

      # GET CAST & CREW
      list_employee = soup.find_all('h3', itemprop="name", class_="css-11yr18h egg5eqo2")
      # GET DIRECTOR
      if len(list_employee) > 0:
        film['Director'] = list_employee[0].get_text()
        #GET EMPLOYEE
        for i in range(1, len(list_employee)):
          employee = list_employee[i]
          film['Employees'].append(employee.get_text())

      print(link, " done")
      arr_film.append(film)

  except (NameError, AttributeError) as e:
    print(e)
    continue
f.close()
```

Hình 8: Code thực hiện quá trình trích xuất dữ liệu

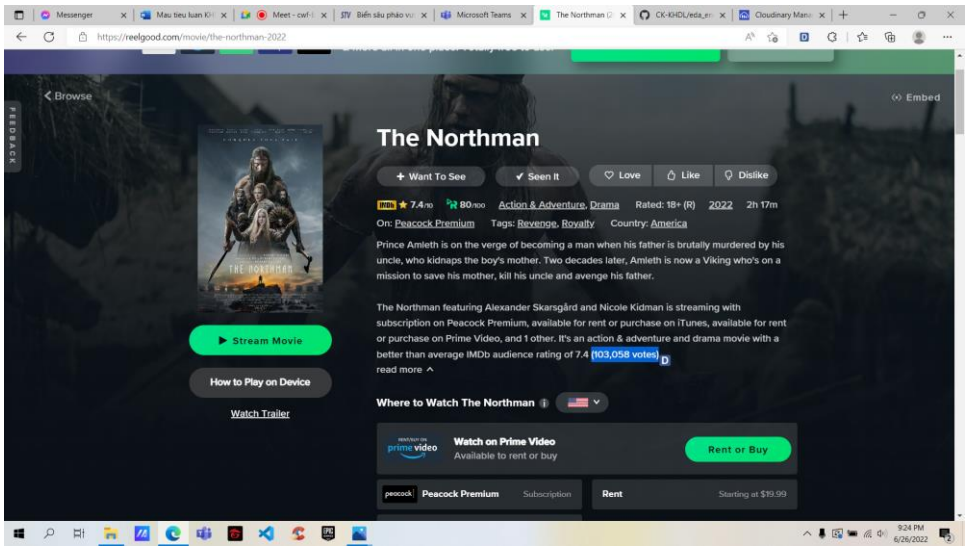
– Tiến hành lưu dữ liệu dưới dạng file csv

```
df = pd.DataFrame(arr_film)
df.to_csv('film_reelgood.csv',index=False, encoding="utf-8")
```

Lưu ý:

Vì một số lý do nào đó mà đôi khi trang web reelgood sẽ load rất lâu làm cho thư viện requests bị timeout và ngắt quá trình thu thập dữ liệu. Nên chúng ta sẽ phải thực hiện lặp đi lặp lại nhiều lần bước trên và chỉ cần thay đổi chỉ số index đến bộ phim được crawl gần nhất.

– Đến đây gần như đã hoàn thành bước crawl dữ liệu tuy nhiên nếu chú ý kĩ hơn thì chúng ta sẽ nhận ra ở cuối phần description của mỗi bộ phim thường sẽ có thông tin số người đánh giá



Hình 9: Description sẽ chứa thông tin số lượng người vote

Có thể thông tin này sẽ hữu ích trong quá trình model hoá để dự đoán rating của phim nên chúng ta cũng sẽ tiến hành trích xuất dữ liệu này.

Kết quả: chúng ta thu được 4455 mẫu dữ liệu với 13 đặc trưng

```
RangeIndex: 4455 entries, 0 to 4454
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   4455 non-null   int64
 1   Title        4455 non-null   object
 2   IMDB_Rating  4455 non-null   float64
 3   Genres       4455 non-null   object
 4   Year         4455 non-null   int64
 5   Tags         4455 non-null   object
 6   Country      2825 non-null   object
 7   Source       2803 non-null   object
 8   Rated        1856 non-null   object
 9   Duration     4324 non-null   object
 10  Description  4455 non-null   object
 11  Director     3548 non-null   object
 12  Employees    4455 non-null   object
 13  User_Votes   4396 non-null   object
dtypes: float64(1), int64(2), object(11)
memory usage: 487.4+ KB
```

```
df.head(3)
```

	Unnamed: 0	Title	IMDB_Rating	Genres	Year	Tags	Country	Source	Rated	Duration	Description	Director	Employees	User_Votes
0	0	The Lost City	6.2	['Romance', 'Action & Adventure']	2022	[]	America	Epix	13+ (PG-13)	PT1H52M	Follows a reclusive romance novelist who was s...	Aaron Nee	['Adam Nee', 'Sandra Bullock', 'Channing Tatum...	48,599 votes
1	1	Chip 'n Dale: Rescue Rangers	7.2	['Action & Adventure', 'Animation']	2022	[]	NaN	Disney+	7+ (PG)	PT1H39M	Thirty years after their popular television sh...	NaN	[]	13,451 votes
2	2	Uncharted	6.4	['Drama', 'Action & Adventure']	2022	['Video Games']	America	NaN	13+ (PG-13)	PT1H56M	A young street-smart, Nathan Drake and his wis...	Ruben Fleischer	['Tom Holland', 'Mark Wahlberg', 'Antonio Band...	118,363 votes

Hình 10: Kết quả sau khi thu thập dữ liệu

2.2. Mô tả dữ liệu

Mẫu dữ liệu của chúng ta gồm 4455 mẫu với 13 đặc trưng, tuy nhiên chúng ta cần tiến hành xử lý đơn giản để phục vụ cho các bước EDA sau này:

```
mlb = MultiLabelBinarizer()
# Change format data
df["Genres"] = df["Genres"].apply(lambda x: x.replace("'", "") if "'" in x else x)
df["Tags"] = df["Tags"].apply(lambda x: x.replace("'", "") if "'" in x else x)
df["Employees"] = df["Employees"].apply(lambda x: x.replace("'", "") if "'" in x else x)

df["Genres"] = df["Genres"].apply(lambda x: x.replace("[", "") if "[" in x else x)
df["Tags"] = df["Tags"].apply(lambda x: x.replace("[", "") if "[" in x else x)
df["Employees"] = df["Employees"].apply(lambda x: x.replace("[", "") if "[" in x else x)

df["Genres"] = df["Genres"].apply(lambda x: x.replace("]", "") if "]" in x else x)
df["Tags"] = df["Tags"].apply(lambda x: x.replace("]", "") if "]" in x else x)
df["Employees"] = df["Employees"].apply(lambda x: x.replace("]", "") if "]" in x else x)

df["Genres"] = df["Genres"].apply(lambda x: x + 'empty' if not x else x)
df["Tags"] = df["Tags"].apply(lambda x: x + 'empty' if not x else x)
df["Employees"] = df["Employees"].apply(lambda x: x + 'empty' if not x else x)

# explode genres
df = df.join(
            pd.DataFrame(
                mlb.fit_transform(df['Genres'].str.split(', ')),
                index=df.index,
                columns="Genre_" + mlb.classes_))
```

```
# convert user votes to number
df["User_Votes"] = df["User_Votes"].astype(str)
df['User_Votes'] = df['User_Votes'].apply(lambda x: int(x.replace(",", "").split(" ")[0].strip()) if 'nan' not in x else 0)

# convert duration to number
df["Duration"] = df["Duration"].astype(str)
def convert_to_minutes(str_duration):
    if str_duration != 'nan':
        str_duration = str_duration[2:].replace("M", "")
        if "H" in str_duration:
            arr_time = str_duration.split("H")
            return int(arr_time[0]) * 60 + int(arr_time[1])
        else:
            return int(str_duration)
    else: return 0
df['Duration'] = df['Duration'].apply(lambda x: convert_to_minutes(x))
```

Hình 11: Thực hiện xử lý dữ liệu đơn giản

Cụ thể thì chúng ta sẽ tiến hành format lại các dữ liệu như Duration, User_Votes từ String về dạng Int, tiến hành label đặc trưng Genre để thực hiện các bước EDA phía dưới.

```
RangeIndex: 4455 entries, 0 to 4454
Data columns (total 37 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                4455 non-null   int64
 1   Title                     4455 non-null   object
 2   IMDB_Rating               4455 non-null   float64
 3   Genres                    4455 non-null   object
 4   Year                      4455 non-null   int64
 5   Tags                      4455 non-null   object
 6   Country                   2825 non-null   object
 7   Source                    2803 non-null   object
 8   Rated                     1856 non-null   object
 9   Duration                  4455 non-null   int64
 10  Description               4455 non-null   object
 11  Director                  3548 non-null   object
 12  Employees                 4455 non-null   object
 13  User_Votes                4455 non-null   int64
 14  Genre_Action & Adventure  4455 non-null   int32
 15  Genre_Animation           4455 non-null   int32
 16  Genre_Anime               4455 non-null   int32
 17  Genre_Biography           4455 non-null   int32
 18  Genre_Children            4455 non-null   int32
 19  Genre_Comedy              4455 non-null   int32
 20  Genre_Crime               4455 non-null   int32
 21  Genre_Cult                4455 non-null   int32
 22  Genre_Documentary         4455 non-null   int32
 23  Genre_Drama               4455 non-null   int32
 24  Genre_Family              4455 non-null   int32
 25  Genre_Fantasy             4455 non-null   int32
 26  Genre_History             4455 non-null   int32
 27  Genre_Horror              4455 non-null   int32
 28  Genre_LGBTQ               4455 non-null   int32
 29  Genre_Musical             4455 non-null   int32
 30  Genre_Mystery             4455 non-null   int32
 31  Genre_Romance             4455 non-null   int32
 32  Genre_Science-Fiction     4455 non-null   int32
 33  Genre_Sport               4455 non-null   int32
 34  Genre_Stand-up & Talk     4455 non-null   int32
 35  Genre_Thriller            4455 non-null   int32
 36  Genre_empty               4455 non-null   int32
dtypes: float64(1), int32(23), int64(4), object(9)
```

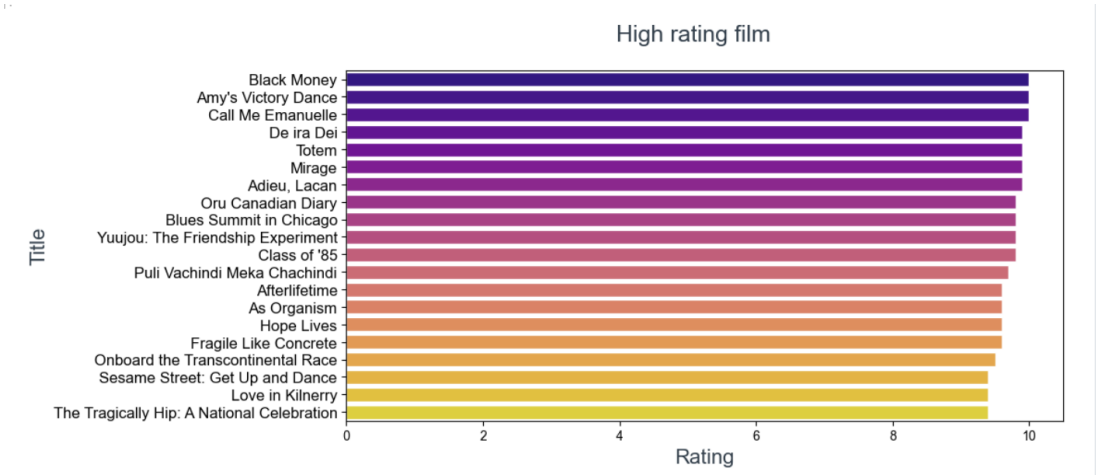
[6]:

	Unnamed: 0	Title	IMDB_Rating	Genres	Year	Tags	Country	Source	Rated	Duration	...	Genre_Horror	Genre_LGBTQ	Genre_Musical	Genre_Mystery	Ge
0	0	The Lost City	6.2	Romance, Action & Adventure	2022	empty	America	Epix	13+ (PG-13)	112	...	0	0	0	0	
1	1	Chip 'n Dale: Rescue Rangers	7.2	Action & Adventure, Animation	2022	empty	NaN	Disney+	7+ (PG)	99	...	0	0	0	0	
2	2	Uncharted	6.4	Drama, Action & Adventure	2022	Video Games	America	NaN	13+ (PG-13)	116	...	0	0	0	0	
3	3	Spider-Man: No Way Home	8.3	Comedy, Drama	2021	Teen, Superhero, Supernatural	America	NaN	13+ (PG-13)	148	...	0	0	0	0	
4	4	Top Gun	6.9	Romance, Action & Adventure	1986	Military, Feel-good	America	Paramount+	13+ (PG-13)	110	...	0	0	0	0	

Hình 12: Kết quả sau khi format lại dữ liệu

Mô tả dữ liệu:

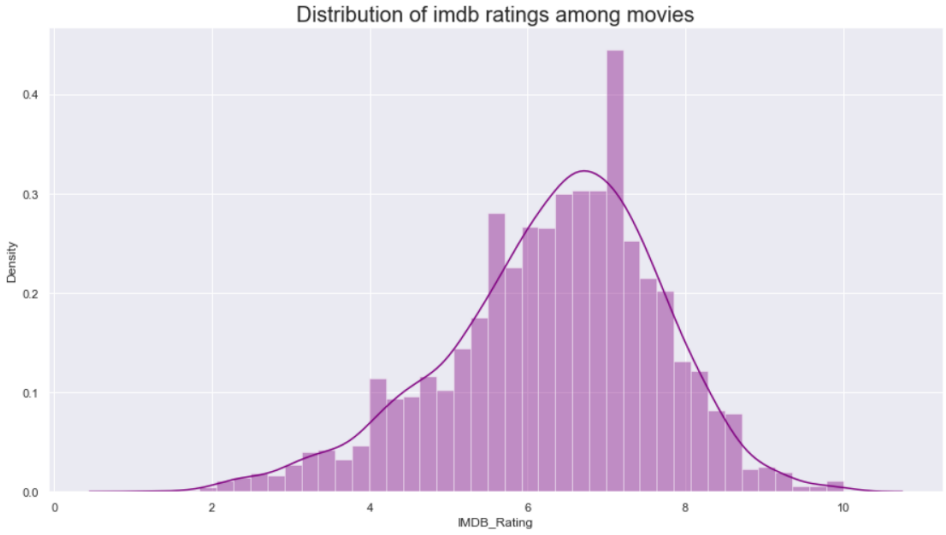
- **Top 20 bộ phim được đánh giá cao nhất**



Hình 13: Top 20 bộ phim được đánh giá cao nhất

- Ta có thể thấy được top 20 film thì điểm rating cách không quá lớn (đa số chạy từ 9-10). Do đó khi xử lý dữ liệu thì ta có thể không cần phải xử lý ngoại lệ đối với đặc trưng này

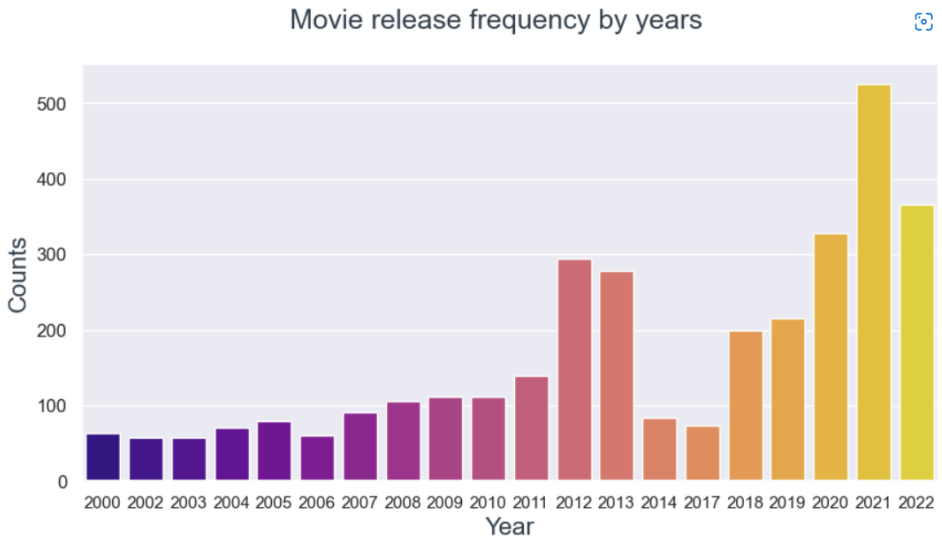
- **Phần lớn các bộ phim được đánh giá như thế nào?**



Hình 14: Phân bố của đánh giá phim

- Điểm rating của các film có phân bố gần với phân bố chuẩn, điểm rating mà các film nhận được nhiều nhất là khoảng 7.

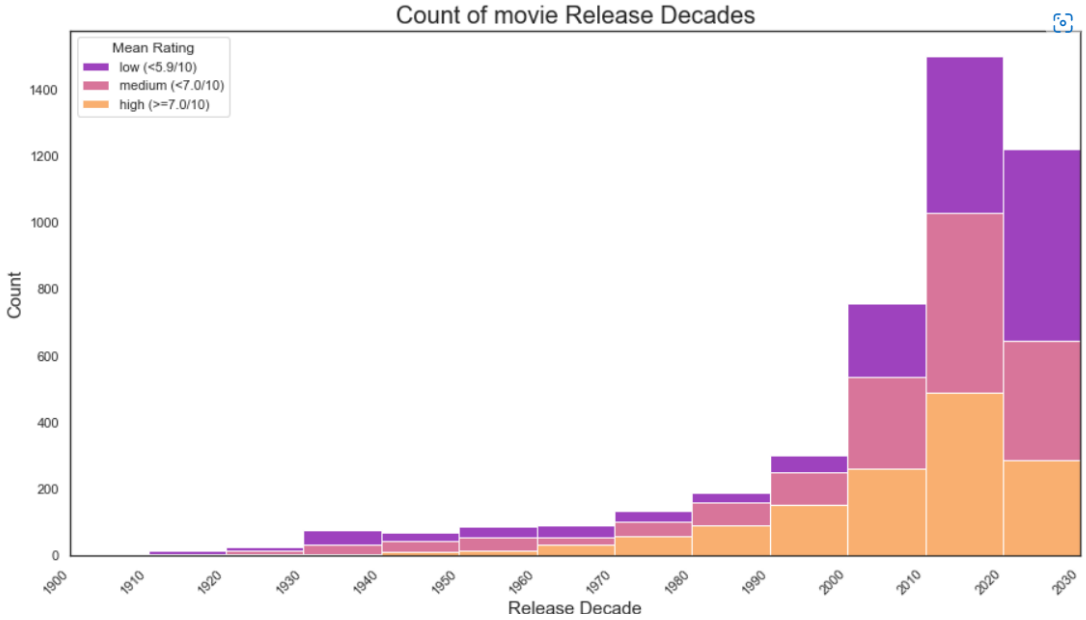
- **Tần suất phát hành phim qua các năm**



Hình 15: Tần suất phát hành phim

- Tần suất ra phim không tăng tuyến tính theo từng năm mà có sự biến động mạnh (giảm) vào năm 2014-2017
- Năm 2021 là năm có tần suất ra phim cao nhất

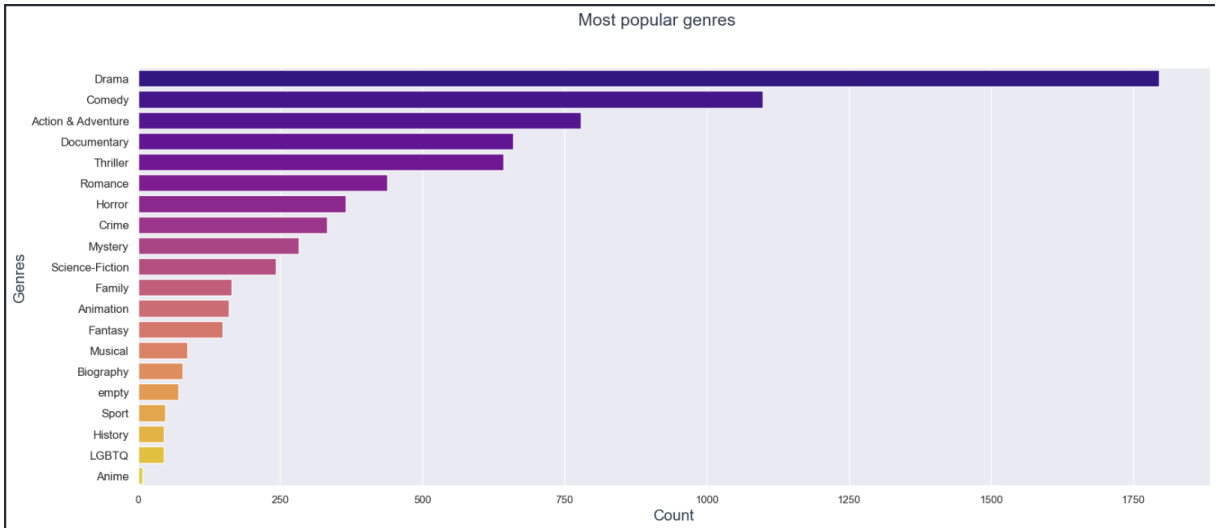
- **Số lượng phim ra mắt cũng như chất lượng trong các thập kỉ qua**



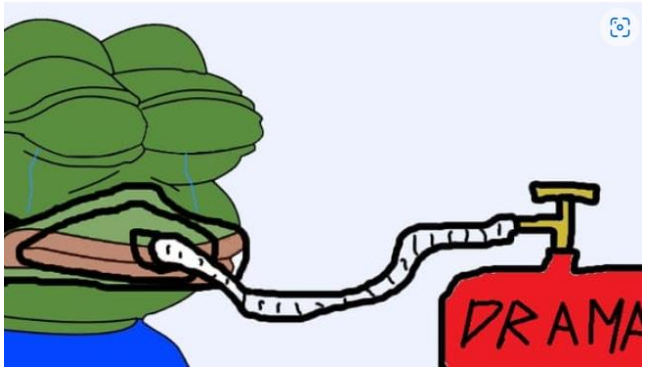
Hình 16: Số lượng và chất lượng phim qua các thập kỉ

- Thập kỉ 2010 có thể được xem là thập kỉ có chất lượng phim tốt nhất và số lượng phim nhiều nhất

- **Thể loại phổ biến nhất**

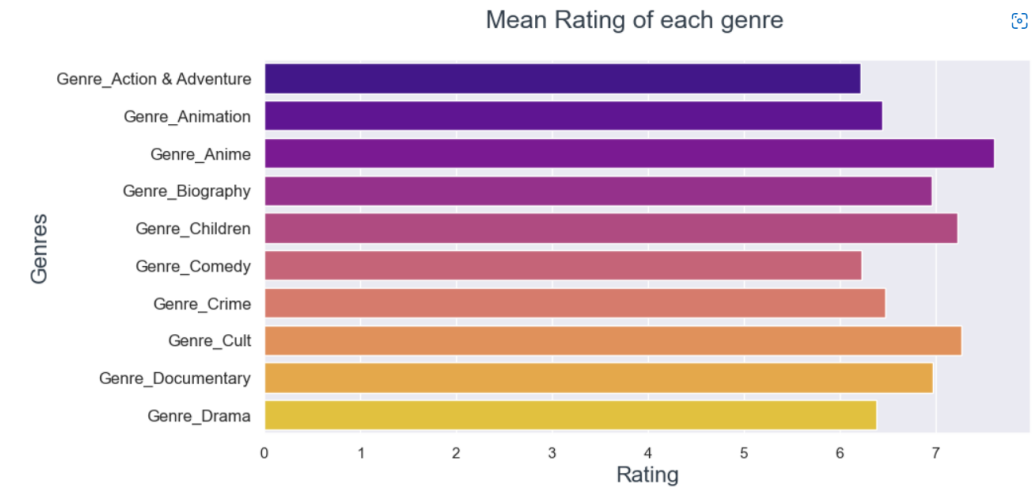


Hình 17: Các thể loại phim phổ biến nhất



- Có thể các nhà làm phim muốn đánh vào thị hiếu số đông (thể loại này dễ gây thích thú, tò mò, tạo hiệu ứng số đông), giúp tăng doanh số của phim

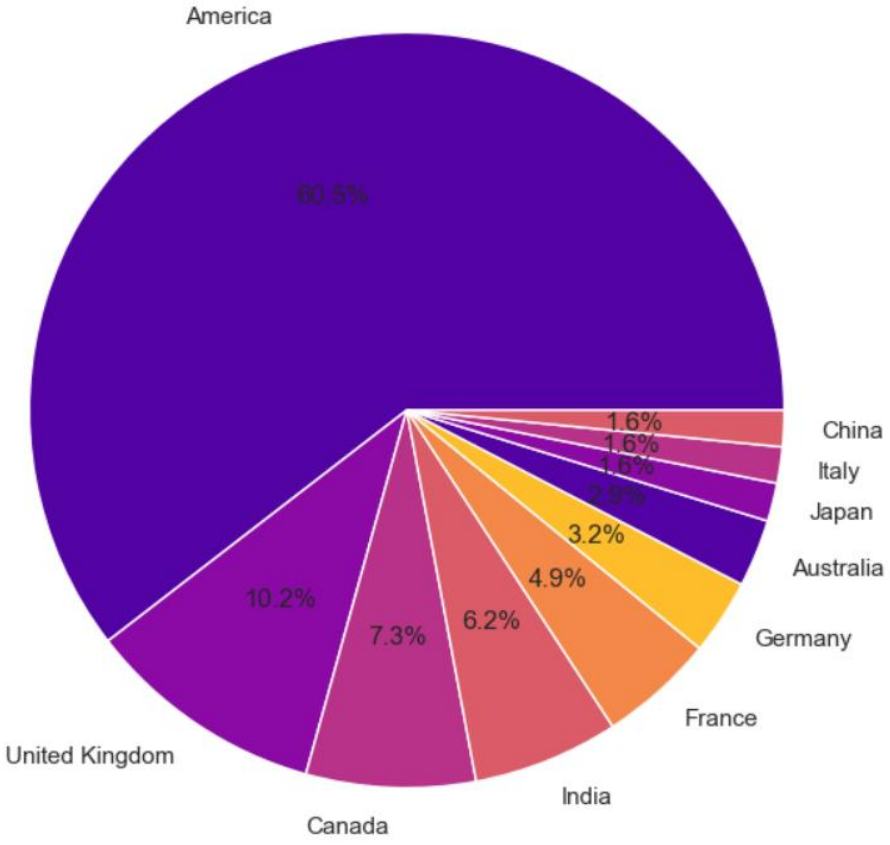
- **Rating trung bình của các thể loại**



Hình 18: Điểm rating trung bình của các thể loại phim

- Các thể loại dành cho độ tuổi thiếu nhi thì có độ rating cao hơn các thể loại khác. Thể loại phim văn hoá, tôn giáo và phim tài liệu cũng được cộng đồng đón nhận mạnh mẽ (số lượng phim thuộc thể loại anime rất ít nên điểm trung bình nó sẽ cao hơn các thể loại khác)

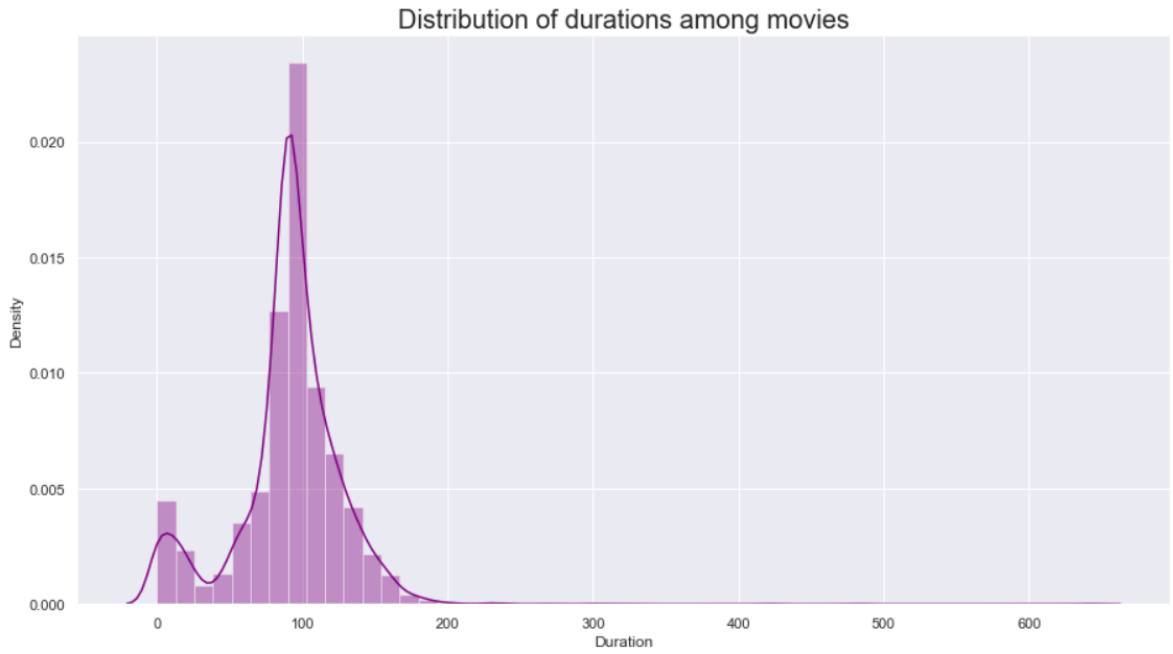
- **Những nước sản xuất phim nhiều nhất**



Hình 19: Các nước sản xuất phim nhiều nhất

- Qua đây ta có thể thấy được nền công nghiệp điện ảnh ở Mỹ phát triển rất mạnh mẽ.
- Và đất nước chuyên sản xuất những bộ phim phi vật lý là Ấn độ lại đứng vị trí thứ 5 trong những quốc gia sản xuất nhiều phim nhất

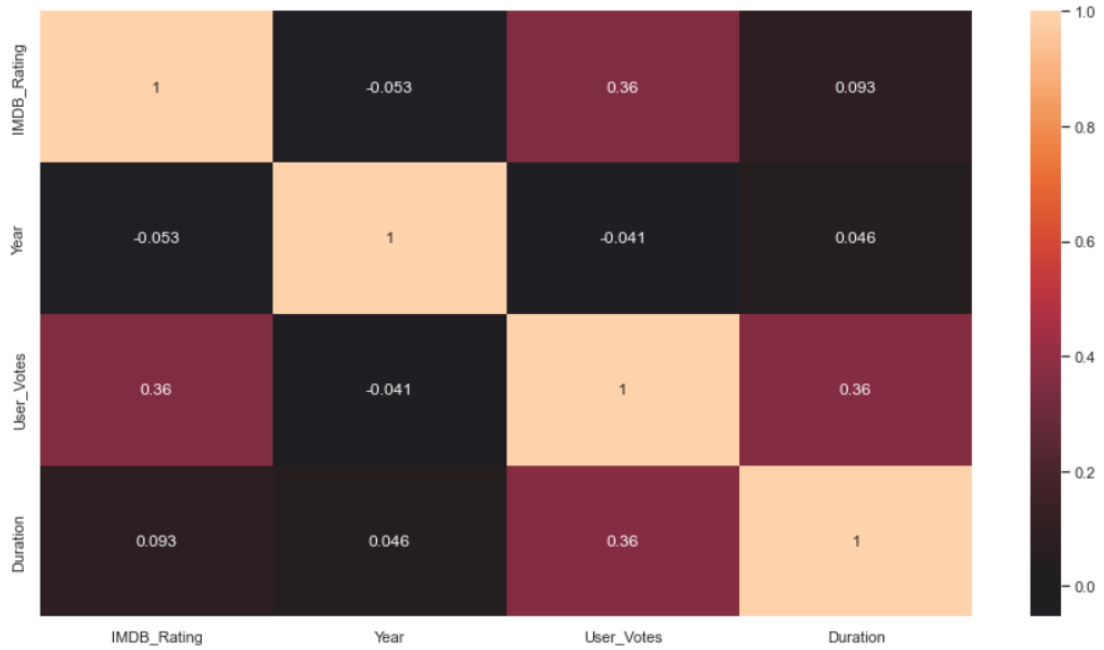
- **Thời lượng các bộ phim thường là bao nhiêu**



Hình 20: Phân bố của thời lượng phim

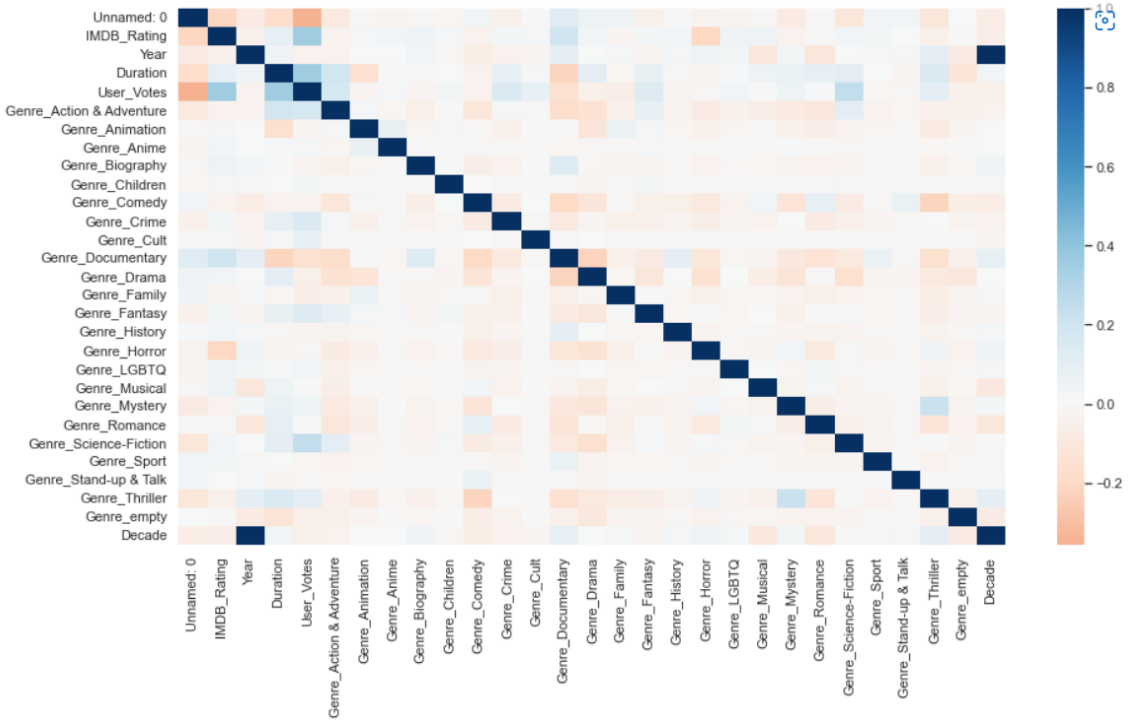
- Thời lượng của các bộ phim đa số rơi vào khoảng 1 tiếng rưỡi

- **Sự tương quan giữa IMDB Rating với các đặc trưng khác**



Hình 21: Sự tương quan giữa IMDB Rating với các đặc trưng khác

- Dataset có độ tương quan thấp, IMDB tương quan cao nhất với đặc trưng User Votes (0.36)



Hình 22: Sự tương quan giữa IMDB Rating với các thể loại phim

- Ta có thể thấy được trong dataset này thì IMDB Rating có độ tương quan thấp đối với các đặc trưng khác
- User Votes và thể loại Documentary thì có độ tương quan với IMDB Rating cao hơn so với các đặc trưng khác. Ta có thể sử dụng 2 đặc trưng này để dự đoán số điểm rating của 1 bộ phim

3. Trích xuất đặc trưng

3.1 Làm sạch dữ liệu

- Bộ dữ liệu gồm có 13 đặc trưng .

- Số lượng đặc trưng dữ liệu số học là : 2, bao gồm: IMDB_Rating và Year

- Số lượng đặc trưng dữ liệu phân loại là: 11 , bao gồm : Title , Genres, Tags, Country, Source, Rated, Duration, Description, Director, Employees, User_Votes.

- Dữ liệu bị trống:

	Missing_Number	Missing_Percent
Rated	2599	58.338945
Source	1652	37.081930
Country	1630	36.588103
Director	907	20.359147
Duration	131	2.940516
User_Votes	59	1.324355
Employees	0	0.000000
Description	0	0.000000
Tags	0	0.000000
Year	0	0.000000
Genres	0	0.000000
IMDB_Rating	0	0.000000
Title	0	0.000000

Hình23: Các biến có giá trị trống

3.1.1 Xử lý biến Genres và Tags

Genres

Genres
['Romance', 'Action & Adventure']
['Action & Adventure', 'Animation']
['Drama', 'Action & Adventure']
['Comedy', 'Drama']
['Romance', 'Action & Adventure']
['Action & Adventure', 'Drama']
['Drama', 'Comedy']

Hình 24 : Mô tả biến Genres

Tags

Tags
[]
[]
['Video Games']
['Teen', 'Superhero', 'Supernatural']
['Military', 'Feel-good']
['Revenge', 'Royalty']

Hình 25: Mô tả biến Tags

Genres và Tags có kiểu dữ liệu giống nhau , đều là kiểu object chứa nhiều giá trị.

- Sử dụng MultilabelBỉnizer để xử lý.

MultilabelBinarizer là thư viện có sẵn xử lý dữ liệu phân loại có nhiều giá trị , chuyển hóa các giá trị trong hàng thành các cột và chuyển hóa về giá trị 0 và 1.

- Cách xử lý :

```
# Phân tách genres
mlb = MultiLabelBinarizer()
df = df.join(
            pd.DataFrame(
                mlb.fit_transform(df['Genres'].str.split(', ')),
                index=df.index,
                columns="Genre_" + mlb.classes_))
print("Genres Exploded: ",df["Genre_" + mlb.classes_].columns)
# Phân tách tag
df = df.join(
            pd.DataFrame(
                mlb.fit_transform(df.pop('Tags').str.split(', ')),
                index=df.index,
                columns="Tag_" + mlb.classes_))
```

Hình 26: Phân tách Genres và Tags

- Kết quả sau khi xử lý

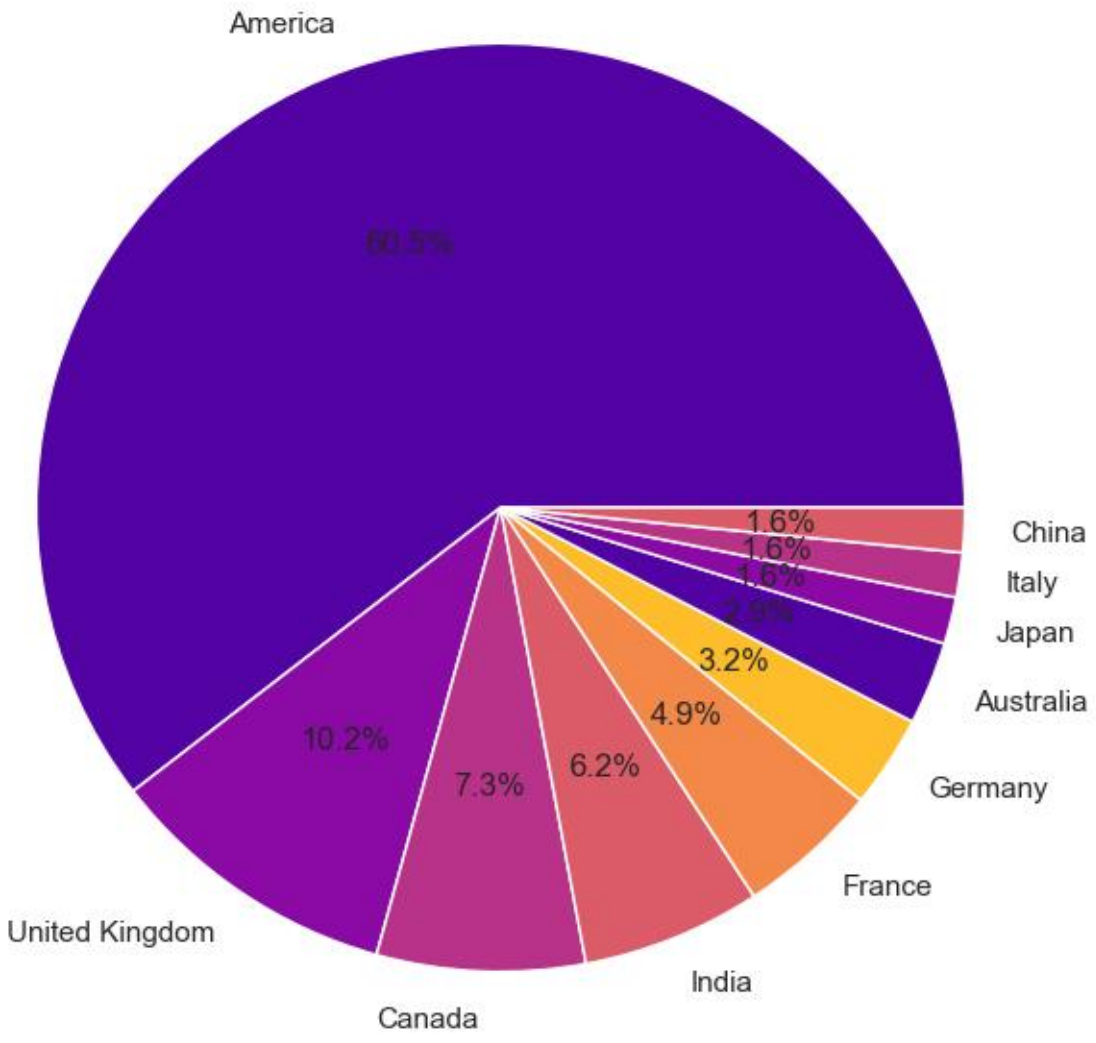
Genre_Action &	Genre_Animatic	Genre_Anime	Genre_Biograpl	Genre_Children	Genre_Comedy
1	0	0	0	0	0
1	1	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	1

Hình 27 :Kết quả sau khi xử lý biến Genres

Tag_Adaptation	Tag_Alien	Tag_Animal	Tag_Apocalypse	Tag_Baseball	Tag_Based On T
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	0
1	0	0	0	0	0

Hình 28: Kết quả sau khi xử lý biến Tags

3.1.2 Xử lý biến Country



Hình 29: EDA Top 10 quốc gia sản xuất phim nhiều nhất

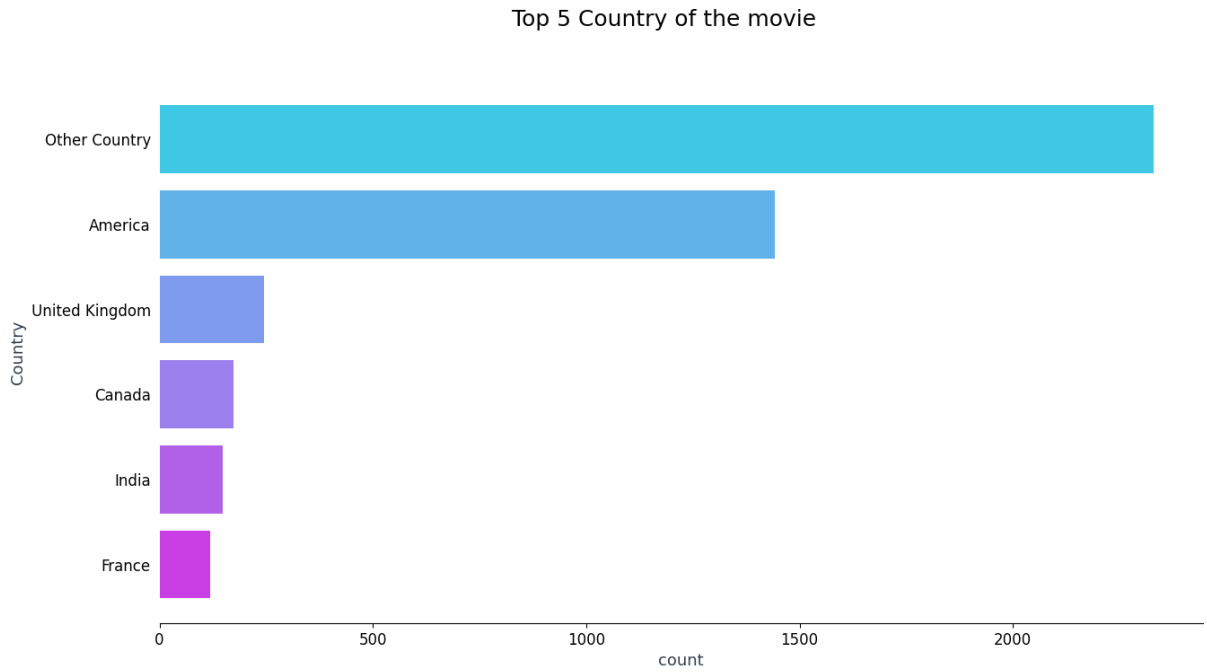
Dựa vào kết quả từ bước EDA, ta sẽ giữ lại 5 quốc gia sản xuất phim nhiều nhất. Các quốc gia còn lại sẽ thuộc nhóm Orther Country.

-Cách thực hiện:

```
# Xử lý biến trống Country
# Nhóm các quốc gia chiếm tỉ lệ thấp nhất -> Other Country
# Giữ lại 5 quốc gia chiếm tỉ lệ cao (America, United Kingdom, Canada, India, France)
for i in range(len(df['Country'])):
    if df.iloc[i,4] != 'America' and df.iloc[i,4] != 'United Kingdom' and df.iloc[i,4] != 'Canada' and df.iloc[i,4]
        df.iloc[i,4] = 'Other Country'
    else:
        df.iloc[i,4] = df.iloc[i,4]
df['Country'].fillna('Other Country', inplace=True) # fillna
print(df['Country'].unique())
```

Hình 30: Xử lý biến Country

-Kết quả sau khi xử lý:



Hình 31: Kết quả sau khi xử lý Country

Nhận xét:

- Sau khi rút gọn còn lại 5 quốc gia phổ biến nhất thì lượng phim của nước Mỹ (America) vẫn chiếm tỷ lệ cao nhất , thấp nhất là Pháp(France).
- Điều này cho thấy Mỹ là quốc gia có nền điện ảnh lớn nhất toàn cầu.

3.1.3 Xử lý biến Source

Giữ lại giá trị Source của top 5 quốc gia đã chọn , các quốc gia còn lại có giá trị Orther Source.

-Cách thực hiện

```
# Xử lý biến trống Source
# Giữ lại Source phim của top 5 quốc gia đã được chọn
for i in range(len(df['Source'])):
    if df.iloc[i,4] != 'America' and df.iloc[i,4] != 'United Kingdom' and df.iloc[i,4] != 'Canada' and df.iloc[i,4] != 'India' and df.iloc[i,4] != 'France':
        df.iloc[i,5] = 'Other Source'
    else:
        df.iloc[i,5] = df.iloc[i,5]
df['Source'].fillna('Other Source', inplace=True) # fillna
```

Hình 32: Xử lý biến Source

3.1.4 Xử lý biến trống Rated

Chuyển đổi các ký hiệu độ tuổi thành tên phù hợp

Giá trị ban đầu:

Rated
13+ (PG-13)
7+ (PG)
13+ (PG-13)
13+ (PG-13)
13+ (PG-13)
18+ (R)
18+ (R)
13+ (PG-13)
7+ (PG)

Hình 33: Mô tả biến Rated

-Cách thực hiện

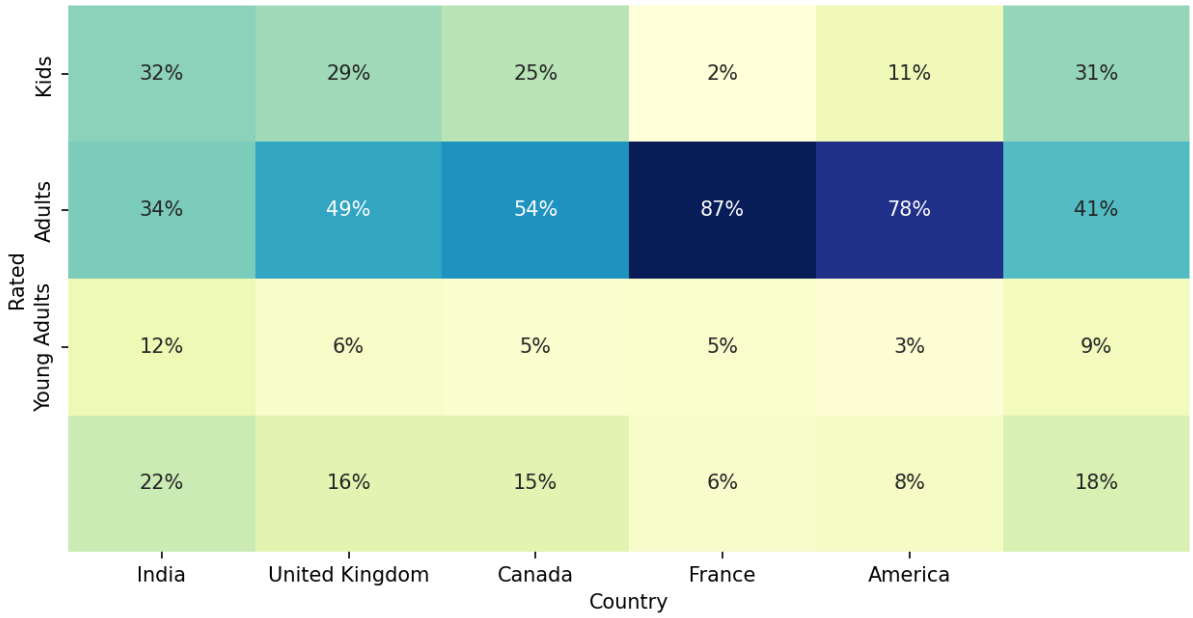
```
# Xử lý biến trống Rated
for i in range(len(df['Rated'])):
    if df.iloc[i,6] == '18+ (R)':
        df.iloc[i,6] = 'Adults'
    if df.iloc[i,6] == '16+' or df.iloc[i,6] == '13+ (PG-13)':
        df.iloc[i,6] = 'Young Adults'
    if df.iloc[i,6] == '7+ (PG)':
        df.iloc[i,6] = 'Kids'
    if df.iloc[i,6] == 'All (G)':
        df.iloc[i,6] = 'General'
df['Rated'].fillna('General', inplace=True) #fillna
print(df['Rated'].unique())
```

✓ 1.1s

```
['Young Adults' 'Kids' 'Adults' 'General']
```

Hình 34: Xử lý biến Rated

-Độ tuổi xem phim ở các quốc gia



Hình 35: Kết quả sau khi xử lý biến Rated

Nhận xét:

- Độ tuổi xem phim nhiều nhất rơi vào độ tuổi thanh niên , trên 18 tuổi (Adults) . Chiếm tỉ lệ cao nhất là nước Mỹ (quốc gia sản xuất phim nhiều nhất thế giới) . Thấp nhất là Ấn Độ.
- Độ tuổi trẻ em chiếm lượt xem nhiều đứng thứ hai.
- Cuối cùng tuổi vị thành nên (Young Adults) là độ tuổi chiếm lượt xem ít nhất.

=> Các nhân tố trên cho thấy rằng , việc đánh giá phim hay hoặc không hay , phổ biến hay không thì độ tuổi cũng là yếu tố có sự ảnh hưởng.

3.1.5 Xử lý biến trống Director

-Giữ lại giá trị Director của top 5 quốc gia sản xuất phim nhiều nhất. Các quốc gia khác đưa về giá trị Other Director

-Cách thực hiện

```
# Xử lý biến trống Director
for i in range(len(df['Director'])):
    if df.iloc[i,4] != 'America' and df.iloc[i,4] != 'United Kingdom' and df.iloc[i,4] != 'Canada' and
        df.iloc[i,9] = 'Other Director'
    else:
        df.iloc[i,9] = df.iloc[i,9]
df['Director'].fillna('Other Director', inplace=True) #fillna
```

Hình 36: Xử lý biến Director

3.2 Label Endcoding

Label endcoding là kỹ thuật chuyển đổi giá trị nhãn sang giá trị số học.

Các biến cần để label endcoding : Country, Source, Rated

Cách thực hiện:

```
# Label encoding
lb = LabelEncoder()
list_feature =['Country', 'Source', 'Rated']
print("Trước khi chuyển đổi:\n{0}".format(df['Rated'].head()))
for item in list_feature:
    lbe = LabelEncoder()
    ds = df[item].values.tolist()
    lbe.fit(ds)
    df[item] = lbe.transform(ds)
```

Hình 37: Xử lý Endcoding

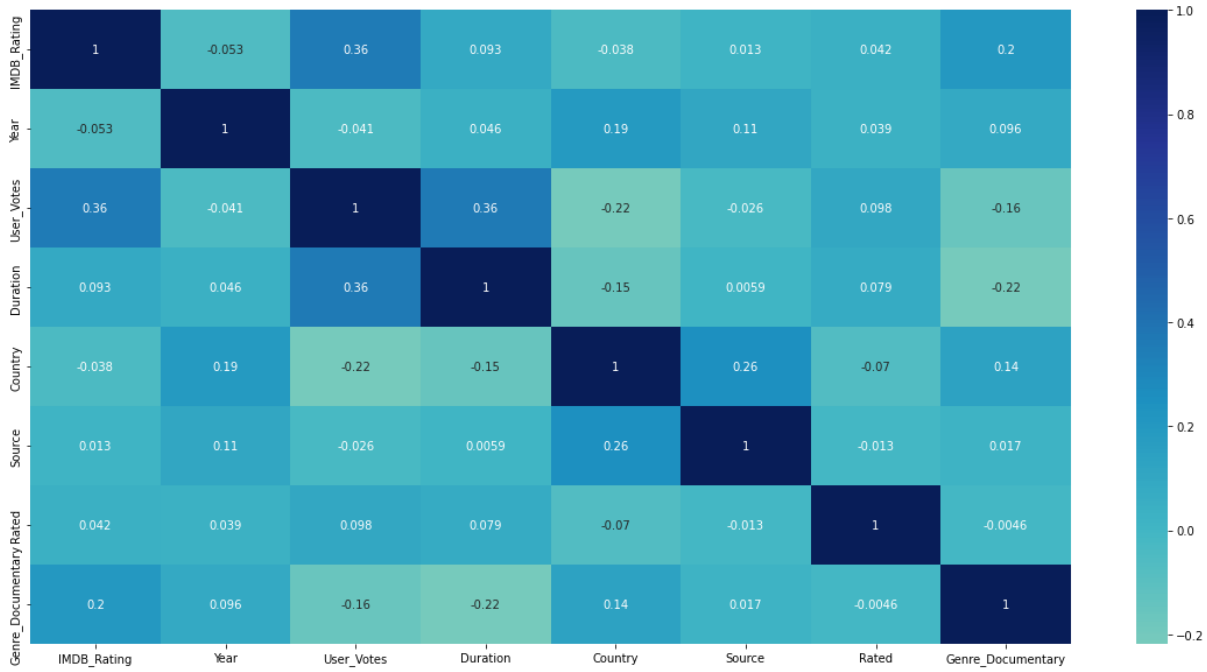
Kết quả:

```
Trước khi chuyển đổi:
0    Young Adults
1            Kids
2    Young Adults
3    Young Adults
4    Young Adults
Name: Rated, dtype: object
Sau khi chuyển đổi:
0    3
1    2
2    3
3    3
4    3
```

Hình 38: Kết quả sau khi Endcoding

3.3 Lựa chọn đặc trưng

Sử dụng đồ thị heatmap để quan sát mối tương quan giữa các biến so với giá trị dự đoán IMDB_Rating



Hình 39: Đồ thị mối tương quan

Các đặc trưng có độ tương quan cao

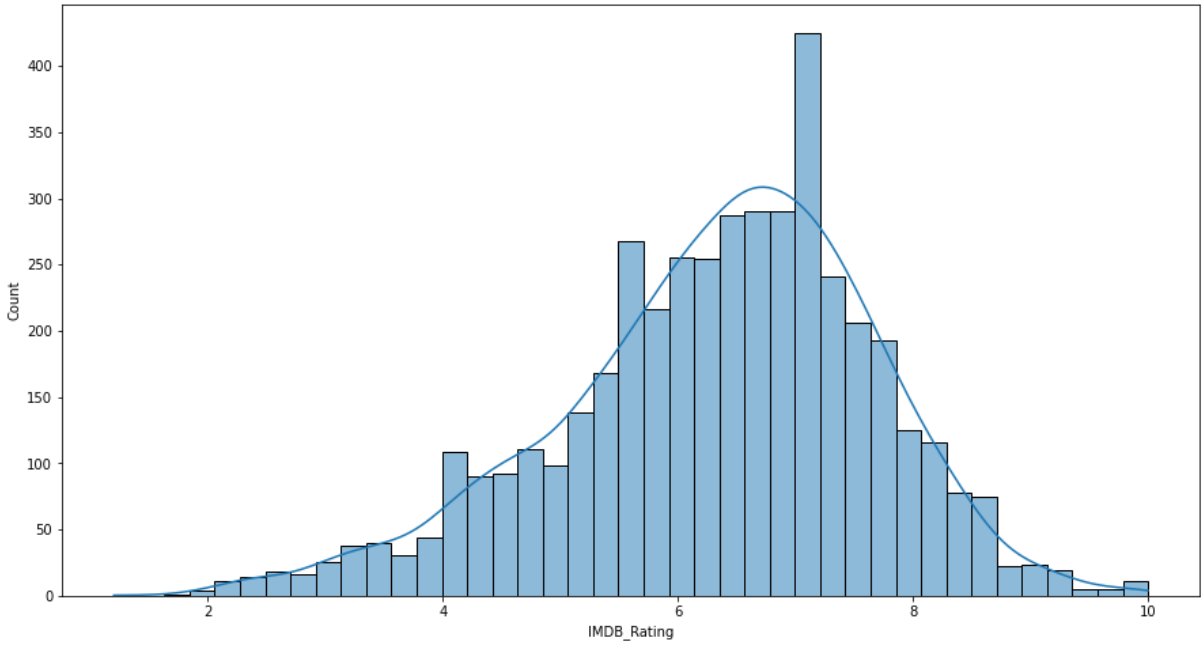
	Đặc trưng có mối tương quan cao
0	IMDB_Rating
1	User_Votes
2	Genre_Documentary
3	Duration
4	Rated

Hình 40: Các đặc trưng có tương quan cao nhất

Nhận xét:

- Sau khi làm sạch dữ liệu thì tương quan giữa các đặc trưng với IMDB_Rating vẫn khá thấp
- User_Votes và Genre_Documentary có độ tương quan tốt nhất với IMDB_Rating

Đặt trưng dự đoán: IMDB_Rating



Hình 41: Phân bố giá trị IMDB_Rating

4. Mô hình hóa dữ liệu

4.1. Lý thuyết về cách giải quyết bài toán và các mô hình được sử dụng

Đối với việc giải bài toán dự đoán điểm theo thang 0 đến 10 dựa trên dữ liệu phim ta thu được bài toán được hướng đến sẽ là Regression. Regression là gì và ý nghĩa Regression Analysis là sao ?

- Regression hay phân tích hồi quy (Regression analysis) là kĩ thuật thống kê dùng để ước lượng phương trình phù hợp nhất với các tập hợp kết quả quan sát của biến phụ thuộc và biến độc lập.

Hình 42: Phân tích hồi quy

- Ý nghĩa bài toán hồi quy : Phân tích hồi qui cho phép đạt được kết quả ước lượng tốt nhất về mối quan hệ chân thực giữa các biến số. Từ phương trình ước lượng được này, người ta có thể dự báo về biến phụ thuộc (chưa biết) dựa vào giá trị cho trước của biến độc lập (đã biết).

Cụ thể hơn các mô hình hồi quy được sử dụng trong bài toán dự đoán bao gồm:

- Linear Regression: Trong thống kê, hồi quy tuyến tính là một cách tiếp cận tuyến tính để mô hình hóa mối quan hệ giữa một phản ứng vô hướng và một hoặc nhiều biến giải thích (còn được gọi là các biến phụ thuộc và độc lập). Mô hình Linear Regression là một mô hình có học có giám sát.

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $^{\mathsf{T}}$ denotes the transpose, so that $\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol{\beta}$.

Often these n equations are stacked together and written in matrix notation as

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

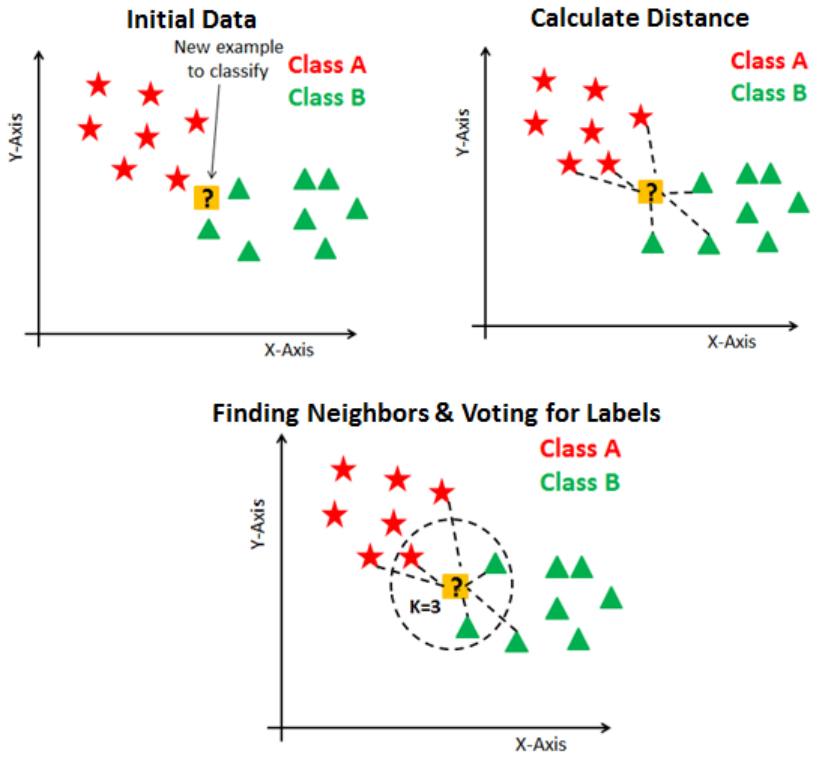
Hình 43: Công thức bài toán Linear Regression

Cụ thể API của mô hình trong thư viện sklearn có dạng: *class sklearn.linear_model.LinearRegression(*, fit_intercept=True, normalize='deprecated', copy_X=True, n_jobs=None, positive=False)*

Với các tham số được cấu hình trên việc ta xây dựng một mô hình Linear Regression tương đối dễ dàng.

- K-nearest neighbors regression: KNN (K-Nearest Neighbors) là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới. Lớp (nhãn) của một đối tượng dữ

liệu mới có thể dự đoán từ các lớp (nhãn) của k hàng xóm gần nó nhất. Bài toán KNN rất linh động có thể sử dụng tốt trong cả qua trình phân lớp và hồi quy tuy thuộc vào mục đích mà người sử dụng hướng tới.

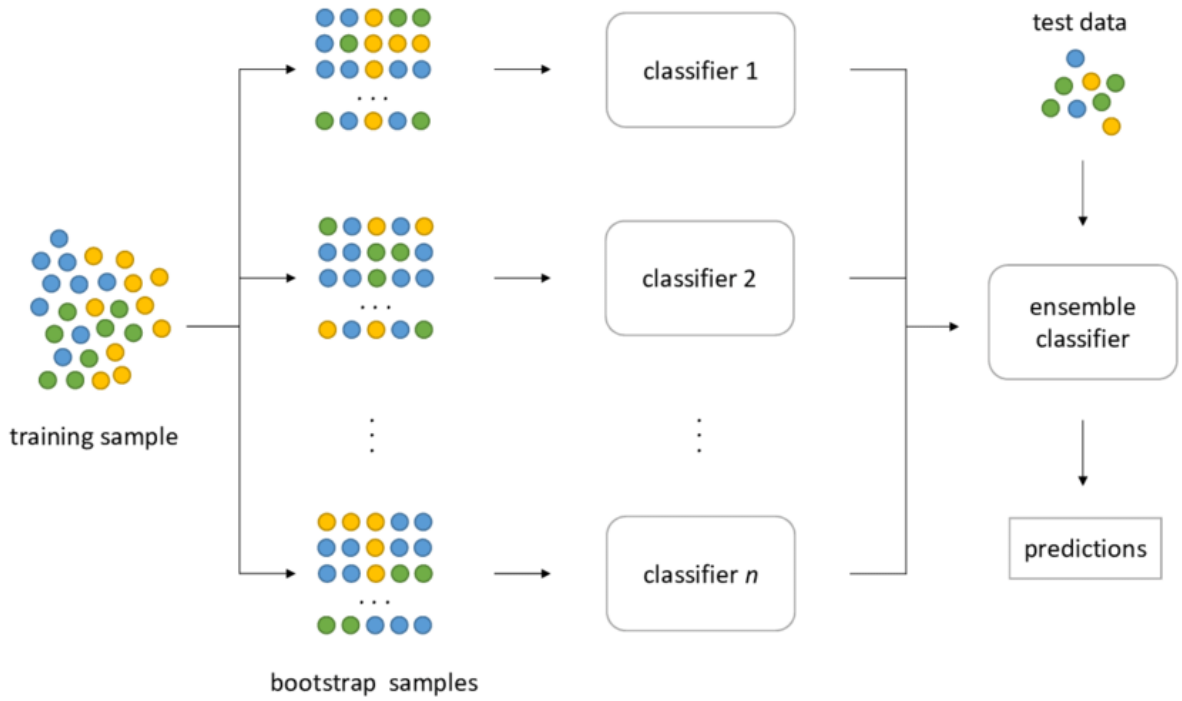


Hình 44: Minh họa của mô hình KNN

Mô hình KNN có API tương đối linh hoạt để có thể tinh chỉnh được sklearn cung cấp : *class sklearn.neighbors.KNeighborsRegressor(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)*

Với cấu trúc dữ liệu cho phép cấu hình nhiều siêu tham số vì thế KNN regression rất linh hoạt và ta có thể sử dụng các cấu hình tham số này để làm tăng độ chính xác và cải thiện chất lượng bài toán.

- Bagging regression : đây được xem như là dạng bài toán bootstrap là thuật toán học máy kiểu tập hợp, hình dùng là mô hình sẽ thực hiện việc lấy mẫu trực tiếp dựa trên một mô hình khác đưa vào trong bao đóng và thực hiện việc bootstrap trên mô hình đó và dùng mô hình thực hiên công việc tương tự là dự đoán.



Hình 45: Minh họa cho mô hình dạng tập hợp cụ thể Bagging regression

API mà sklearn cung cấp cho ta với mô hình này cũng tương đối phức tạp để chọn lựa cho phù hợp : class sklearn.ensemble.BaggingRegressor(base_estimator=None, n_estimators=10, *, max_samples=1.0, max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None, random_state=None, verbose=0)

Với mô hình trên chúng ta nên tập trung vào 2 tham số đầu tiên đó là base_estimator cho ta truyền tham số trực tiếp vào dưới dạng các mô hình và thứ hai là n_estimators là số lượng mà nó sample lên, đây là một công cụ có thể giúp ta giải quyết vấn đề trong bài toán dự đoán nếu muốn nâng cao chất lượng dự đoán của mô hình.

4.2. Giải quyết bài toán

Thực hiện việc chia dữ liệu ta sẽ tiến hành theo 2 công đoạn cụ thể là thể đó là chia dữ liệu Training set và Test set theo tỉ lệ 7 : 3 rồi ta tiếp tục chia dữ liệu thành Training set và Validation set theo tỉ lệ 7 : 3 của Trainning set. Tổng kết ta thu được Train/validation/test theo tỉ lệ 5 : 2 : 3 đây là lượng dữ liệu phù hợp cho bài toán dự đoán. Tất cả các công đoạn này ta sử dụng *train_test_split* của sklearn.

4.2.1. Thực hiện dự đoán với mô hình Linear Regression

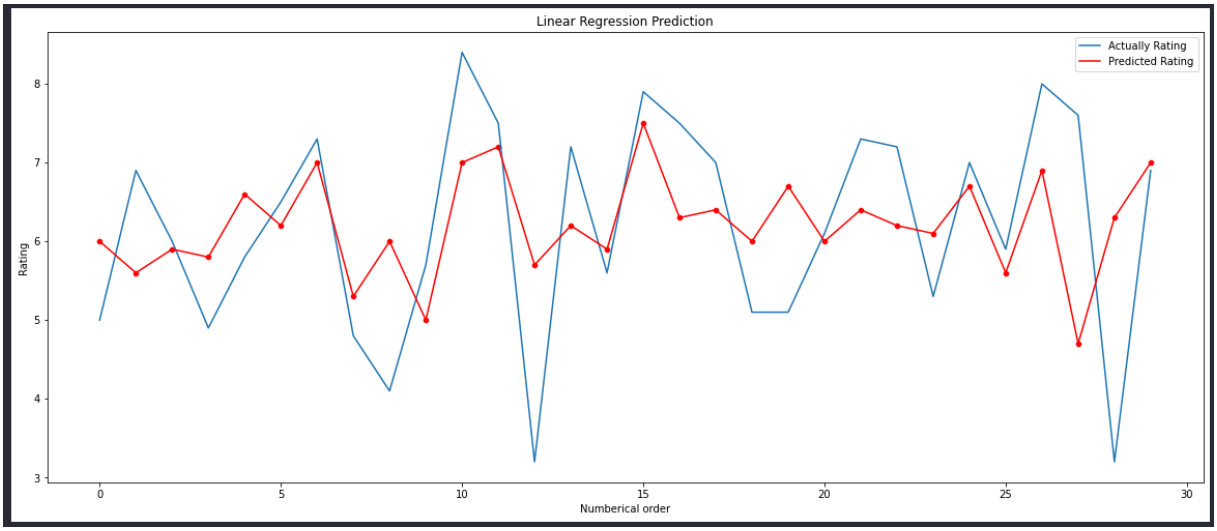
Ta lần lượt đi qua các bài toán với các mô hình và tham số được cấu hình trong LR :

Với Linear Regression (Sử dụng với tham số mặc định) và sử dụng metric : RMSE để đánh giá quá trình train và validation ta thu được

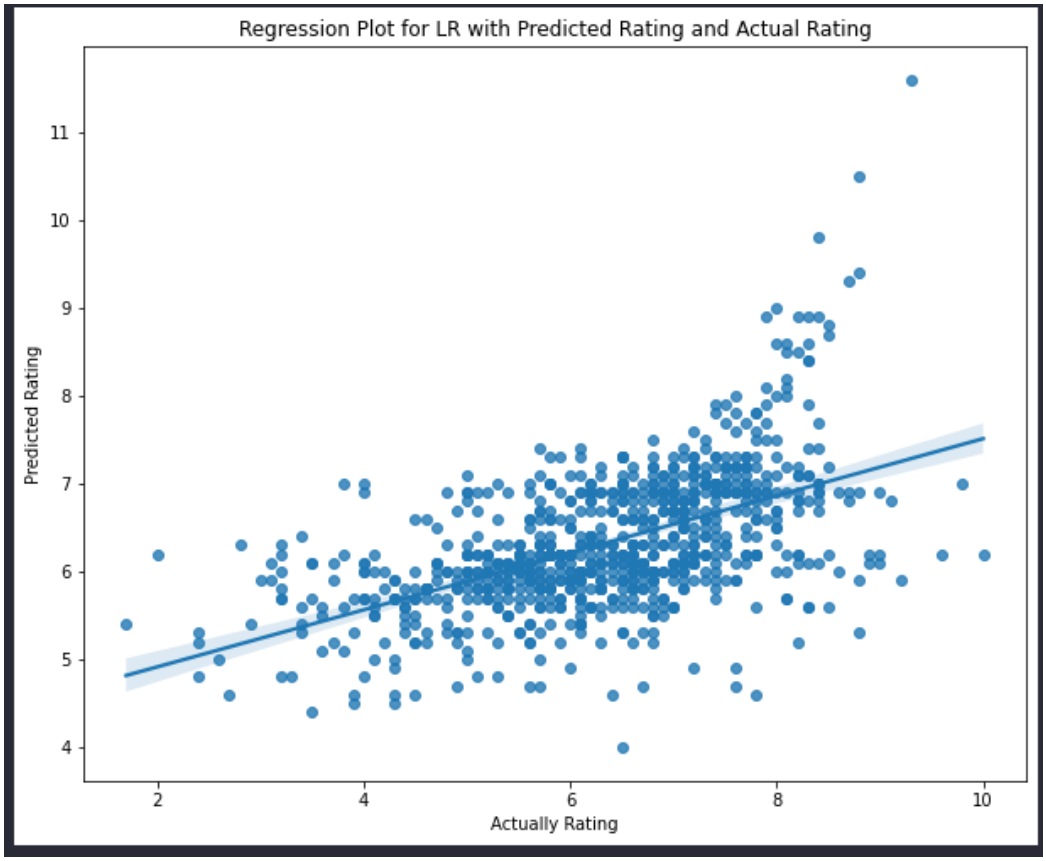
```
# LinerRegression
LR = LinearRegression()
LR.fit(X_train, y_train)

# Predict
predict_train = LR.predict(X_train)
predict_test  = LR.predict(X_test)
RMSE_LR_TRAINING_BASIC = np.sqrt(mean_squared_error(y_train, predict_train))
RMSE_LR_VALID_BASIC = np.sqrt(mean_squared_error(y_test, predict_test))
print("RMSE OF LReg on Training Set: ", RMSE_LR_TRAINING_BASIC)
print("RMSE OF LReg on Validation Set: ", RMSE_LR_TRAINING_BASIC)

RMSE OF LReg on Training Set:  1.1288618519327878
RMSE OF LReg on Validation Set:  1.1288618519327878
```

Hình 46: Kết quả mô hình Linear Regression trên Train and validation set

Sau đây là kết qua áp dụng mô hình trên cho dữ liệu test để xem độ chính xác của dự đoán của bài toán Linear Regression



Hình 47: Kết quả dự đoán với 30 nhãn đầu tiên trong Test set



Hình 48: Kết quả độ hồi quy của dự đoán sử dụng Linear Regression

Nhận xét: Những điểm dự đoán của mô hình Linear rất sát với đường hồi quy cho một Quả tương đối là chính xác, tuy nhiên vẫn còn một số điểm sai lệch và sai lệch lớn đây là vấn đề tất yếu khi dự đoán.

4.2.2. Thực hiện dự đoán với mô hình KNN Regression

Đối với bài toán dự đoán bằng KNN, thì bài toán đặt ra câu hỏi trước tiên là ta cần phải tìm K như thế nào để bài toán đạt được hiểu quả cao nhất?

Có một trích dẫn[2] đối với số lượng K trong KNN thì K được xem là một tham số rất quan trọng, là nhân tố quyết định bài toán. Thông thường thì K là một số lẻ vì nếu số lượng lớp là 2 thì K =1 thì thuật toán hiểu đó là điểm gần kế nhất (Nearest neighbor). Vì thế đó chính là nguyên do mà ta cần phải chọn số lẻ K cho bài toán KNN.

```
1  NUMBER_OF_NEIGHBORS = np.arange(1, 51, 2)
2  RMSE_OF_NEIGHBORS_WITH_TRAIN = []
3  RMSE_OF_NEIGHBORS_WITH_VALIDATION = []
```

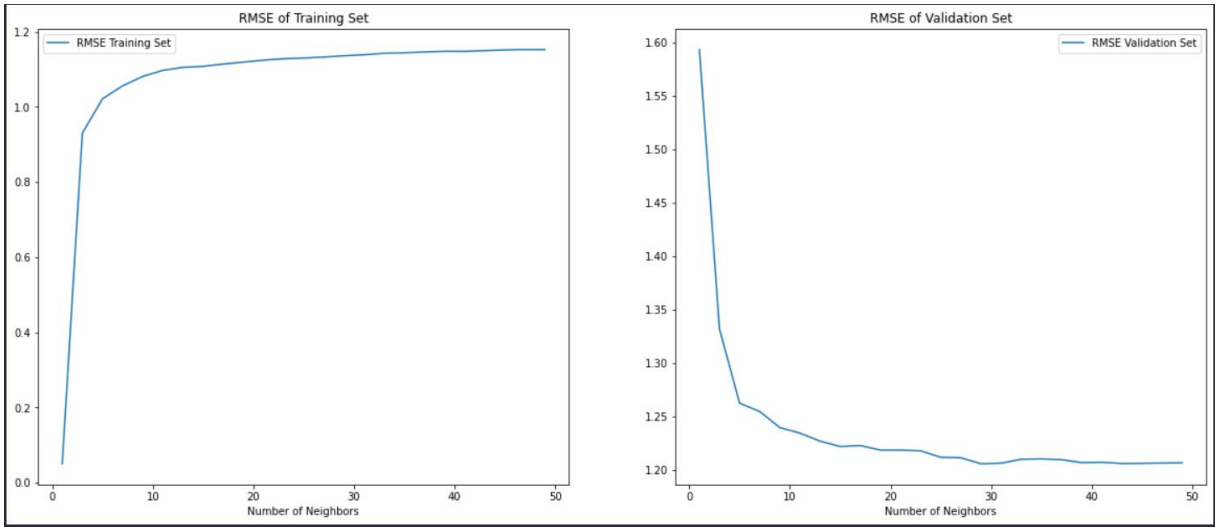
Hình 49: Chọn số lượng K để thực hiện bài toán KNN

Với số lượng một mảng K ta tạo được ta sẽ tiến hành đánh giá thông qua việc train mô hình KNN với số lượng K sau đó kiểm thử và đánh giá trên tập Validation để xem xét đâu là tham số K thích hợp để sử dụng cho bài toán dự đoán trên tập Test.

```
for neighbor in NUMBER_OF_NEIGHBORS:
    model = KNeighborsRegressor(n_neighbors=neighbor)
    model.fit(X_train, y_train)
    # RMSE of training set
    y_train_pred = model.predict(X_train)
    RMSE_OF_NEIGHBORS_WITH_TRAIN.append(np.sqrt(mean_squared_error(y_train, y_train_pred)))
    # RMSE of validation set
    y_valid_pred = model.predict(X_test)
    RMSE_OF_NEIGHBORS_WITH_VALIDATION.append(np.sqrt(mean_squared_error(y_test, y_valid_pred)))
```

Hình 50: Phương pháp dự đoán K cho bài toán KNN

Với thông RMSE dùng để đánh giá đối với bài toán Regression ta sẽ sử dụng các hình ảnh độ biến thiên của thông số đó để trực quan đường elbow của mô hình và xem đâu là kết quả phù hợp cho bài toán dự đoán điểm số.



Hình 51: Kết quả RMSE sau khi ta đánh giá quá trình train và validation với tập K

Nhận xét: Nhìn vào trong hình ảnh ta có thể thấy rằng giá trị RMSE của tập validation có xu hướng giảm với số K tăng dần. Tại K = 11, ta có thể thấy được đường gấp khúc rất rõ ràng vì thế có thể chọn đây là tham số K tốt nhất cho bài toán KNN. Với kết quả trên ta thu được với K = 11 ta thu được lần lượng RMSE trên hai tập là 1.09 (đối với Training set) và 1.23 (đối với Validation Set).

Tuy nhiên với việc nhìn hình bằng mắt thường thì kết quả thường sẽ là ý kiến chủ quan vì thế ta có thể cải thiện việc này bằng cách sử dụng GridSearchCV thay cho đường elbow để tránh cho việc nhập nhằng và gây rối cho người làm làm cho việc đánh giá số

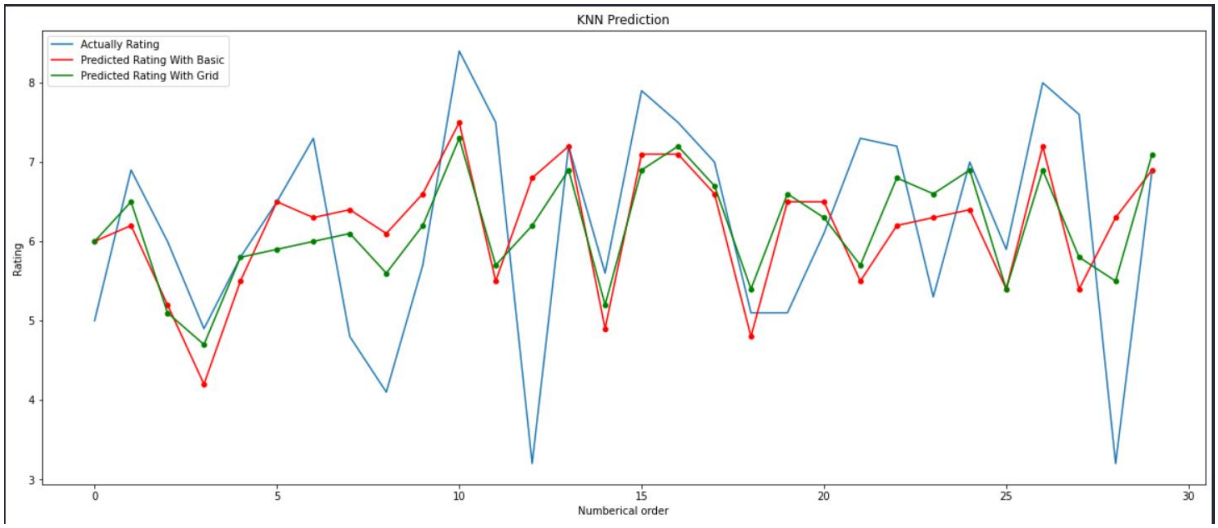
lượng K trở nên khách quan hơn và với GridSearchCV còn giúp ta có thể đánh giá thêm các tham số khác của mô hình KNN với tương đối nhiều tham số tiềm năng cho bài toán.

```
params = {'n_neighbors': NUMBER_OF_NEIGHBORS, 'weights': ['uniform', 'distance']}
knn = KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5, scoring='neg_root_mean_squared_error')
model.fit(X_train, y_train)
best_param =  model.best_params_
print("The best parameters for KNN: ", best_param)
# Prediction on Training set
y_train_pred = model.predict(X_train)
print("RMSE of Training set by GridSearchCV: ", np.sqrt(mean_squared_error(y_train, y_train_pred)))
# Prediction on Validation set
y_valid_pred = model.predict(X_test)
print("RMSE of Validation set by GridSearchCV: ", np.sqrt(mean_squared_error(y_test, y_valid_pred)))
```

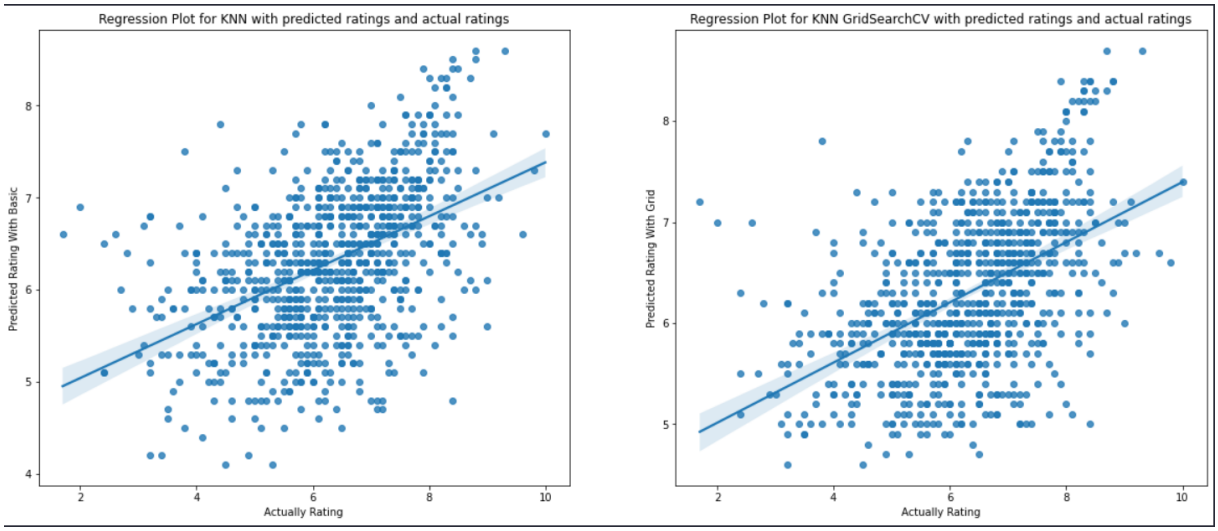
Hình 52: Tìm tham số K bằng GridSearchCV

Kết quả cho ra là K = 49, từ đó ta có thể nhận thấy được sự khác biệt rất rõ ràng trong 2 bài toán bằng việc xem kết quả RMSE lần lượt 0.035 (đối với Train set) và 1.19 (đối với Validation Set). Kết quả rất là tìm năng so với việc dùng thủ công bằng mắt thường để chọn K.

Sau khi chọn được K và các thông số phù hợp cho mô hình KNN Regression ta sẽ tiến hành đánh giá mô hình đó hoạt động như thế nào với Test set.



Hình 53: Kết quả dự đoán 30 nhãn đầu tiên của tập Test với KNN



Hình 54: Kết quả đường hồi quy dự đoán đối với KNN và KNN GridSearch

Nhận xét: Nhìn vào cả hai đồ thị, đối với đồ thị đường thì ta thấy rằng KNN bám sát với dữ liệu ở phần trong tậm với giá trị thực tế còn các điểm cận trên và dưới của dữ liệu thật thì dự đoán kém và kết quả đó được biểu thị khá rõ ràng trong đồ thị hồi quy có thể thấy rằng việc dùng KNN sẽ làm cho kết quả được phân bố rộng hơn tuy nhiên độ hội tụ có thể là mức chấp nhận nhưng vẫn sẽ không tốt hơn so với mô hình LR.

4.2.3. Cải thiện bài toán và kết quả cuối cùng của bài toán dự đoán

Có một trích dẫn[1] đối với một mô hình khi mà chỉ thực hiện 1 lần thì sẽ kém hiểu quả trong việc ta thực hiện tập hợp hàng chục hàng trăm mô hình để đánh giá bài toán thì khi đó kết quả sai sẽ thấp hơn, đó chính là lý do mà ta có thể áp dụng Bagging regression để thực hiện cả thiện bài toán.

Ta sẽ thực hiện trên cả 2 mô hình KNN và LR để xem kết quả hiệu năng bài toán.

```
def BaggingModel(model_estimator, n_estimators = 100):
    bagging_model = BaggingRegressor(model_estimator, n_estimators)
    bagging_model.fit(X_train, y_train)
    y_bagging_pred = bagging_model.predict(data_test)
    rmse_pred =  np.sqrt(mean_squared_error(pred_test, y_bagging_pred))
    return rmse_pred, y_bagging_pred

```

Hình 55: Xây dựng hàm BaggingModel để cải thiện bài toán

Sau khi đánh giá và cải thiện bài toán ta sẽ thu được kết quả như hình để xem các mô hình dự đoán số điểm phim như thế nào?

	Type of Model	RMSE	MAE	Mean Err	Std Err	Result Rating
0	Linear Regression	1.0956	0.8354	0.8360	0.7103	6.3212± 0.5889
1	LR with Bagging	1.0945	0.5935	0.8343	0.7103	6.3312± 0.5929
2	KNN	1.1603	0.8837	0.8846	0.7528	6.3088± 0.6413
3	KNN with GridSearchCV	1.1310	0.8563	0.8570	0.7385	6.3047± 0.6316
4	KNN with GridSearchCV and Bagging	1.1305	0.8613	0.8622	0.7328	6.2998± 0.6242

Hình 56: Kết quả của bài toán dự đoán thông qua 2 mô hình LR và KNN

Nhận xét: có thể thấy được rằng hiệu năng mà 2 mô hình đem lại là khá hiệu quả với độ sai lệch từ 1.09 tới 1.16, có thể thấy đây là một kết quả tạm chấp nhận được nó có thể cho ta cái nhìn tổng quan về 2 bài toán và có thể dữ liệu crawl về phim cho kết quả dự đoán của mô hình Linear là tốt hơn so với mô hình KNN.

5. Kết luận

Tổng quan và cách giải quyết quyết bài toán:

- Nhóm đã thực hiện và hoàn thiện tất cả các yêu cầu tổng quan của bài tập thông qua các cộng đoạn crawl, trực quan hóa, làm sạch dữ liệu và xây dựng mô hình và thực hiện dự đoán dữ liệu (cụ thể là dự đoán đánh giá phim)
- Kết quả các quá trình trong công việc của các thành viên trong nhóm đóng góp và phù hợp đáng kể cho các quá trình tiếp theo cho các thành viên còn lại.
- Thông qua tất cả các quá trình kết quả của quá trình là tương đối tốt với các mô hình hồi quy, tuy nhiên đối với việc sử dụng mô hình hồi quy nào có thể ảnh hưởng lớn tới bài toán.
- Với 2 mô hình mà nhóm sử dụng là Linear Regression và KNN Regression thì kết quả tốt hơn nghiên về với Linear Regression, đối với KNN thì kết quả tương đối phức tạp và số chiều tương đối lớn điều đó làm ảnh hưởng tới quá trình đánh giá của model KNN.
- Tuy nhiên KNN rất linh hoạt và có thể cả thiện hiệu năng rất đáng kể bằng cách sử dụng gridsearch để tìm tham số phù hợp và tương đối ít với Bagging regression để có thể đánh giá được trung bình sau 100 lần mô hình hóa dữ liệu KNN. Còn phần LR thì thụ động hơn với việc sử dụng bộ tham số mặc định bằng cách thay đổi siêu tham số, nếu ta sử dụng Bagging thì kết quả cải thiện tương đối ít như KNN.

- Nhìn chung thì Linear đem lại hiệu quả tốt nhưng về việc cải thiện là chưa có nhiều cách để giải quyết do đó nó thụ động so với lại KNN.

Phương hướng cải thiện:

- Về hướng giải quyết đánh giá và dự đoán thì ta có thể cải thiện bằng cách xây dựng các hàm tính khoảng cách và trọng số phù hợp cho KNN như Gaussian Weighting hay là các công thức tính khoảng cách phù hợp cho dữ liệu.
- Tối ưu hóa về các siêu tham số trong các hàm dựng mô hình để kết quả cho ra hiểu quả tốt hơn và có thể thực hiện giảm chiều hoặc tối ưu hóa dữ liệu để bài toán KNN có thể tốt hơn.
- Nếu muốn cải thiện bài toán dự đoán trên dữ liệu này ta nên sử dụng các mô hình mạng neural hay mô hình xác suất để cho nó bao quát tất cả dữ liệu và cho nó học được dữ liệu đó thì điều đó sẽ làm cho bài toán tốt hơn so với các mô hình supervise như KNN hay là LR.

6. Tài liệu tham khảo

[1] Realpython, "*The k-Nearest Neighbors (kNN) Algorithm in Python*",2021. [Trực tuyến]. Địa chỉ The k-Nearest Neighbors (kNN) Algorithm in Python – Real Python

[2] Analyticsvidhya, "*A Practical Introduction to K-Nearest Neighbors Algorithm for Regression (with Python code)*",2021.[Trực tuyến] . Địa chỉ K-Nearest Neighbors Algorithm | KNN Regression Python (analyticsvidhya.com)

[3] phamdinhkhanh, "*Thiết kế pipeline*", 2021. [Trực tuyến]. Địa chỉ 6.1. Thiết kế pipeline — Deep AI KhanhBlog (phamdinhkhanh.github.io)

[4] phamdinhkhanh, "*Feature Engineering*",2021. [Trực tuyến]. Địa chỉ 11.1. Feature Engineering — Deep AI KhanhBlog (phamdinhkhanh.github.io)

[5] Phạm Minh Tuấn, "*Slide04-Khoahocdulieu-Thu thập và lưu trữ dữ liệu*",2022